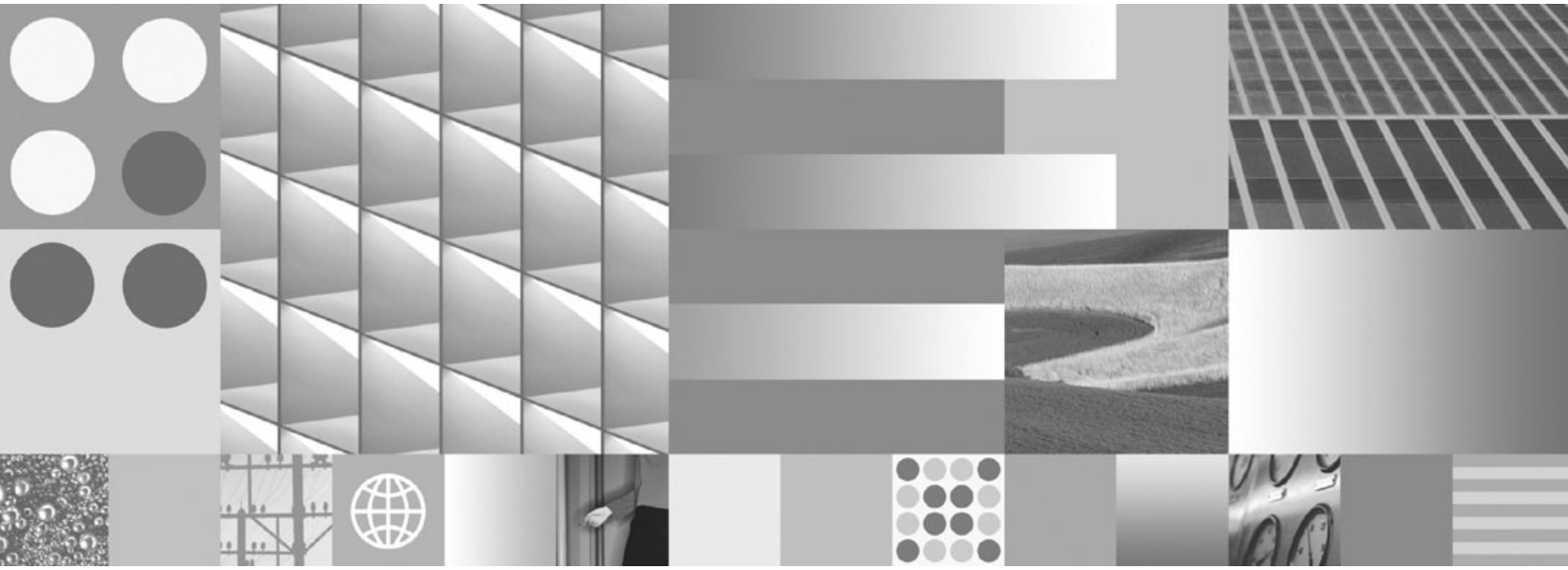
Active Correlation Technology

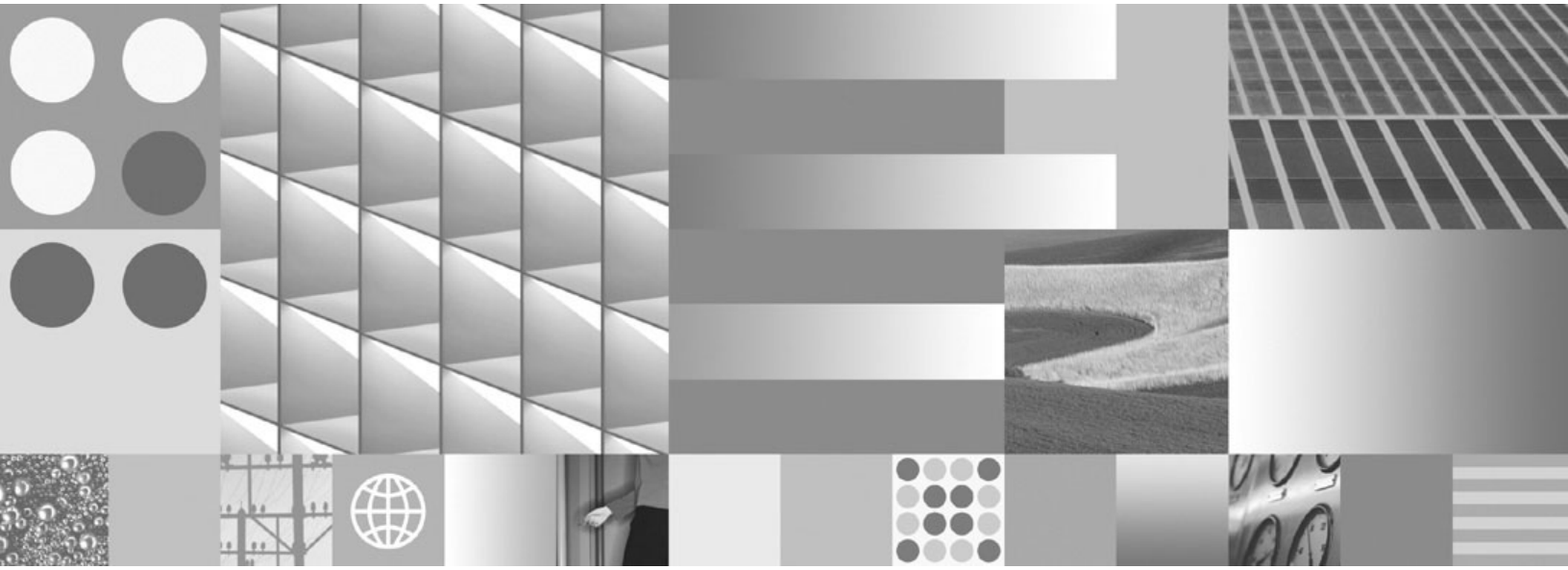
Version 6.0



Handbuch und Referenz für Regelautoren

Active Correlation Technology

Version 6.0



Handbuch und Referenz für Regelautoren

Hinweis

Vor Verwendung dieser Informationen und des darin beschriebenen Produkts sollten die Informationen unter „Bemerkungen", auf Seite 125 gelesen werden.

Erste Auflage (Juni 2006)

Informationen zu dieser Dokumentation

Dieses Dokument bietet eine Übersicht über IBM Active Correlation Technology (ACT) und ihre Bedeutung für die Verarbeitung komplexer Ereignisse. Die ACT-Regelsprache ist eine XML-basierte Sprache zum Schreiben von Regeln für die Ereigniskorrelation. Dieses Dokument ist für Regelautoren gedacht, die verstehen wollen, wie Regeln zur Ereigniskorrelation für ihr Unternehmen geschrieben werden.

Inhaltsverzeichnis

Teil 1. Handbuch für Regelautoren

Kapitel 1. Einführung

Diese Einführung bietet eine kurze Beschreibung zur Verarbeitung komplexer Ereignisse (Complex Event Processing, CEP) und eine Übersicht über Active Correlation Technology (ACT) und ihre Bedeutung für die Verarbeitung komplexer Ereignisse.

Das heutige Geschäftsumfeld

Wirtschafts- und Regierungsorganisationen sind heutzutage abhängig von der elektronischen Informationsverarbeitung über Computernetze insbesondere über das Internet. Mit zusätzlichen Technologien wie beispielsweise Grid-Computing führen die Organisationen geschäftskritische Anwendungen zu jeder Zeit und überall in der Welt aus. Geschäftsprozesse, Aktivität und Infrastruktur und dadurch unsere weltweite Gesellschaft sind abhängig von der Informationstechnologieebene von Organisationen.

Die Organisationen müssen jederzeit wissen, was in ihren Geschäftsbereichen geschieht. Geschäftskritische Anwendungen müssen beispielsweise verfügbar und uneingeschränlt betriebsfähig sein, und mögliche Probleme bei Geschäftsprozessen, Aktivität oder Infrastruktur müssen erkannt und verhindert werden. Im Falle eines Problems muss der Fehler sofort verstanden und behoben werden, und die Ursache muss ermittelt werden.

Die Bedeutung der meisten Ereignisse, die mit Geschäftsprozessen, Aktivität und Infrastruktur einhergehen, wird nie erkannt oder verstanden, da der Umfang an Informationen zu groß und schwierig zu verarbeiten ist, weil sie in einzelnen, zusammenhanglosen Stücken vorhanden sind. Wenn die Ereignisse aber zusammengefasst und korreliert werden, so dass ihre Beziehung leicht verstanden werden kann, können sie eine Fülle von Informationen hervorbringen.

Die Verarbeitung komplexer Ereignisse dient dazu, in Echtzeit bessere Informationen zu Ereignisse zu erhalten.

Verarbeitung komplexer Ereignisse

Ein *Ereignis* ist einfach eine Benachrichtigung über eine Situation, die aufgetreten ist.

Die *Verarbeitung komplexer Ereignisse* ist die Ableitung problemorientierter Ereignisse von der Analyse, Korrelation und Zusammenfassung von Ereignissen der unteren Ebene in ereignisgesteuerten Systemen. Diese problemorientierten Ereignisse, so genannte komplexe Ereignisse, sind geeignet, um Personen in leicht verständlichen Begriffen auf Geschäftschancen oder -probleme hinzuweisen oder um automatisierter Prozesse auszulösen. Organisationen können dann effizienter arbeiten mit einer Frühwarnung vor möglichen Gelegenheiten oder Problemen und mit einem besseren Verständnis der eigentlichen Ursachen, die die Zustände in ihren Geschäftsprozessen bzw. in ihrer Aktivität oder in ihrer Infrastruktur ändern.

Ereigniskorrelation ist der Prozess des Definierens und Ermittelns von Mustern in Ereignisströmen in Echtzeit und des Implementierens von Aktionen als Antwort auf zusammengehörige Ereignisse. Sie wird zur Identifizierung eines Problems, basierend auf den ermittelten Symptomen, verwendet. Ereignisse können durch

Ursache, Zeit, Zugehörigkeit oder eine beliebige Kombination dieser Punkte korreliert werden. Ereigniskorrelation ist ein integraler Bestandteil bei der Verarbeitung komplexer Ereignisse.

Active Correlation Technology

Active Correlation Technology verwendet Regeln zur Ermittlung von Mustern in Ereignisströmen in Echtzeit. Diese Technologie basiert auf der Voraussetzung, dass Antwortaktionen häufig nicht durch ein einzelnes Ereignis der unteren Ebene ausgelöst werden sollten, sondern durch eine komplexe Zusammensetzung von Ereignissen, die zu unterschiedlichen Zeiten und in unterschiedlichen Kontexten auftreten. Active Correlation Technology verwendet die Beziehungen unter Ereignissen, um das Vorhandensein von Geschäftschancen und Problemen anzuzeigen. Basierend auf dem Geschäftsverständnis, das über die Korrelation von Ereignissen in Echtzeit erlangt wurde, kann eine Organisation beispielsweise die folgenden Arten von Aktionen ausführen:

- Angebot reduzierter Versandkosten für einige oder alle Kunden bei einem Einkauf während einer Sonderaktion.
- Kalkulieren der Versandkosten über die nächsten 30 Tage basierend auf dem Transportunternehmen, dem Auftragsbetrag und der Auftragsmenge.
- Kunden, die zwischen 1. Juli 2005 und 31. Dezember 2005 Waren im Wert von mehr als 500 Euro gekauft haben, einen Geschenkgutschein über 25 Euro schicken.
- Benachrichtigen eines Administrators, wenn eine Auftragsabwicklung nicht innerhalb von 36 Stunden abgeschlossen ist.
- Benachrichtigen eines Administrators, wenn innerhalb von 30 Sekunden mehr als vier Anmeldeversuche auf dem gleichen Computer festgestellt wurden.

Active Correlation Technology besteht aus den folgenden primären Elementen:

ACT-Regelsprache (Active Correlation Technology Rule Language)
Eine XML-basierte Sprache für das Schreiben von Regeln zur Korrelation von Ereignissen. Diese Regeln können anschließend in einer ACT-Laufzeitumgebung eingesetzt werden.

ACT-Engine (Active Correlation Technology Engine)
Die Komponente von Active Correlation Technology, die Ereignisse entsprechend der Ausgabe des ACT-Compilers verarbeitet.

ACT-Regelerstellungsprogramm (Active Correlation Technology Rule Builder)
Eine GUI für das Schreiben von Korrelationsregeln in der ACT-Regelsprache.

Eine ACT-Laufzeitumgebung ist eine Anwendung, in die die ACT-Engine eingebettet ist.

Kapitel 2. Übersicht über die Regelsprache

In dieser Übersicht werden die Schlüsselkonzepte der ACT-Regelsprache beschrieben.

Ein Regelmuster ist die Darstellung einer Ereigniskorrelationssituation (wie z. B. eine Schwellenwertbedingung oder eine Erkennung von duplizierten Ereignissen). Die ACT-Regelsprache schließt sieben bewährte Regelmuster ein, die den meisten Ereigniskorrelationssituationen entsprechen, die IBM Kunden adressieren müssen. Sechs der sieben Regelmuster definieren statusabhängige Regeln, und ein Muster definiert eine statusunabhängige Regel.

Statusabhängige Regeln korrelieren mehrere Ereignisse, die während eines bestimmten Zeitraums auftreten, und generieren eine Antwort auf diese Ereignisse. Statusunabhängige Regeln verarbeiten nur ein einzelnes Ereignis, das einer bestimmten Bedingung entspricht, und generieren eine Antwort auf dieses Ereignis.

Statusabhängige Regel (Stateful Rule)
Eine Regel, die Statusinformationen festhält, die Informationen zu Merkmalen einer Regelinstanz darstellen, um für eine Gruppe von Ereignissen über einen Zeitraum ausgeführt zu werden. Regeln, die durch eines der folgenden Regelmuster definiert sind, sind statusabhängige Regeln: Erfassungsmuster, Berechnungsmuster, Duplikatmuster, Sequenzmuster, Schwellenwertmuster oder Zeitgebermuster.

Statusunabhängige Regel (Stateless Rule)
Eine Regel, die keine Statusinformationen festhält und somit nur für jeweils ein Ereignis gleichzeitig ausgeführt werden kann. Eine Regel, die durch das Filtermuster definiert ist, ist eine statusunabhängige Regel.

Zugehöriger Verweis

Kapitel 4, „Organisation eines Regelsatzes - Zusammenfassung", auf Seite 47
Diese Referenz listet alle Sprachelemente eines Regelsatzes, eines Regelblocks und jedes Regeltyps auf. Sie dient als Kurzübersicht zum Codieren eines Regelsatzes.

Kapitel 5, „Sprachelementreferenz", auf Seite 59
Diese Referenz beschreibt die Details der Sprachelemente im XML-Schema für die ACT-Regelsprache. Die Sprachelemente sind in alphabetischer Reihenfolge aufgeführt, und die Attribute, die für die einzelnen Elemente verfügbar sind, werden im Abschnitt für das jeweilige Element beschrieben.

Kapitel 6, „Glossar", auf Seite 121
Dieses Glossar enthält die Begriffe und Definitionen für wichtige Konzepte in Active Correlation Technology.

Aufbau einer Regel

Die grundlegendsten Teile einer Regel sind die Ereignisauswahl, der Gruppierungsschlüssel, das Zeitfenster für statusabhängige Regeln, die Regelantwort, das Aktivierungsintervall und die Lebenszyklusaktionen. Eine Regel enthält auch Ausdrücke und Variablen. Ein Ausdruck ist Code, der eine angepasste Logik enthält, die einer Regel hinzugefügt werden kann.

Ereignisauswahl

Ereignisauswahlkriterien bestimmen, welche Ereignisse für die Verarbeitung durch die Regel akzeptiert werden. Das Element <eventSelector> definiert die Ereignisauswahlkriterien für eine Regel. Die Ereignisauswahl gilt für alle Regeln, mit Ausnahme der Regeln, die durch das Zeitgebermuster definiert werden. Da die Zeitgeberregel keine Ereignisse verarbeitet, enthält sie keine Ereignisauswahlkriterien.

Gruppierungsschlüssel

Normalerweise verfügt jede aktive Regel über eine Regelinstanz (oder Kopie), die in der ACT-Engine ausgeführt wird. Manchmal ist allerdings dieselbe Regel für verschiedene Ereignisgruppen erforderlich, die oft zu verschiedenen Ressourcengruppen gehören. Der Gruppierungsschlüssel ist eine Methode, eine Regel anzuweisen, eine separate Regelinstanz (oder eine Kopie von sich selbst) für jede Gruppe von Ereignissen zu erstellen, die allgemeine Merkmale gemeinsam verwenden.

Der Gruppierungsschlüssel dient als zusätzliche Form der Ereignisauswahl. Wenn eine Regel mit einem Gruppierungsschlüssel definiert wird und wenn die Regel ein Ereignis mit dem Merkmal empfängt, das vom Gruppierungsschlüssel definiert wird, wird das Ereignis an die Regelinstanz gesendet, die die Ereignisse verarbeitet, die dieses Merkmal gemeinsam verwenden. Sie können beispielsweise eine Regel definieren, die alle sicherheitsrelevanten Ereignisse des Typs `Audit Failure` erfasst, und den Gruppierungsschlüssel als Hostnamensattribut eines Ereignisses definieren. Die Regel kann nun mehrmals verwendet werden, wobei eine separate Kopie der Regel für jeden eindeutigen Wert des Hostnamensattributs ausgeführt wird. Sie können außerdem alle Systeme überwachen, die das Ereignis `Audit Failure` empfangen, um festzustellen, ob mehr als 10 solcher Ereignisse in einem Zeitraum von 2 Minuten für die einzelnen Hostnamen auftreten.

Das Element <groupingKey> definiert den Gruppierungsschlüssel für eine Regel, und es gilt für die Regeln, die von den Erfassungs-, Berechnungs-, Duplikat-, Sequenz- und Schwellenwertmustern definiert werden.

Zeitfenster für statusabhängige Regeln

Da statusabhängige Regeln mehrere Ereignisse korrelieren, die während eines bestimmten Zeitraums auftreten, ist ein grundlegender Teil einer statusabhängigen Regel das Zeitfenster, das durch das Element <timeWindow> definiert wird. Das Zeitfenster gibt den Zeitraum an, in dem die statusabhängige Regel verarbeitet wird, um ihrem Muster zu entsprechen.

Regelantwort

Regelantwortaktionen definieren die Aktionen, die ausgeführt werden sollen, wenn die Verarbeitung der Regel beendet ist. Jedes der folgenden Sprachelemente definiert einen anderen Typ von Regelantwortaktion:

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

Welche Typen von Regelantwortaktionen für eine Regel verfügbar sind, hängt vom Regelmuster ab.

Aktivierungsintervall

Das Aktivierungsintervall definiert, wann eine Regel aktiv und inaktiv ist. Das Element <activationInterval> definiert das Aktivierungsintervall für eine Regel.

Eine Regel kann zu einem diskreten Zeitpunkt oder durch ein bestimmtes Ereignis aktiviert oder inaktiviert werden.

Wenn Sie angeben, dass eine Regel zu einem diskreten Zeitpunkt *und* durch ein bestimmtes Ereignis aktiviert oder inaktiviert werden soll, wird die Regel aktiviert oder inaktiviert, sobald der Zeitpunkt erreicht oder das Ereignis empfangen wird, je nach dem, was zuerst eintritt. In diesem Fall wird die Regel während ihres Lebenszyklus jedoch möglicherweise durch eine Vielzahl Ereignisse aktiviert oder inaktiviert. Zum Beispiel wird eine Regel möglicherweise durch ein Ereignis aktiviert, inaktiviert, zu einem definierten Zeitpunkt aktiviert, erneut inaktiviert und durch ein weiteres Ereignis aktiviert.

Das Element <activationByGroupingKey> ist ein Element, das im Element <activationInterval> enthalten ist. Das Element <activationByGroupingKey> enthält Elemente, die die Ereignisse angeben, die eine Regelinstanz aktivieren und inaktivieren können, die durch das Element <groupingKey> definiert wird.

Lebenszyklusaktionen

Die Lebenszyklusaktionen definieren die Aktionen, die in den folgenden vier Hauptphasen im Lebenszyklus einer Regel ausgeführt werden sollen: Laden, Aktivierung, Inaktivierung und Entladen.

Das Element <lifeCycleActions> enthält die folgenden Elemente, die diese Aktionen definieren:

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

Lebenszyklus einer Regel

Jede Phase im Lebenszyklus einer Regel kann mehrere Ursachen und Auswirkungen haben. Durch Schreiben und Einschließen von Ausdrücken in Lebenszyklusaktionen (durch das Element <lifeCycleActions> definiert) kann der Autor der Regeln die Aktionen definieren, die in jeder Phase ausgeführt werden sollen.

Phasen im Lebenszyklus einer Regel

Der Lebenszyklus einer Regel umfasst die folgenden vier primären Phasen:

Laden Das Laden der Regel in die aktive ACT-Engine, die die Aktionen im Element <onLoad> auslöst.

Aktivierung
Die Aktivierung der Regel, die die Aktionen im Element <onActivation> auslöst.

Inaktivierung
Die Inaktivierung der Regel, die die Aktionen im Element <onDeactivation> auslöst.

Entladen
Das Entladen der Regel aus der aktiven ACT-Engine, die die Aktionen im Element <onUnload> auslöst.

Die Phasen der Aktivierung und der Inaktivierung können im Lebenszyklus einer Regel mehrfach vorkommen, die Phasen des Ladens und Entladens jedoch nur einmal.

Normalerweise müssen Sie keine Lebenszyklusaktionen definieren. Es folgen Beispiele zum Definieren einer besonderen Lebenszyklusaktion:

- Wenn eine bestimmte Regel geladen wird, könnten Sie eine Verbindung zu einem externen System (wie z. B. einem Datenbankmanager) herstellen wollen, auf das in dieser Regel zugegriffen werden muss. Wenn diese Regel entladen wird, soll die Verbindung unterbrochen werden, und es sollen erforderliche Bereinigungsprozesse ausgeführt werden.
- Wenn eine bestimmte Regel aktiviert wird, möchten Sie möglicherweise prüfen, ob bestimmte Ressourcen für diese Regel verfügbar sind.
- Wenn eine Schwellenwertregel inaktiviert wird, der Schwellenwert aber nicht erreicht wurde und der Zeitraum noch nicht verstrichen ist, möchten Sie möglicherweise eine Nachricht mit diesen Informationen an jemanden weiterleiten.

Da die Aktivierung und Inaktivierung einer Regel mehrmals im Lebenszyklus auftreten kann, können beliebige Aktionen, die Sie für diese Phasen codieren, öfters ausgeführt werden.

Ursachen und Auswirkungen jeder Lebenszyklusphase

Tabelle 1 listet die Ursachen und Auswirkungen jeder Lebenszyklusphase auf.

Tabelle 1. Ursachen und Auswirkungen jeder Lebenszyklusphase

Lebenszyklusphase	Ursachen	Auswirkungen
Laden	Beliebige folgende Umstände: • Eine Regel oder ein Regelblock wird hinzugefügt oder ersetzt, wodurch die neue Regel bzw. die neuen Regeln geladen werden. • Der Regelsatz wird in der ACT-Engine ersetzt, wodurch die Regeln im neuen Regelsatz geladen werden.	Die Aktionen im Element <onLoad> werden ausgeführt.
Aktivierung	Die Regel wird aktiviert. Eine Regel kann wie folgt aktiviert werden: • Entsprechend den Definitionen im Element <activationInterval> • Durch die Methode „activate()", die über die Variable „act_lib" verfügbar ist • Durch Anwendungsaufrufe an die Methode „activate()" in der ACT-Engine	Wenn die Regel inaktiv ist, werden die Aktionen im Element <onActivation> ausgeführt.

Tabelle 1. Ursachen und Auswirkungen jeder Lebenszyklusphase (Forts.)

Lebenszyklusphase	Ursachen	Auswirkungen
Inaktivierung	Die Regel wird inaktiviert. Eine Regel kann wie folgt inaktiviert werden: • Entsprechend den Definitionen im Element <activationInterval>, mit der Ausnahme, dass das Element <deactivateOnEvent> im Element <activationByGroupingKey> keine Inaktivierung der Regel bewirkt • Durch die Methode „deactivate()", die über die Variable „act_lib" verfügbar ist • Durch Anwendungsaufrufe an die Methode „deactivate()" in der ACT-Engine	Wenn die Regel aktiv ist, werden die Aktionen im Element <onDeactivation> ausgeführt.
Entladen	Beliebige folgende Umstände: • Die ACT-Engine wird beendet, wodurch die Regeln entladen werden. • Eine Regel oder ein Regelblock wird entfernt oder ersetzt, wodurch die alte Regel bzw. die alten Regeln entladen werden. • Der Regelsatz wird in der ACT-Engine gelöscht oder ersetzt, wodurch die Regeln im alten Regelsatz entladen werden.	Wenn die Regel aktiv ist, werden die Aktionen im Element <onDeactivation> und anschließend die Aktionen im Element <onUnload> ausgeführt. Anderenfalls werden nur die Aktionen im Element <onUnload> ausgeführt.

Organisation von Regeln

Die ACT-Regelsprache fasst Regeln zu Regelblöcken zusammen, die Bestandteil eines Regelsatzes sind.

Regelsatz

Der Regelsatz enthält die in Regelblöcke zusammengefassten Regeln, die von einer ACT-Engine ausgeführt werden sollen. Er ist die Regelausführungseinheit. Jede ACT-Engine führt zu jedem Zeitpunkt nur einen Regelsatz aus.

Die Regeln in einem Regelsatz werden durch Ereignisse ausgelöst, die an die ACT-Engine gesendet werden. Die Ereignisse werden nacheinander an die entsprechenden Regeln zugestellt, basierend auf den Auswahlkriterien für Ereignisse in jeder Regel, und zu jedem Zeitpunkt wird nur eine Regel ausgeführt. Das gleiche Ereignis kann auf mehrere Regeln zutreffen und sie somit auslösen. Diese Regeln müssen nicht notwendigerweise zusammengehörig sein.

Die Reihenfolge der Regelblöcke und Regeln in einem Regelsatz legen fest, wie Ereignisse durch den Regelsatz laufen.

Variablen und Importe können auf der Ebene des Regelsatzes definiert werden, damit sie in Ausdrücken (Code mit angepasster Logik) im gesamten Bereich des Regelsatzes verwendet werden können. Ein Import ist eine Möglichkeit, über eine Programmiersprache auf externen Code zuzugreifen. Der Autor einer Regel kann einen Import definieren, um externe Module (wie z. B. Java-Klassen) zur Verwendung in Ausdrücken in Regeln zu importieren.

Regelblock

Der Regelblock ist eine Organisationseinheit zur Gruppierung von Regeln nach Funktion in Domänen des Regelsatzes. Eine Domäne ist die Kategorie, auf die eine Gruppe von Regeln angewendet wird, basierend auf ihren Funktionen. Eine Domäne kann z. B. ein bestimmtes geographisches Gebiet, einen IT-Managementfachbereich (wie z. B. Sicherheitserkennung oder Netzereigniskorrelation) oder eine Geschäftsorganisation (wie z. B. ein bestimmtes Unternehmen oder eine Abteilung in einem Unternehmen) darstellen.

Regelblöcke können Regeln und weitere Regelblöcke enthalten. Da Regelblöcke verschachtelt sein können, kann eine Regelhierarchie erstellt werden. Ein Regelsatz könnte z. B. einen Regelblock für eine Netzereigniskorrelation enthalten und dieser wiederum zwei weitere Regelblöcke: einen für eine Ebene-2-Korrelation und einen für eine IP-Korrelation.

Deshalb stellt ein Regelsatz die Funktionalität für die Ereigniskorrelation für viele Domänen bereit, und ein Regelblock stellt die Organisation für diese unterschiedlichen Domänen bereit, die Zugriff auf eine ähnliche Ereignisgruppe benötigen könnten.

Variablen und Importe können auf der Ebene der Regelblöcke definiert werden, damit sie in Ausdrücken im gesamten Bereich des Regelblocks verwendet werden können. Der Bereich eines Regelblocks schließt beliebige Regeln und weitere Regelblöcke ein, die im Regelblock enthalten sind.

Regel

Die Regel ist die Korrelationseinheit, die zum Erkennen der Beziehungen zwischen Ereignissen und zum Ausführen der geeigneten Regelantworten verwendet wird. Eine Regel ist eine Implementierung eines der folgenden sieben Regelmuster. Sie ist je nach Funktion in einem Regelblock organisiert, der Teil eines Regelsatzes ist:

- Erfassungsmuster
- Berechnungsmuster
- Duplikatmuster
- Filtermuster
- Sequenzmuster
- Schwellenwertmuster
- Zeitgebermuster

Jede Regel kann eindeutige Funktionalität für die Ereigniskorrelation je nach Muster bereitstellen, und Regeln können durch Ereignisweiterleitung verkettet werden. Durch dieses Verketten von Regeln kann die Funktionalität für die Ereigniskorrelation verschiedener Muster kombiniert oder verschachtelt werden.

Variablen können auf der Regelebene definiert werden, damit sie in Ausdrücken im gesamten Regelbereich verwendet werden können.

Zusammenfassung

Der Regelsatz ist die Ausführungseinheit, der Regelblock ist die Organisationseinheit, und die Regel ist die Korrelationseinheit. Ein Regelsatz enthält mindestens einen Regelblock, der weitere Regelblöcke enthalten kann. Jeder Regelblock enthält Regeln für eine bestimmte Domäne. Regelblöcke können

verschachtelt sein, um eine Regelhierarchie zu bilden. Die Reihenfolge der Regelblöcke und Regeln in einem Regelsatz legen fest, wie Ereignisse durch den Regelsatz laufen.

Variablen und Importe können auf der Ebene des Regelsatzes oder der Regelblöcke definiert werden, damit sie in Ausdrücken von Regeln verwendet werden können. Der Bereich der Variablen oder des Imports ist der jeweilige Regelsatz oder Regelblock. Variablen können auch auf Regelebene definiert werden, ihr Bereich wird dadurch aber auf die Regel begrenzt.

Regelmuster

Ein Regelmuster ist die Darstellung einer Ereigniskorrelationssituation (wie z. B. eine Schwellenwertbedingung oder eine Erkennung von duplizierten Ereignissen). Die ACT-Regelsprache definiert die folgenden Regelmuster: Datenerfassung, Berechnung, Duplikat, Filter, Sequenz, Schwellenwert und Zeitgeber.

Das Muster einer Regel stimmt überein, wenn die durch die Regel definierte Situation eintritt. Wenn das Muster übereinstimmt, schließt die Regel ihre Verarbeitung ab, indem Sie die entsprechenden Regelantwortaktionen ausführt. Während eine Regel aktiv ist, kann das Regelmuster mehrfach abgeglichen werden.

Durch ein Filtermuster definierte Regeln sind die einzigen statusunabhängigen Regeln in der Regelsprache. Alle anderen Regeln sind statusabhängig.

Erfassungsmuster

Eine Datenerfassungsregel wird durch das Erfassungsmuster definiert. Sie erfasst eine Gruppe von ausgewählten Ereignissen innerhalb eines Zeitintervalls. Sie ist eine statusabhängige Regel.

Übersicht

Das Erfassungsmuster wird verwendet, um ähnliche Ereignisse über einen Zeitraum zu erfassen. Der Zeitraum wird durch ein verbindliches Zeitfenster angegeben, das im Element <timeWindow> der Regelsprache definiert ist.

Bedingungen, unter denen die Regelantwort ausgeführt wird

Beim Erfassungsmuster wird die Regelantwort ausgeführt, wenn das Zeitfenster beendet ist, wie durch das Element <onTimeWindowComplete> definiert.

Verwendungsbeispiel dieses Regelmusters

Ein Verwendungsbeispiel für das Erfassungsmuster ist eine Regel, die Folgendes ausführt:

> Sie erfasst Ereignisse, die die Bedingungen eines bestimmten Ereignisselektors während des Zeitraums erfüllen. Wenn der Zeitraum endet, fasst sie die erfassten Ereignisse in einem einzigen Ereignis zusammen, das die Gesamtzahl der Ereignisse und die charakteristischen Informationen zu den zusammengefassten Ereignissen enthält.

Zugehörige Verweise

„Datenerfassungsregel - Zusammenfassung” auf Seite 49
In dieser Zusammenfassung werden alle Sprachelemente der Datenerfassungsregel aufgeführt.

Berechnungsmuster

Eine Berechnungsregel wird durch das Berechnungsmuster definiert. Es wendet eine Berechnung (durch einen Ausdruck) auf erfasste Ereignisse an, während die einzelnen Ereignisse innerhalb eines Zeitintervalls empfangen werden. Es ist eine statusabhängige Regel.

Übersicht

Das Berechnungsmuster führt eine durch das Element <computeFunction> der Regelsprache definiert Berechnungsfunktion für jedes Ereignis aus, das während eines Zeitraums akzeptiert wird. Der Zeitraum wird durch ein verbindliches Zeitfenster angegeben, das durch das Element <timeWindow> definiert wird.

Bedingungen, unter denen die Regelantwort ausgeführt wird

Wenn das Zeitfenster beendet ist, wird beim Berechnungsmuster die Regelantwort ausgeführt, die im Element <onTimeWindowComplete> definiert ist. Der Wert der Berechnung ist während der Aktion <onTimeWindowComplete> verfügbar.

Verwendungsbeispiel dieses Regelmusters

Angenommen, eine Anwendung verarbeitet Kundenauftragsereignisse. Ein Verwendungsbeispiel für das Berechnungsmuster ist eine Regel, die Folgendes ausführt:

> Jedes Mal, wenn ein Ereignis empfangen wird, wird der Gesamtwert des Auftrags zum Gesamtwert aller Aufträge addiert, die während des angegebenen Zeitraums erfolgt sind, und der aktualisierte Gesamtwert aller Aufträge wird in einer Benutzeroberfläche veröffentlicht.

Zugehörige Verweise

„Berechnungsregel - Zusammenfassung” auf Seite 50
In dieser Zusammenfassung werden alle Sprachelemente der Berechnungsregel aufgeführt.

Duplikatmuster

Eine Duplikatregel wird durch das Duplikatmuster definiert. Es zählt das zweite Ereignis sowie nachfolgende Ereignisse, die innerhalb eines angegebenen Zeitintervalls akzeptiert werden, überspringt jedoch die Regelsatzverarbeitung für diese Ereignisse. Es ist eine statusabhängige Regel.

Übersicht

Das Duplikatmuster wird normalerweise verwendet, um ähnliche (duplizierte) Ereignisse über einen Zeitraum einzugrenzen. Ein dupliziertes Ereignis ähnelt in gewisser Hinsicht einem vorherigen Ereignis, ohne notwendigerweise eine exakte Kopie dieses Ereignisses zu sein. Ereignisse zählen als Duplikate, wenn sie dieselben Ereignisauswahlkriterien für die Regel erfüllen. Der Zeitraum wird durch ein verbindliches Zeitfenster angegeben, das im Element <timeWindow> der Regelsprache definiert ist.

Bedingungen, unter denen die Regelantwort ausgeführt wird

Bei einem Duplikatmuster wird die Regelantwort in den folgenden Fällen ausgeführt:

- Wenn das erste Ereignis erkannt wird, das durch das Element <onDetection> definiert ist.
- Während jedes duplizierte Ereignis verarbeitet wird, das durch das Element <onNextEvent> definiert ist.
- Wenn das Zeitfenster beendet wird, das durch das Element <onTimeWindowComplete> definiert ist.

Das erste Ereignis löst die <onDetection>-Aktion aus, auch wenn keine duplizierten Ereignisse empfangen wurden. Der Grund für dieses Verhalten ist, dass Sie möglicherweise das erste Ereignis weiterleiten möchten und die Regelsatzverarbeitung für duplizierte Ereignisse überspringen möchten. In diesem Fall können Sie eine Regelantwortaktion hinzufügen, die das erste Ereignis weiterleitet, wenn die <onDetection>-Aktion für die Regel ausgelöst wird.

Die Standardverarbeitung für duplizierte Ereignisse (das zweite Ereignis und nachfolgende Ereignisse) umfasst das Zählen eines duplizierten Ereignisses und das Überspringen der Regelsatzverarbeitung für ein dupliziertes Ereignis. Wenn Sie zusätzliche Aktionen für ein dupliziertes Ereignis ausführen möchten, können Sie explizit eine <onNextEvent>-Aktion definieren. Beispiel: In bestimmten Fällen stellt das duplizierte Ereignis ein Ereignis dar, das eventuell bereits in einer Datenbank oder in einem anderen Repository protokolliert ist. Daher möchten Sie vielleicht eine <onNextEvent>-Aktion codieren, um das duplizierte Ereignis an diesen anderen Positionen zu entfernen.

Eine <onTimeWindowComplete>-Aktion kann verwendet werden, um einen Summensatz für alle duplizierten Ereignisse zu erstellen, der die Anzahl verarbeiteter Duplikate umfasst.

Verwendungsbeispiel dieses Regelmusters

Angenommen, für denselben Ressourcentyp (ein Sicherheitsmonitor) tritt immer wieder die Nachricht „Denial of Service" auf. Dies ist ein Hinweis auf eine mögliche Sicherheitsverletzung. Ein Verwendungsbeispiel für das Duplikatmuster ist eine Regel, die Folgendes ausführt:

> Nachdem die Nachricht „Denial of Service" bei einem Sicherheitsmonitor aufgetreten ist, werden alle Duplikate dieses Ereignisses gezählt, die während eines Zeitraums von 30 Sekunden auftreten, aber nicht an die Operatorkonsole gesendet. Zudem generiert die Regel nach Ablauf der 30 Sekunden ein Ereignis, das die Anzahl „Denial of Service"-Nachrichten angibt, die in diesem Zeitraum aufgetreten sind.

Zugehörige Verweise

„Duplikatregel - Zusammenfassung" auf Seite 51
In dieser Zusammenfassung werden alle Sprachelemente der Duplikatregel aufgeführt.

Filtermuster

Eine Filterregel wird durch das Filtermuster definiert. Sie führt eine bestimmte Aktion aus, wenn sie ein Ereignis akzeptiert. Sie wird nur für ein einzelnes Ereignis ausgeführt und ist somit eine statusunabhängige Regel.

Übersicht

Das Filtermuster wird verwendet, um einzelne Ereignissen zu prüfen, die die Ereignisauswahlkriterien erfüllen. Im Gegensatz zu den anderen Regelmustern behält es keine zugehörigen Statusinformationen bei (wie z. B. das Protokoll zu vergangenen Ereignissen).

Bedingungen, unter denen die Regelantwort ausgeführt wird

Beim Filtermuster wird die Regelantwort ausgeführt, wenn ein beliebiges Ereignis empfangen wird, das die Ereignisauswahlkriterien erfüllt, die durch das Element <onDetection> definiert sind.

Verwendungsbeispiel dieses Regelmusters

Ein Verwendungsbeispiel für das Filtermuster ist eine Regel, die Folgendes ausführt:

> Wenn das Ereignis `ServerStatus` eine Serverauslastung (`serverLoad`) angibt, die größer als 95% ist, führt die Regel eine Aktion aus, durch die ein Administrator gerufen wird.

Zugehörige Verweise

„Filterregel - Zusammenfassung" auf Seite 53
In dieser Zusammenfassung werden alle Sprachelemente der Filterregel aufgeführt.

Sequenzmuster

Eine Sequenzregel wird durch das Sequenzmuster definiert. Es ermittelt, ob eine gewisse Sequenz von Ereignissen in einem Zeitintervall ankommt. Die Reihenfolge kann sortiert oder zufällig sein. Eine Sequenzregel ist eine statusabhängige Regel.

Übersicht

Das Sequenzmuster überprüft eine Sequenz von Ereignissen über einen Zeitraum und ermittelt, ob die Sequenz vollständig oder unvollständig ist. Eine unvollständige Sequenz schließt mindestens ein Ereignis in der angegebenen Sequenz ein, aber nicht alle.

Der Zeitraum wird durch ein verbindliches Zeitfenster angegeben, das im Element <timeWindow> der Regelsprache definiert ist. Jedes Ereignis der Sequenz wird durch ein eigenes Element <eventSelector> in der Regel definiert. Die Sequenz von Ereignissen kann wie folgt ermittelt werden:

- In der Reihenfolge, in der die Elemente <eventSelector> für die Regel codiert sind. Wenn die Regel das Ereignis erkennt, das durch das erste Element <eventSelector> definiert ist, wird die Erkennung der Sequenz gestartet. Die Regel erwartet nun das Ereignis, das durch das zweite Element <eventSelector> definiert ist.
- In Zufallsreihenfolge. Wenn die Regel beliebige Ereignisse erkennt, die durch die Elemente <eventSelector> definiert sind, wird die Erkennung der Sequenz gestartet. Die Regel erwartet nun ein weiteres Ereignis, das durch die Elemente <eventSelector> definiert ist.

Das Sequenzmuster unterscheidet sich vor allem wie folgt von den anderen Regelmustern:

- Es verfügt über mehrere Elemente <eventSelector> zum Definieren, welche Ereignisse von der Regel akzeptiert werden. Es sind mindestens zwei Elemente <eventSelector> erforderlich.
- Wenn ein Ereignis die Kriterien erfüllt, die durch eines der Elemente <eventSelector> definiert wurden, wird dieses Element <eventSelector> in dieser Regelinstanz von der weiteren Ereignisverarbeitung ausgeschlossen.
- Das Aliasnamensattribut im Element <eventSelector> ist nur in einer Sequenzregel gültig. Es gibt einem Ereignis, das durch einen bestimmten Ereignisselektor in der Sequenzregel ausgewählt wird, einen eindeutigen Namen. In einem Ausdruck in einem Filterprädikat oder in einer Filteraktion können Sie die Variable 'act_eventList' verwenden, um mit einem Aliasnamen auf ein Ereignis in einer Sequenzregel zuzugreifen.

Bedingungen, unter denen die Regelantwort ausgeführt wird

Mit dem Sequenzmuster wird die Regelantwort wie folgt ausgeführt:

- Wenn die vollständige Sequenz der Ereignisse im Zeitfenster erkannt wird, das durch das Element <onDetection> definiert ist.
- Wenn mindestens ein Ereignis, aber nicht die vollständige Sequenz im Zeitfenster empfangen wird, das durch das Element <onTimeOut> definiert ist.

  Das Sequenzmuster kann hilfreich sein, um zu erkennen, ob eine angegebene Sequenz unvollständig ist. Tritt z. B. ein Ereignis „system down" ohne nachfolgendes Ereignis „system up" auf, kann der Autor der Regel eine Aktion <onTimeOut> codieren, um diese Art eines fehlenden Ereignisses abzuwickeln.

Verwendungsbeispiel dieses Regelmusters

Szenario zur Erkennung einer vollständigen Sequenz:
Nehmen Sie an, dass ein Administrator in einer IT-Umgebung wissen möchte, ob sich der Wert der Größe des DB2-Heapspeichers auf WebSphere Application Server auswirkt, und falls zutreffend, dass er dieses Problem beheben möchte. Wenn die folgenden Ereignisse in der aufgeführten Reihenfolge in einem bestimmten Zeitraum auftreten, wird der Administrator also den Wert der Größe des DB2-Heapspeichers erhöhen und den Datenbankmanager erneut starten:

1. Eine Ausnahmebedingung bei der WebSphere Application Server-Ressourcenzuordnung. Nehmen Sie an, dass es sich um ein Ereignis des Typs `WASResourceAllocationException` handelt.
2. Eine DB2-Fehlernachricht „Nicht genügend Speicher im Zwischenspeicher für Anwendungen (Application Heap) für die Verarbeitung der Anweisung". Nehmen Sie an, dass es sich um ein Ereignis des Typs `DB2NotEnoughHeap` handelt.

In diesem Szenario sind zwei Elemente <eventSelector> in einer Sequenzregel definiert, und die Ereignisse müssen (anstatt in zufälliger Reihenfolge) in der Reihenfolge empfangen werden, in der die Elemente <eventSelector> codiert sind. Das erste Element <eventSelector> überprüft das Vorhandensein von Ereignis `WASResourceAllocationException`, und das zweite Element <eventSelector> überprüft das Vorhandensein von Ereignis `DB2NotEnoughHeap`. Setzen Sie voraus, dass die folgenden Ereignisse im angegebenen Zeitfenster an das System übergeben werden:

1. `WASResourceAllocationException`
2. `DB2BackupStarted`
3. `WASResourceAllocationException`

4. `WASResourceAllocationException`
5. `DB2NotEnoughHeap`

Die Regel verhält sich wie folgt:

1. Das erste Ereignis, `WASResourceAllocationException`, wird akzeptiert. Weil die Bedingungen für das erste Element <eventSelector> erfüllt wurden, wird das erste Element <eventSelector> nun von der weiteren Ereignisverarbeitung in dieser Regel ausgeschlossen.
2. Das zweite Ereignis, `DB2BackupStarted`, wird ignoriert.
3. Das dritte Ereignis, `WASResourceAllocationException`, wird ignoriert.
4. Das vierte Ereignis, `WASResourceAllocationException`, wird ignoriert.
5. Das fünfte Ereignis, `DB2NotEnoughHeap`, wird akzeptiert, und es beendet die Sequenz. Die Regelantwortaktion <onDetection> wird ausgeführt. Diese Aktion ist definiert, um den Wert der Größe des DB2-Heapspeichers zu erhöhen und den Datenbankmanager erneut zu starten. Die Regel kehrt in ihren Anfangsstatus zurück.

   Das erste Element <eventSelector> wird nun in die weitere Ereignisverarbeitung durch diese Regel eingeschlossen.

Szenario zur Erkennung einer unvollständigen Sequenz:
Nehmen Sie an, dass ein Unternehmen alle Kundenaufträge innerhalb einer Stunde nach Auftragserteilung zur Lieferung bereitgestellt haben will und erfahren möchte, wenn dies nicht eintritt.

In diesem Szenario sind zwei Elemente <eventSelector> in einer Sequenzregel definiert, und die Ereignisse müssen (anstatt in zufälliger Reihenfolge) in der Reihenfolge empfangen werden, in der die Elemente <eventSelector> codiert sind. Das erste Element <eventSelector> überprüft das Vorhandensein von Ereignis `Netsales` mit `operationType=Order`, und das zweite Element <eventSelector> überprüft das Vorhandensein von Ereignis `Netsales` mit `operationType=Delivery`. Setzen Sie voraus, dass die folgenden Ereignisse im angegebenen Zeitfenster von einer Stunde an das System übergeben werden:

1. Ein Ereignis `Netsales` mit `operationType=Order`
2. Ein Ereignis `Netsales` mit `operationType=Order`

Die Regel verhält sich wie folgt:

1. Das erste Ereignis wird akzeptiert. Weil die Bedingungen für das erste Element <eventSelector> erfüllt wurden, wird das erste Element <eventSelector> nun von der weiteren Ereignisverarbeitung in dieser Regel ausgeschlossen.
2. Das zweite Ereignis wird ignoriert.
3. Da innerhalb des angegebenen Zeitfensters kein Ereignis `Netsales` mit `operationType=Delivery` empfangen wurde, wird die Regelantwortaktion <onTimeOut> ausgeführt. Diese Aktion ist definiert, um einen Geschäftsführer darüber zu informieren, dass ein Kundenauftrag nicht innerhalb einer Stunde nach Auftragserteilung zur Lieferung bereit steht. Die Regel kehrt in ihren Anfangsstatus zurück.

   Das erste Element <eventSelector> wird nun in die weitere Ereignisverarbeitung durch diese Regel eingeschlossen.

   Zugehörige Konzepte

   „Zugriff auf ereignisbezogene Informationen" auf Seite 23
   In den folgenden Beispielen wird dargestellt, wie Sie mit Hilfe der Variablen, die von Active Correlation Technology bereitgestellt werden, auf ereignisbezogene Informationen zugreifen können.

Zugehörige Verweise

„Sequenzregel - Zusammenfassung" auf Seite 54
In dieser Zusammenfassung werden alle Sprachelemente der Sequenzregel aufgeführt.

Schwellenwertmuster

Eine Schwellenwertregel wird durch das Schwellenwertmuster definiert. Es erfasst eine Gruppe ausgewählter Ereignisse in einem Zeitintervall und ermittelt, nachdem alle Ereignisse empfangen wurden, ob eine Schwellenwertbedingung zutrifft. Es ist eine statusabhängige Regel.

Übersicht

Das Schwellenwertmuster erfasst Ereignisse über einen Zeitraum, bis ein Schwellenwert zutrifft. Der Zeitraum wird durch ein verbindliches Zeitfenster angegeben, das im Element <timeWindow> der Regelsprache definiert ist.

Das Schwellenwertmuster stellt die folgenden drei Optionen für den Typ des Schwellenwerts zur Verfügung:

Schwellenwert für Ereigniszähler

Bei diesem Schwellenwerttyp können Sie die Anzahl Ereignisse definieren, die den Auswahlkriterien für Ereignisse in einem gewissen Zeitraum entsprechen müssen. Der definierte Schwellenwert wird mit der Anzahl akzeptierter Ereignisse verglichen. Wenn der Ereigniszähler dem definierten Grenzwert im Zeitfenster entspricht, ist der Schwellenwert erreicht.

Dieser Typ des Schwellenwerts kann zur Überprüfung eines sehr einfachen Ereigniszählers nützlich sein. Z. B. könnte er eine Antwort auf die folgende Frage liefern: „Traten fünf Anmeldefehler in einer Minute auf?"

Dieser Schwellenwert wird durch das Element <eventCountThreshold> definiert. Das Element <eventCountThreshold> gibt auch einen der folgenden beiden möglichen Zeitintervallmodi für das Zeitfenster an:

Festgelegtes Intervall

Ein festgelegtes Intervall beginnt, wenn das erste Ereignis empfangen wird, das die Ereignisauswahlkriterien erfüllt, und endet, wenn eine der folgenden Aussagen eintritt:

- Der Schwellenwert der Regel wird im angegebenen Zeitraum erreicht.
- Die angegebene Dauer ist abgelaufen.

Schiebeintervall

Ein Schiebeintervall beginnt, wenn das erste Ereignis empfangen wird, das die Ereignisauswahlkriterien erfüllt. Wenn die Regel ihren Schwellenwert jedoch nicht erreicht hat und die angegebene Dauer abgelaufen ist, passt das Zeitfenster die Anfangszeit an die Ereignisempfangszeit für ein neues „erstes" Ereignis an, das normalerweise das nächste Ereignis ist, das akzeptiert wird. Die Anfangszeit wird somit geschoben. Das Schiebeintervall fährt mit der Anpassung auf diese Weise fort, bis eine der folgenden Situationen eintritt:

- Der Schwellenwert der Regel wird im angegebenen Zeitraum erreicht.

- Nachdem das Ereignis empfangen wird, das das Zeitfenster startet, werden innerhalb der angegebenen Dauer keine folgenden Ereignisse empfangen.

Das Ereignis, das das Zeitfenster startet (das neue „erste" Ereignis), ist das Ereignis mit einer Empfangszeit, die die folgende Bedingung erfüllt: Die Empfangszeit, in Addition mit der Zeitintervalldauer für die Regel, ist größer als die aktuelle Uhrzeit. Die Bedingung kann auch in Form einer Gleichung ausgedrückt werden:

```
ereignisempfangszeit + zeitintervalldauer für regel > aktuelle uhrzeit
```

Wenn ein solches Ereignis nicht vorhanden ist, kann das Schiebeintervall die Zeit nicht weiter nach vorne schieben, und das Intervall endet.

Berechneter Schwellenwert

Bei diesem Schwellenwerttyp können Sie Code schreiben (oder den Code eines anderen Autors verwenden), der eine Berechnung für jedes akzeptierte Ereignis ausführt und einen berechneten Schwellenwert zurückgibt, der in einer zuvor definierten Variablen gespeichert wird. Dieser berechnete Schwellenwert wird anschließend mit einem definierten Schwellenwert verglichen, um festzustellen, ob der Schwellenwert erreicht wurde.

Deshalb kann eine komplexe Berechnung zum Erstellen (oder Aktualisieren) eines berechneten Schwellenwerts angewendet werden. Möglicherweise werden Daten verwendet, die von vorherigen Ereignissen gespeichert wurden, und der Autor der Regel kann den definierten Schwellenwert unabhängig von der Logik festlegen, die den berechneten Schwellenwert ermittelt.

Dieser Schwellenwerttyp kann für die Zusammenfassung und den Vergleich eines Wertes mit einem definierten Schwellenwert nützlich sein. Sie kann z. B. verwendet werden, um die Summe (in Euro) der Verkäufe an einen bestimmten Kunden über eine gewissen Zeitraum zu berechnen und sie mit einem definierten Schwellenwert zu vergleichen.

Dieser Schwellenwert wird durch das Element <computedThreshold> definiert.

Boolescher Schwellenwert

Bei diesem Schwellenwerttyp können Sie Code schreiben (oder den Code eines anderen Autors verwenden), der den Wert `true` oder `false` für jedes akzeptierte Ereignis übergibt. Wenn der Wert `true` ist, wurde der Schwellenwert erreicht. Wenn der Wert `false` ist, wird die Schwellenwertregel weiter verarbeitet, bis der Zeitraum überschritten ist oder ein anderes Ereignis akzeptiert wurde.

Dieser Schwellenwerttyp kann zur Überprüfung von Wertebereichen nützlich sein. Wenn die CPU-Auslastung z. B. immer zwischen 30% und 80% liegen muss, kann dieser Schwellenwert ständig prüfen, ob die Auslastung in diesem Bereich liegt.

Dieser Schwellenwert wird durch das Element <booleanThreshold> definiert.

Bedingungen, unter denen die Regelantwort ausgeführt wird

Mit dem Schwellenwertmuster wird die Regelantwort wie folgt ausgeführt:

- Wenn der Schwellenwert erreicht wurde, der durch das Element <onDetection> definiert ist.
- Wenn mindestens ein Ereignis akzeptiert wurde, der Schwellenwert im Zeitfenster, das im Element <onTimeOut> definiert ist, aber nicht erreicht wurde.

Verwendungsbeispiel dieses Regelmusters

Ein Verwendungsbeispiel des Schwellenwertmusters mit einem Schwellenwert für Ereigniszähler ist eine Regel, die folgende Aktionen ausführt:

> Wenn mehr als vier Ereignisse `Server unreachable` in einem Schiebeintervall von 30 Sekunden vom selben Teilnetz ausgehen, führt die Regel eine Aktion aus, um den Status eines Routers zu überprüfen.

Zugehörige Verweise

„Schwellenwertregel - Zusammenfassung" auf Seite 55
In dieser Zusammenfassung werden alle Sprachelemente der Schwellenwertregel aufgeführt.

Zeitgebermuster

Eine Zeitgeberregel wird durch das Zeitgebermuster definiert. Sie leitet Aktionen in regelmäßigen Intervallen ein. Es ist eine statusabhängige Regel. Obwohl eine Zeitgeberregel keine Ereignisse verarbeitet, kann sie durch ein Ereignis aktiviert oder inaktiviert werden.

Übersicht

Ein Zeitgebermuster entspricht einem Zeitgeber, der am Anfang eines Zeitraums startet und am Ende des Zeitraums stoppt. Der Zeitraum wird durch ein verbindliches Zeitfenster angegeben, das im Element <timeWindow> der Regelsprache definiert ist.

Das Zeitgebermuster wird wiederholt, bis die Zeitgeberregel inaktivert wird. Soll das Zeitgebermuster nicht wiederholt werden, muss dies ausdrücklich angegeben werden. Deshalb wartet die Zeitgeberregel nach dem Starten den angegebenen Zeitraum ab, bevor sie Aktionen einleitet. Sie wird so lange wiederholt, bis sie inaktiviert wird oder die ACT-Engine heruntergefahren wird.

Die Zeitgeberregel ist einzigartig, da sie keine Auswahlkriterien für Ereignisse enthält. Die Zeitgeberregel beginnt die Verarbeitung entsprechend dem Aktivierungsintervall für die Regel, das durch das Element <activationInterval> definiert wird. Wenn das Standardelement <activationInterval> verwendet wird und das Zeitgebermuster zum Wiederholen festgelegt ist, wird die Zeitgeberregel gestartet, wenn sie von der ACT-Engine geladen wird, und gestoppt, wenn die ACT-Engine heruntergefahren wird. Zum Aktivieren einer Zeitgeberregel durch ein Ereignis müssen Sie das Ereignis im Element <activateOnEvent> innerhalb des Elements <activationInterval> für die Regel definieren.

Bedingungen, unter denen die Regelantwort ausgeführt wird

Mit dem Zeitgebermuster wird die Regelantwort ausgeführt, wenn das Zeitfenster beendet ist, das im Element <onTimeWindowComplete> definiert ist.

Verwendungsbeispiel dieses Regelmusters

Das Zeitgebermuster kann zum Implementieren von Bereinigungsregeln nützlich sein. Ein Verwendungsbeispiel des Zeitgebermusters ist eine Regel, die Folgendes ausführt:

> Alle 30 Minuten führt die Regel eine Aktion aus, die harmlose Ereignisse und Informationsereignisse löscht, die länger als 48 Stunden geöffnet waren.

Zugehörige Verweise

„Zeitgeberregel - Zusammenfassung" auf Seite 57
In dieser Zusammenfassung werden alle Sprachelemente der Zeitgeberregel aufgeführt.

Allgemeine und eindeutige Aspekte verschiedener Regelmuster

Diese Matrix stellt eine problemorientierte Übersicht über die allgemeinen und eindeutigen Aspekte der verschiedenen Regelmuster bereit.

Tabelle 2 listet die primären Sprachelemente für die Regeln auf und zeigt ein X in der Spalte für jeden Regeltyp an, für den das Element gültig ist. Die primären Sprachelemente sind die direkten untergeordneten Elemente der verschiedenen Regeltypen. In der Liste werden keine Elemente aufgeführt, die in diesen direkten untergeordneten Elementen enthalten sind, die zudem vom Regeltyp abhängig sein können. Darüber hinaus kann die Gültigkeit bestimmter Elementattribute je nach Regeltyp variieren.

Tabelle 2. Matrix zur Anzeige der allgemeinen und eindeutigen Aspekte verschiedener Regelmuster

Element	Erfassung	Berechnung	Duplikat	Filter	Sequenz	Schwellenwert	Zeitgeber
<comment>	X	X	X	X	X	X	X
<variable>	X	X	X	X	X	X	X
<activationInterval>	X	X	X	X	X	X	X
<lifeCycleActions>	X	X	X	X	X	X	X
<eventSelector>	X	X	X	X	X	X	
<groupingKey>	X	X	X		X	X	
<timeWindow>	X	X	X		X	X	X
<computeFunction>		X					
<booleanThreshold>						X	
<computedThreshold>						X	
<eventCountThreshold>						X	
<onDetection>			X	X	X	X	
<onNextEvent>			X				
<onTimeOut>					X	X	
<onTimeWindowComplete>	X	X	X				X

Ausdrücke

Ein Ausdruck ist Code, der eine angepasste Logik enthält, die einer Regel hinzugefügt werden kann. Ausdrücke können auch auf Code außerhalb der ACT-Engine zugreifen. In der Regelsprache sind Ausdrücke nur in bestimmten Kontexten, oder Regelsprachelementen, gültig.

Autoren von Regeln können Ausdrücke für verschiedene Zwecke codieren, abhängig vom Kontext und vom Ergebnis, das sie erzielen möchten. Ausdrücke

werden häufig für die Initialisierung von Variablen, die Definition von Ereignisauswahlkriterien und für die Spezifikation von Regelantwort- und Lebenszyklusaktionen verwendet.

Sprachelemente, die Ausdrücke enthalten

Jedes Sprachelement, das einen Ausdruck enthält, weist ein expressionLanguage-Attribut auf, das die Programmiersprache angibt, in der der Ausdruck geschrieben wurde. Die Programmiersprache Java ist die einzige unterstützte Ausdruckssprache.

Ausdrücke können in den folgenden Regelsprachelementen enthalten sein.

- <varInitializer> für eine Regelsatz-, Regelblock- oder Regelvariable
- <filteringPredicate> in <eventSelector>
- <computedValue> in <groupingKey>
- <computeFunction> in einer Berechnungsregel
- <booleanThreshold> in einer Schwellenwertregel
- <computedThreshold> in einer Schwellenwertregel
- Regelantwortaktionen für eine Regel:
  - <action> innerhalb <onDetection>. Diese Aktion ist nur für die Duplikat-, Filter-, Sequenz- und Schwellenwertregel gültig.
  - <action> innerhalb <onNextEvent>. Diese Aktion ist nur für die Duplikatregel gültig.
  - <action> innerhalb <onTimeOut>. Diese Aktion ist nur für die Sequenz- und Schwellenwertregel gültig.
  - <action> innerhalb <onTimeWindowComplete>. Diese Aktion ist nur für die Datenerfassungs-, Berechnungs-, Duplikat- und Zeitgeberregel gültig.
- Lebenszyklusaktionen für eine Regel:
  - <action> in <onLoad>
  - <action> in <onActivation>
  - <action> in <onDeactivation>
  - <action> in <onUnload>

Unterstützung durch Active Correlation Technology bei der Codierung von Ausdrücken

Zur Unterstützung des Autors einer Regel bei der Codierung von Ausdrücken stellt Active Correlation Technology folgende Funktionalitäten bereit:

- Import von externen Modulen (wie z. B. Java-Klassen) und Objekten für die Verwendung in Ausdrücken.
- Initialisierung von und Zugriff auf Regelsatz-, Regelblock- und Regelvariablen.
- Durch die Variable 'act_event' Zugriff auf das aktuelle Ereignis, das eine Regel gerade verarbeitet.
- Durch die Variable 'act_eventCount' Zugriff auf die Anzahl Ereignisse, die von einer Regel akzeptiert wurden.
- Durch die Variable 'act_eventList' Zugriff auf die Liste der Ereignisse, die von einer Regel akzeptiert wurden. Dazu gehören der Zugriff auf verschiedene Attribute eines Ereignisses sowie der Zugriff auf die einzelnen Ereignisse in einer Sequenzregel nach Aliasname.

- Durch die Variable 'act_lib' Zugriff auf Methoden, die die Funktion umfassen, Variablen abzurufen und festzulegen und den Ereignisfluss durch einen Regelsatz zu steuern.
- Durch die Variable 'act_location' Zugriff auf die Position eines Ausdrucks innerhalb der Regelhierarchie.
- Durch die Variable 'act_nodeName' Zugriff auf den vollständig qualifizierten Namen eines Knotens.
- Durch die Variable 'act_threshold' Zugriff auf den definierten Schwellenwert für eine Schwellenwertregel.

Externe Module und Objekte importieren und darauf zugreifen

In diesem Beispiel wird gezeigt, wie Sie externen Code (wie z. B. Java-Klassen) sowie externe Objekte für Ausdrücke zugänglich machen können. Ein externes Objekt ist ein Objekt, das von einer Anwendung erstellt wird, um mit Ausdrücken zu kommunizieren.

Bevor Sie auf externen Code von einem Ausdruck zugreifen können, müssen Sie diesen Code für Ausdrücke zugänglich machen.

Ein Import ist eine programmiersprachenspezifische Möglichkeit, externen Code für Ausdrücke zugänglich zu machen. Das Element <import> enthält einen speziellen Ausdruckstyp, der die externen Module (wie z. B. Java-Klassen) angibt, die für die Verwendung in anderen Ausdrücken in Regeln importiert werden sollen. Ein Import kann auf der Ebene des Regelsatzes oder eines Regelblocks definiert werden.

Das folgende Element <import> enthält einen Ausdruck, der in der Programmiersprache Java geschrieben wurde und der die Klassen 'StaticHelper' und 'Queue' importiert, auf die von anderen Ausdrücken verwiesen werden kann:

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

Obwohl die Verwendung des vollständigen Klassennamens in der Importanweisung nicht erforderlich ist, sollten Sie den vollständigen Namen angeben, um eine lange Kompilierzeit zu vermeiden. Zum Beispiel sollte die Java-Klasse als `com.ibm.act.sample.StaticHelper` an Stelle von `com.ibm.act.sample.*` oder `com.ibm.act.*` angegeben werden.

Auf eine statische Methode zugreifen

Das folgende Beispiel zeigt, wie ein Ausdruck in einer Regelantwortaktion auf die Klasse 'StaticHelper' verweist, nachdem die Klasse importiert wurde:

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

Auf eine Instanzdefinitionsmethode für ein Objekt zugreifen

Das folgende Beispiel zeigt, wie ein Ausdruck in einer Regelantwortaktion auf die Klasse 'Queue' verweist, nachdem die Klasse importiert wurde. In diesem Beispiel wird ein externes Objekt mit dem Namen `OutputQueueOne` und dem Typ `Queue` abgerufen und dazu verwendet, ein Ereignis in eine bestimmte Warteschlange zu stellen.

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

Variablen initialisieren und darauf zugreifen

In diesem Beispiel wird gezeigt, wie Sie Regelsatz-, Regelblock- oder Regelvariablen initialisieren und auf diese zugreifen können.

Eine Variable kann auf der Ebene des Regelsatzes, eines Regelblocks oder einer Regel definiert werden. Bevor auf eine Variable zugegriffen werden kann, muss sie mit einem Initialisierungsausdruck initialisiert werden. Der folgende Ausdruck initialisiert zwei Variablen mit den Namen 'hostsList' und 'hostsString':

```
<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>
```

Auf alle Variablen wird durch Ausdrücke zugegriffen. Das folgende Beispiel zeigt, wie durch einen Ausdruck in einer Regelantwortaktion auf die Variablen 'hostsList' und 'hostsString' zugegriffen wird, die im vorherigen Beispiel initialisiert wurden. In diesem Beispiel wird 'hostsList' geändert, und 'hostsString' wird ein neuer Wert gegeben.

```
<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>
```

Zugriff auf ereignisbezogene Informationen

In den folgenden Beispielen wird dargestellt, wie Sie mit Hilfe der Variablen, die von Active Correlation Technology bereitgestellt werden, auf ereignisbezogene Informationen zugreifen können.

Beispiel für den Zugriff auf das aktuelle Ereignis:
Der folgende Code zeigt, wie die Variable 'act_event' verwendet wird, um das Hostnamensattribut für ein Ereignis abzurufen:

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

Beispiel für den Zugriff auf Ereignisse mit Hilfe der Ereignisliste nach Index:
Der folgende Code zeigt, wie die Variable 'act_eventList' verwendet wird, um das erste Ereignis in der Ereignisliste abzurufen:

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

Beispiel für den Zugriff auf Ereignisse mit Hilfe der Ereignisliste nach Aliasnamen:
Im Gegensatz zu anderen Regeltypen lässt die Sequenzregel mehrere Ereignisselektoren zu. Tatsächlich sind sogar mindestens zwei Ereignisselektoren erforderlich. Das Aliasnamensattribut im Element <eventSelector> ist nur in einer Sequenzregel gültig. Es gibt einem Ereignis, das durch einen bestimmten Ereignisselektor in der Sequenzregel ausgewählt wird, einen eindeutigen Namen. In einem Ausdruck in einem Filterprädikat oder in einer Filteraktion können Sie die Variable 'act_eventList' verwenden, um mit einem Aliasnamen auf ein Ereignis in einer Sequenzregel zuzugreifen.

Der folgende Code zeigt zwei Ereignisselektoren für eine Sequenzregel. Die Aliasnamen lauten TECevent und WASevent.

```
<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>
```

Der folgende Code zeigt, wie die Variable 'act_eventList' verwendet wird, um das Ereignis abzurufen, das vom ersten Ereignisselektor (mit dem Namen TECevent) akzeptiert wurde:

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

Bewährte Verfahren zur Codierung von Ausdrücken

Diese Informationen enthalten einige bewährte Verfahren, Hinweise und Tipps zur effizienten und effektiven Codierung von Ausdrücken.

- Zum leichteren Verständnis enthalten die meisten Ausdrucksbeispiele, die in diesen Informationen bereitgestellt werden, Java-Code direkt in den XML-Anweisungen. Bei der Erstellung von Regeln wird jedoch empfohlen, externe Module für Java-Code zu verwenden und diese externen Module als Bestandteil der Ausdrücke aufzurufen.

  Sie können im Regelerstellungsprogramm auch vorhandene Snippets verwenden oder bearbeiten oder neue Snippets erstellen, um Code für den Aufruf von externen Modulen bereitzustellen. Snippets sind Auszüge aus dem Quellcode, die in Ausdrücken verwendet werden können. Im Regelerstellungsprogramm sind diese Snippets über die Snippetanzeige verfügbar.

  Mit Hilfe dieses Konzepts können der Entwurf, die Entwicklung, die Bearbeitung, das Testen und das Debugging von Java-Code in der integrierten Entwicklungsumgebung Ihrer Wahl erfolgen und als fester Bestandteil Ihres Entwicklungsprozesses gesteuert werden.
- Der Ausdruckscode sollte in einen CDATA-Abschnitt eingeschlossen werden, damit er nicht als XML-Code ausgewertet wird. Hierbei steht wie im folgenden Beispiel <![CDATA[direkt vor dem Code und]]> direkt nach dem Code:

  ```
  <onTimeOut>
   <action expressionLanguage="java">
  <![CDATA[
    IEvent firstEvent = act_eventList.get(0);
  ```

```
  System.out.println("Abgelaufenes Element: " + firstEvent.getAttribute("sourceComponentId.location"));
]]>
</action>
</onTimeOut>
```

XML-Parser ignorieren alle Angaben in einem CDATA-Abschnitt.

- Wenn Sie die Methode `act_lib.getExternalContext()` in einem Ausdruck verwenden, sollten Sie das Objekt, das von der Methode zurückgegeben wird, nicht in einer Regelsatz- oder Regelblockvariablen speichern. Dies liegt daran, dass eine Anwendung den Verweis auf die Objektinstanz ändern kann und die zugeordnete Regelsatz- oder Regelblockvariable nicht aktualisiert wird.
- Wenn Sie eine Rückkehranweisung (`return;`) in einem Ausdruck im Element <action> verwenden und zusätzliche Elemente <action> für die jeweilige Regelantwort oder Lebenszyklusaktion codiert sind, wird die Ausführung des Codes bei der Anweisung `return;` beendet und im Ausdruck im nächsten Element <action> erneut gestartet.
- Das Regelmanagement und andere Methoden der ACT-Engine können nicht aus einer Regelantwort oder Lebenszyklusaktion heraus aufgerufen werden.

Zugehörige Informationen

„Externe Module und Objekte importieren und darauf zugreifen" auf Seite 22
In diesem Beispiel wird gezeigt, wie Sie externen Code (wie z. B. Java-Klassen) sowie externe Objekte für Ausdrücke zugänglich machen können. Ein externes Objekt ist ein Objekt, das von einer Anwendung erstellt wird, um mit Ausdrücken zu kommunizieren.

„Snippets in Ausdrücke innerhalb von Regeln einschließen" auf Seite 43
Sie können Snippets über die Anzeige **Snippet** in der Eclipse-Workbench in Ausdrücke innerhalb von Regeln einschließen.

Variablen

In der Regelsprache werden bestimmte Variablen zum Speichern ereignisbezogener Informationen in unterschiedliche Ereignisvorkommen oder Regeln verwendet. Ausdrücke in Regeln können anschließend auf diese ereignisbezogenen Informationen zugreifen. Manche Variablentypen werden vom Autor der Regel definiert, andere werden von Active Correlation Technology bereitgestellt. Auf einige Typen kann direkt in einem Ausdruck zugegriffen werden, auf andere kann nur mit den von Active Correlation Technology bereitgestellten Methoden zugegriffen werden.

Variablen, die in <variable>-Elementen definiert sind und auf die über Methoden zugegriffen wird

Sie können eine Variable im Element <variable> für eine Regel, einen Regelblock oder einen Regelsatz definieren. Sie können anschließend in einem Ausdruck mit einer der folgenden Methoden auf diese Variable zugreifen:

- Methode „getVariable()" oder eine der Methoden „get*javatyp*Variable()"
- Methode „setVariable()" oder eine der Methoden „set*javatyp*Variable()"

Wenn Sie z. B. die Variable *regelautorvariable* im Element <variable> für eine Regel definieren, können Sie mit dem folgenden Code auf diese Variable zugreifen:

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

Variablen, die von Active Correlation Technology bereitgestellt werden und auf die direkt in einem Ausdruck zugegriffen wird

Die folgenden Variablen werden von Active Correlation Technology bereitgestellt. Sie können diese Variablen in einem Ausdruck integriert verwenden.

- act_event
- act_eventList
- act_lib

Mit dem folgenden Code können Sie z. B. auf die Variable „act_event" zugreifen, um das Attribut „hostname" eines Ereignisses abzurufen:

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

Variablen, die von Active Correlation Technology bereitgestellt werden und auf die über Methoden zugegriffen wird

Die folgenden Variablen werden von Active Correlation Technology bereitgestellt. Sie können in einem Ausdruck mit der Methode „getVariable()" oder einer der Methoden „get*javatyp*Variable()" auf diese Variablen zugreifen.

- act_eventCount
- act_location
- act_nodeName
- act_threshold

Mit dem folgenden Code können Sie z. B. auf die Variable „act_eventCount" zugreifen:

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

Tabelle 3 zeigt die Konstanten, die die IACTLibrary-Schnittstelle für diese Variablen bereitstellt. Damit sichergestellt ist, dass eventuelle Schreibfehler während der Kompilierung und nicht während der Ausführung gefunden werden, sollten Sie in Ihrem Code an Stelle der Variablen immer die Konstanten verwenden, die diese Variablen darstellen. Verwenden Sie z. B. `act_lib.getIntVariable(IACTLibrary.EVENTCOUNT);` an Stelle von `act_lib.getIntVariable("act_eventCount");`.

Tabelle 3. Variablen mit zugeordneten Konstanten

Variable	Zugeordnete Konstante
act_eventCount	EVENTCOUNT
act_location	LOCATION
act_nodeName	NODENAME
act_threshold	THRESHOLD

Zugehörige Verweise

„Element 'variable'" auf Seite 117
Das Element <variable> definiert eine Variable. Es enthält Informationen in einem Format, auf das Ausdrücke verweisen können. Eine Variable kann auf der Ebene des Regelsatzes, eines Regelblocks oder einer Regel definiert werden.

Datentypen für Active Correlation Technology-Variablen

Die Variablen, die von Active Correlation Technology bereitgestellt werden, haben unterschiedliche Datentypen.

Tabelle 4 zeigt die Datentypen für diese Variablen.

Tabelle 4. Datentypen für Active Correlation Technology-Variablen

Variable	Datentyp
act_ event	Der Typ, der von der Schnittstelle 'com.ibm.correlation.IEvent' definiert wird.
act_ eventCount	int
act_ eventList	Der Typ, der von der Schnittstelle 'com.ibm.correlation.IEventList' definiert wird.
act_ lib	Der Typ, der von der Schnittstelle 'com.ibm.correlation.IACTLibrary' definiert wird.
act_ location	java.lang.String
act_ nodeName	java.lang.String
act_ threshold	Da diese Variable der Wert des Schwellenwertattributs im Element <computedThreshold> oder <eventCountThreshold> einer Schwellenwertregel ist, muss der Datentyp mit dem Datentyp für den Wert des Schwellenwertattributs identisch sein.

Ausdruckkontexte, in denen Variablen gültig sind

Die Variablen, die von Active Correlation Technology bereitgestellt werden, sind nur in bestimmten Ausdruckkontexten gültig.

Tabelle 5 listet die Ausdruckkontexte auf, in denen diese Variablen gültig sind. Die Tabelle enthält ein X in der Spalte für jede Variable, die im jeweiligen Ausdruckskontext gültig ist. Weitere Nutzungsbeschränkungen, die auf diese Variablen angewendet werden, werden im Abschnitt für die jeweilige Variable beschrieben.

Tabelle 5. Ausdruckkontexte, in denen Variablen gültig sind

Ausdruckkontext	act_ event	act_ eventCount	act_ eventList	act_ lib	act_ location	act_ nodeName	act_ threshold
<action> in <onActivation>	X			X	X	X	X
<action> in <onDeactivation>	X	X	X	X	X	X	X
<action> in <onDetection>	X	X	X	X	X	X	X
<action> in <onLoad>				X	X	X	X
<action> in <onNextEvent>	X	X	X	X	X	X	
<action> in <onTimeOut>		X	X	X	X	X	X
<action> in <onTimeWindowComplete>		X	X	X	X	X	
<action> in <onUnload>				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
<varInitializer> für eine Regel				X	X	X	X

Variable 'act_event'

Die Variable 'act_event' ermöglicht den Zugriff auf Methoden, die für das aktuelle Ereignis gelten.

Details

Da eine Zeitgeberregel keine Ereignisse verarbeitet, gilt die Variable 'act_event' in einer Zeitgeberregel nur für die Ereignisse, die die Regel aktivieren oder inaktivieren.

Die Variable 'act_event' gilt in den Aktionen <onActivation> und <onDeactivation> nur dann, wenn die Regel von einem Ereignis aktiviert oder inaktiviert wurde. Andernfalls ist diese Variable null.

Codebeispiel

Der folgende Code greift auf die Variable 'act_event' zu, um das Hostnamensattribut eines Ereignisses abzurufen:

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

Methoden, auf die zugegriffen werden kann

Die Methoden, für die die Variable 'act_event' einen Zugriff ermöglicht, werden in der IEvent-Schnittstelle definiert, wie in Tabelle 6 dargestellt.

Tabelle 6. IEvent-Schnittstelle mit entsprechenden Methoden und Speicherposition von Javadoc-Methodenbeschreibungen

Schnittstelle	Methoden	Speicherposition der Javadoc-Methodenbeschreibung
IEvent	• get • getAttribute • getBooleanAttribute • getByteAttribute • getShortAttribute • getIntAttribute • getLongAttribute • getFloatAttribute • getDoubleAttribute • getStringAttribute • set • getTimeStamp • setTimeStamp • getType • getOriginal	com.ibm.correlation.IEvent

Variable 'act_eventCount'

Die Variable 'act_eventCount' ist eine Ganzzahl gleich der Anzahl Ereignisse, die von einer Regel akzeptiert wurden.

Details

Bei einer Duplikatregel ist der Wert der Variablen 'act_eventCount' die Gesamtzahl der akzeptierten Ereignisse, wozu das ursprüngliche Ereignis und alle Duplikate zählen. Bei allen anderen Regeltypen entspricht der Wert der Größe der Ereignisliste, die durch die Variable 'act_eventList' mit der Methode 'act_eventList.size()' abgerufen werden kann.

Die Variablen 'act_eventCount' und 'act_eventList' sind in einer Zeitgeberregel nicht gültig, da eine Zeitgeberregel keine Ereignisse verarbeitet.

Wenn eine Regel in einem Gruppierungsschlüssel definiert wird, sind die Variablen 'act_eventCount', 'act_eventList' und 'act_threshold' in den folgenden Ausdruckskontexten nicht gültig:

- Lebenszyklusaktionen
- <filteringPredicate> in <activateOnEvent> oder <deactivateOnEvent> in <activationInterval>
- <computedValue>

Dies ist der Fall, weil die Regelvariablen in diesem Fall nur auf eine Regelinstanz angewendet werden und Regelinstanzen bei der Ausführung dieser Ausdrücke nicht vorhanden sind.

Codebeispiel

Der folgende Code greift auf die Variable 'act_lib' zu, um die Anzahl Ereignisse abzurufen, die von einer Regel akzeptiert wurden:

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

Variable 'act_eventList'

Die Variable 'act_eventList' ermöglicht den Zugriff auf Methoden, die für die Liste der Ereignisse gelten, die von einer Regel akzeptiert wurden.

Details

Eine Filterregel und eine Duplikatregel haben grundsätzlich eine Liste mit nur einem Ereignis, da es sich bei einer Filterregel um eine statusunabhängige Regel handelt und da eine Duplikatregel nur das erste analysierte Ereignis behält.

Die Variablen 'act_eventCount' und 'act_eventList' sind in einer Zeitgeberregel nicht gültig, da eine Zeitgeberregel keine Ereignisse verarbeitet.

Wenn eine Regel in einem Gruppierungsschlüssel definiert wird, sind die Variablen 'act_eventCount', 'act_eventList' und 'act_threshold' in den folgenden Ausdruckskontexten nicht gültig:

- Lebenszyklusaktionen
- <filteringPredicate> in <activateOnEvent> oder <deactivateOnEvent> in <activationInterval>
- <computedValue>

Dies ist der Fall, weil die Regelvariablen in diesem Fall nur auf eine Regelinstanz angewendet werden und Regelinstanzen bei der Ausführung dieser Ausdrücke nicht vorhanden sind.

Codebeispiel

Der folgende Code greift auf die Variable 'act_eventList' zu, um das zweite Ereignis in der Ereignisliste abzurufen:

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

Methoden, auf die zugegriffen werden kann

Die Methoden, für die die Variable 'act_eventList' den Zugriff ermöglicht, werden in der IEventList-Schnittstelle definiert, wie in Tabelle 7 dargestellt.

Tabelle 7. IEventList-Schnittstelle mit entsprechenden Methoden und Speicherposition von Javadoc-Methodenbeschreibungen

Schnittstelle	Methoden	Speicherposition der Javadoc-Methodenbeschreibung
IEventList	• get • size • isEmpty • listIterator	com.ibm.correlation.IEventList

Variable 'act_lib'

Die Variable „act_lib" stellt den Zugang zu Bibliotheksmethoden in Active Correlation Technology zur Verfügung.

Details

Die Methoden, auf die die Variable „act_lib" zugreifen kann, hängen vom Element der Regelsprache ab, das den Ausdruck enthält, in dem die Variable verwendet wird. Siehe Tabelle 8.

Tabelle 8. Methoden, auf die die Variable "act_lib" zugreifen kann, basierend auf dem Kontext des übergeordneten Ausdrucks

Ausdruckkontext	Methoden für IACTLibrary	Methoden für IExitableActionLibrary	Methoden für IActionLibrary
<action> in <onActivation>	X		
<action> in <onDeactivation>	X		
<action> in <onDetection>	X	X	X
<action> in <onLoad>	X		
<action> in <onNextEvent>	X	X	X
<action> in <onTimeOut>	X		X
<action> in <onTimeWindowComplete>	X		X
<action> in <onUnload>	X		
<booleanThreshold>	X		
<computedThreshold>	X		
<computedValue>	X		
<computeFunction>	X		
<filteringPredicate>	X		
<varInitializer>	X		

Codebeispiel

Der folgende Code greift auf die Variable 'act_lib' zu, um die Methode aufzurufen, mit der der Regelsatz beendet wird:

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

Methoden, auf die zugegriffen werden kann

Die Methoden, auf die die Variable „act_lib variable" Zugriff bietet, sind in den Schnittstellen „IACTLibrary", „IExitableActionLibrary" und „IActionLibrary" definiert (siehe Tabelle 9).

Tabelle 9. Schnittstellen mit entsprechenden Methoden und Speicherposition der Javadoc-Methodenbeschreibungen

Schnittstelle	Methoden	Speicherposition der Javadoc-Methodenbeschreibung
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 Die in der Schnittstelle „IACTLibrary" definierten Methoden sind auch in der Schnittstelle „IActionLibrary" verfügbar.	com.ibm.correlation.IActionLibrary
IExitableActionLibrary	• exitRuleSet • exitRuleBlock Die in den Schnittstellen „IACTLibrary" und „IActionLibrary" definierten Methoden sind auch in der Schnittstelle „IExitableActionLibrary" verfügbar.	com.ibm.correlation.IExitableActionLibrary

Variable 'act_location'

Die Variable „act_location" ist eine Zeichenfolge, die die Position eines Ausdrucks innerhalb der Regelhierarchie definiert.

Details

Die Position ist ein vollständig qualifizierter Name, der die Position des Ausdrucks in der Regelhierarchie angibt. Er hat das Format *id.id....*, wobei jedes Auftreten von *id* für eine der folgenden Angaben steht:

- Der Wert des Namensattributs für ein XML-Element in der jeweiligen Hierarchie.
- Bei Elementen, die mehrmals in einem Regelblock oder einer Regel auftreten und kein Namensattribut haben: das XML-Element, das den Ausdruck enthält, mit nachfolgender Indexnummer in eckigen Klammern. Diese Indexnummer zeigt die Position des Ausdrucks im übergeordneten Element an. Der Zähler zum Zuordnen der Indexnummern beginnt mit 0 und nicht mit 1. Wenn ein Element z. B. im dritten Element <action> enthalten ist, erhält es daher die Indexnummer `action[2]`.

Diese Kennungen werden in absteigender Reihenfolge vom Regelblock der höchsten Ebene zum Element der niedrigsten Ebene, das den Ausdruck enthält, aufgeführt.

Codebeispiel

Der folgende Code greift auf die Variable 'act_lib' zu, um die Position des Ausdrucks abzurufen:

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

Beispiele für die Position, die von der Variablen übergeben wird

Die folgenden Werte sind Beispiele für die Position, die von der Variablen „act_location" übergeben werden.

`ruleBlockA.ruleA.eventSelector[3].filteringPredicate`
Dieser Ausdruck ist in den folgenden Elementen enthalten:
- Der Regelblock mit dem Namensattributwert `ruleBlockA`
- Die Regel mit dem Namensattributwert `ruleA`
- Das vierte Element <eventSelector>
- Das Element <filteringPredicate>

`ruleBlockA.ruleA.onDetection.action[5]`
Dieser Ausdruck ist in den folgenden Elementen enthalten:
- Der Regelblock mit dem Namensattributwert `ruleBlockA`
- Die Regel mit dem Namensattributwert `ruleA`
- Das Element <onDetection>
- Das sechste Element <action>

`ruleBlockA.ruleA.variableA.varInitializer`
Dieser Ausdruck ist in den folgenden Elementen enthalten:
- Der Regelblock mit dem Namensattributwert `ruleBlockA`
- Die Regel mit dem Namensattributwert `ruleA`
- Die Variable mit dem Namensattributwert `variableA`
- Das Element <varInitializer>

Variable 'act_nodeName'

Die Variable „act_nodeName" ist eine Zeichenfolge, die den vollständig qualifizierten Namen eines Knotens angibt.

Details

In Active Correlation Technology ist ein Knoten ein Objekt in einer Regelhierarchie, das individuell und unabhängig in einem Regelsatz hinzugefügt, gelöscht oder ersetzt werden kann. Insbesondere die folgenden Objekte sind Knoten: Regeln, Regelblöcke, Variablen für Regelblöcke und Variablen für Regelsätze. Weil für ein Objekt keine individuelle und unabhängige Verarbeitung unterhalb der Regelebene ausgeführt werden kann, ist eine Regelvariable kein Knoten.

Der vollständig qualifizierte Knotenname für eine Regel mit dem Namensattributwert `rule1`, die in einem Regelblock mit dem Namensattributwert `ruleBlockA` enthalten ist, ist `ruleBlockA.rule1`. Da die Regeln in einem Regelsatz hierarchisch organisiert sind, wird ein Punkt (.) in einem vollständig qualifizierten Knotenname verwendet, um auf einen Knoten einer niedrigeren Ebene hinzuweisen.

Codebeispiel

Der folgende Code greift auf die Variable 'act_nodeName' zu, um den vollständig qualifizierten Namen eines Knotens abzurufen:

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

Variable 'act_threshold'

Die Variable 'act_threshold' enthält den Wert des Schwellenwertattributs, d. h. den definierten Schwellenwert, im Element <computedThreshold> oder <eventCountThreshold> einer Schwellenwertregel.

Details

Die Variable „act_threshold" ist nur in einer Schwellenwertregel gültig.

Wenn eine Regel in einem Gruppierungsschlüssel definiert wird, sind die Variablen 'act_eventCount', 'act_eventList' und 'act_threshold' in den folgenden Ausdruckskontexten nicht gültig:

- Lebenszyklusaktionen
- <filteringPredicate> in <activateOnEvent> oder <deactivateOnEvent> in <activationInterval>
- <computedValue>

Dies ist der Fall, weil die Regelvariablen in diesem Fall nur auf eine Regelinstanz angewendet werden und Regelinstanzen bei der Ausführung dieser Ausdrücke nicht vorhanden sind.

Codebeispiel

Der folgende Code greift auf die Variable 'act_lib' zu, um einen definierten Schwellenwert abzurufen:

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

Ereignisfluss durch einen Regelsatz

Ereignisse fließen in der Reihenfolge durch einen Regelsatz, in der die Regelblöcke und Regeln codiert sind. Wenn die ACT-Engine ein Ereignis empfängt, stellt die Engine den Ereignistyp fest und ermittelt die Regeln, die diesen Ereignistyp für die Regelaktivierung, die Ereignisverarbeitung oder die Regelinaktivierung verwenden.

Verwendung von Ereignissen durch Regeln

Jede Regel, die das Ereignis verwendet, stellt zunächst fest, ob das Ereignis alle angegebenen Bedingungen für die Regelaktivierung, die Ereignisverarbeitung oder die Regelinaktivierung erfüllt. Wenn dies der Fall ist, führt die Regel die folgenden Aktionen aus:

Für die Regelaktivierung
Die Aktionen im Element <onActivation> für die Regel werden ausgeführt, falls sie codiert sind.

Für die Ereignisverarbeitung
Die Regel verarbeitet das Ereignis. Wenn das Regelmuster übereinstimmt, werden die Regelantwortaktionen ausgeführt, falls diese codiert sind. In einigen Situationen können Regelantwortaktionen Folgendes ausführen:

- Die Aktion kann veranlassen, dass die Verarbeitung des Ereignisses im Rest des Regelblocks oder Regelsatzes übersprungen wird.
- Die Aktion kann ein neues oder vorhandenes Ereignis zur Verarbeitung an eine andere Regel oder an einen anderen Regelblock senden.

Für die Regelinaktivierung
Die Aktionen im Element <onDeactivation> für die Regel werden ausgeführt, falls sie codiert sind.

Methoden, die den Fluss von Ereignissen beeinflussen können

Active Correlation Technology stellt die folgenden Methoden bereit, die aufgerufen werden können, um den Fluss von Ereignissen durch den Regelsatz zu beeinflussen. Diese Methoden sind durch die Variable 'act_lib' verfügbar.

exitRuleSet
Diese Methode gibt an, dass das aktuelle Ereignis nicht durch weitere Regeln im Regelsatz verarbeitet wird.

exitRuleBlock
Diese Methode gibt an, dass das aktuelle Ereignis nicht durch weitere Regeln im aktuellen Regelblock oder in anderen Regelblöcken verarbeitet wird, die dieser Regelblock enthält. Es wird jedoch von weiteren Regeln verarbeitet, die sich außerhalb des aktuellen Regelblocks befinden.

forward
Diese Methode gibt an, dass ein Ereignis an andere Regeln und Regelblöcke gesendet werden soll, auch wenn die aktuelle Regel ihre Verarbeitung noch nicht beendet hat. Die anderen Regeln und Regelblöcke verarbeiten das Ereignis dann vollständig, bevor sie das Ereignis an die Regel zurückgeben, die die Weiterleitungsmethode aufgerufen hat.

forwardOnCompletion
Diese Methode gibt an, dass ein Ereignis an andere Regeln und Regelblöcke gesendet werden soll, nachdem die aktuelle Regel ihre Verarbeitung beendet hat.

Kapitel 3. Übersicht über das Schreiben von Regeln

Vor dem Schreiben von Regeln zum Korrelieren von Ereignissen muss die Ereigniskorrelation verstanden und geplant werden. Anschließend können die Regeln entworfen werden. Sie können das ACT-Regelerstellungsprogramm zum Schreiben der Regeln und Kompilieren des Regelsatzes verwenden.

Das ACT-Regelerstellungsprogramm ist eine GUI zum Schreiben von Regeln. Es enthält eine Onlinehilfe. Im Regelerstellungsprogramm ist die resultierende Regelsatzdatei eine XML-Datei mit dem Dateityp „actl".

Active Correlation Technology stellt kein Änderungsmanagement oder System zur Versionssteuerung bereit.

Planung einer Ereigniskorrelation

Zur Planung einer Ereigniskorrelation gehört ein grundlegendes Verständnis darüber, was eine Ereigniskorrelation ist und wie sie in ihrer Anwendung angewendet werden kann.

Vergewissern Sie sich, dass Sie die folgenden Konzepte verstehen:

- Die in Kapitel 1, „Einführung", auf Seite 3 und Kapitel 2, „Übersicht über die Regelsprache", auf Seite 5 enthaltenen Informationen
- Die Ereignisse, die Ihre Anwendung verarbeitet

Jede Anwendung kann eine unterschiedliche Gruppe von Ereignissen verarbeiten, wie in den folgenden Beispielen beschrieben:

Beispiel für das Versicherungswesen
Im Versicherungswesen können Ereignisse, die den Arbeitsablauf zur Bearbeitung von Leistungsansprüchen aufzeichnen, erzeugt und korreliert werden, um festzustellen, ob die Geschäftsprozesse zeitnah beendet werden.

Beispiel für den Vertrieb
Im Vertrieb können Umsatzergebnisse regelmäßig zusammengefasst, aufgelistet und mit einem Ziel verglichen werden, um den Status der zu erreichenden Verkaufsziele in einem bestimmten Zeitraum anzuzeigen.

Beispiel für eine IT-Umgebung
In einer IT-Umgebung kann ein unternehmenskritisches System jede Minute ein Ereignis erzeugen, um anzuzeigen, dass ein Datenbankserver normal ausgeführt wird. Korrelationsregeln können geschrieben werden, um den Erhalt dieser Überwachungsereignisse zu überwachen und bestimmte Regelantwortaktionen auszuführen, wenn ein erwartetes Überwachungsereignis nicht empfangen wird.

Außerdem sollten Sie das Format der Ereignisse verstehen, die ihre Anwendung verarbeitet. Active Correlation Technology stellt Java-Klassen und -Methoden zur Verfügung, um auf Daten in Ereignissen zuzugreifen, die durch die ACT-Engine verarbeitet werden. Wenn Sie diese Klassen und Methoden verwenden wollen, um bei der Verarbeitung auf Ereignisse zuzugreifen oder diese zu ändern, sollten Sie in jedem Fall ein grundlegendes Verständnis für die zugrundeliegenden Ereignisobjekte haben.

Gehen Sie zur Planung einer Ereigniskorrelation wie folgt vor:

1. Ermitteln Sie die Ereignisse Ihrer Anwendung, die korreliert werden sollen.
2. Ermitteln Sie die Regelmuster zum Korrelieren der Ereignisse.

   Ein Regelmuster stellt eine bestimmte Situation einer Ereigniskorrelation dar. Es kann zum Korrelieren der Ereignisse verwendet werden, die etwas zu dieser Situation beitragen. Denken Sie daran, wie sich die Ereignisse, die von Ihrer Anwendung verarbeitet werden, auf die Regelmuster beziehen, die durch die ACT-Regelsprache definiert werden. Dies ist möglicherweise hilfreich, um festzustellen, welche Regelmuster verwendet werden müssen.

   Verwenden Sie immer das Muster, das für Ihre Situation einer Ereigniskorrelation am besten geeignet ist. Wenn Sie z. B. möchten, dass eine Regel eine bestimmte Folge von Ereignissen erkennt, sollten Sie keinen Code schreiben, um das Sequenzmusterverhalten in die Regelantwortaktionen für eine Filterregel einzuschließen. Verwenden Sie stattdessen das Sequenzmuster, um eine Sequenzregel zu erstellen.
3. Ermitteln Sie die Anweisungen jedes Regelmusters, das Sie verwenden wollen.

   In den folgenden Informationen werden die primären Anweisungen der Regelsprache zusammengefasst, wobei die Details jeder Anweisung für das Regelmuster eindeutig sind. Diese Informationen sind ähnlich organisiert, wie sie in der GUI des Regelerstellungsprogramms dargestellt werden:

   Merkmale
   Definition der Regelmerkmale einschließlich Regelname, Beschreibung und Muster. Details finden Sie in den folgenden Themen:
   - „Element 'collectionRule'" auf Seite 75
   - „Element 'computationRule'" auf Seite 77
   - „Element 'duplicateRule'" auf Seite 85
   - „Element 'filterRule'" auf Seite 93
   - „Element 'sequenceRule'" auf Seite 107
   - „Element 'thresholdRule'" auf Seite 112
   - „Element 'timerRule'" auf Seite 115

   Variablen
   Definition der Regelvariablen einschließlich Name, Typ, Beschreibung und Initialisierungsausdruck für jede Variable. Details finden Sie in „Element 'variable'" auf Seite 117.

   Ereignisauswahl
   Definition der Kriterien, die festlegen, welche Ereignisse von der Regel für die Verarbeitung akzeptiert werden. Details finden Sie in „Element 'eventSelector'" auf Seite 90.

   Gruppierungsschlüssel
   Definition des Gruppierungsschlüssels, über den die Regel angewiesen wird, eine getrennte Regelinstanz (oder eine Kopie) für jede Gruppe von Ereignissen zu erstellen, die allgemeine Merkmale gemeinsam nutzen. Details finden Sie in „Element 'groupingKey'" auf Seite 94.

   Musterspezifikationen
   Spezifikation des Zeitraums, während dessen die statusabhängige Regel verarbeitet wird, um ihr Muster und die Definition von eindeutigen Aspekten bestimmter statusabhängiger Regelmuster abzugleichen. Details finden Sie in „Element 'timeWindow'" auf Seite 116.

Bei der Berechnungsregel ist die Definition der Berechnung mit eingeschlossen, die auf die erfassten Ereignisse angewendet wird. Details finden Sie in „Element 'computeFunction'" auf Seite 81.

Bei der Schwellenwertregel sind die Definition des Schwellenwerttyps und weitere für den Typ spezifische Informationen mit eingeschlossen. Details finden Sie in den folgenden Themen:

- „Element 'booleanThreshold'" auf Seite 75
- „Element 'computedThreshold'" auf Seite 78
- „Element 'eventCountThreshold'" auf Seite 87

Regelantworten
Definition der Aktionen, die ausgeführt werden sollen, wenn die Regel ihre Verarbeitung beendet.

Details finden Sie in den folgenden Themen:

- Duplikat-, Filter-, Sequenz- oder Schwellenwertregeln: „Element 'onDetection'" auf Seite 100
- Duplikatregeln: „Element 'onNextEvent'" auf Seite 101
- Sequenz- oder Schwellenwertregeln: „Element 'onTimeOut'" auf Seite 102
- Datenerfassungs-, Berechnungs-, Duplikat- oder Zeitgeberregeln: „Element 'onTimeWindowComplete'" auf Seite 103

Aktivierungsintervall
Definition, wann eine Regel aktiv bzw. inaktiv ist. Details finden Sie in „Element 'activationInterval'" auf Seite 69.

Lebenszyklus
Definition der Aktionen, falls vorhanden, die in den folgenden vier primären Phasen im Lebenszyklus einer Regel stattfinden sollen: Laden, Aktivierung, Inaktivierung und Entladen. Normalerweise müssen diese Aktionen nicht definiert werden. Details finden Sie in „Element 'lifeCycleActions'" auf Seite 97.

4. Geben Sie Java-Methoden und zugeordnete Snippets an, die innerhalb von Regelausdrücken aufgerufen werden sollen. Anstatt umfassenden Java-Code innerhalb von Regelausdrücken zu schreiben, sollten Regelautoren Java-Methoden für den Aufruf externer Module verwenden. Diese externen Module können von der Anwendung bereitgestellt werden, in die Active Correlation Technology eingebettet ist, oder vom Regelautor nach Bedarf erstellt werden. Die Snippets, die den einzelnen Java-Methoden zugeordnet sind, sollten ebenfalls angegeben werden. Weitere Informationen finden Sie in „Bewährte Verfahren zur Codierung von Ausdrücken" auf Seite 24.

Fahren Sie mit „Regeln für die Korrelation von Ereignissen entwerfen" fort.

Regeln für die Korrelation von Ereignissen entwerfen

Wenn Sie eine Ereigniskorrelation geplant haben, müssen Sie die Regeln entwerfen, mit denen Ereignisse korreliert werden sollen.

Führen Sie zunächst „Planung einer Ereigniskorrelation" auf Seite 35 aus.

Wenn Sie die Regeln entwerfen möchten, führen Sie die folgenden Schritte aus:

1. Entwerfen Sie die Organisation der Regeln und Regelblöcke innerhalb des Regelsatzes.

2. Entwerfen Sie die Ebene, auf der verschiedene Variablen definiert werden sollen, beispielsweise auf Regelsatz-, Regelblock- oder Regelebene.
3. Entwerfen Sie die Elemente der einzelnen Regeln entsprechend dem Regelmuster.

   In diesem Schritt werden die Anweisungen der einzelnen Regelmuster verwendet, die Sie im Planungsstadium angegeben haben.
4. Entwerfen Sie die Interaktionen zwischen Regeln, insbesondere was das Weiterleiten von Ereignissen und das Überspringen von Regelsatzverarbeitungen für duplizierte Ereignisse betrifft.

   Weitere Details finden Sie in „Ereignisfluss durch einen Regelsatz" auf Seite 33.
5. Entwerfen, entwickeln und testen Sie alle Java-Methoden und zugeordneten Snippets, die Sie erstellt haben, damit sie in Ausdrücken aufgerufen werden.

Fahren Sie mit „Erste Schritte mit dem Regelerstellungsprogramm" fort.

Erste Schritte mit dem Regelerstellungsprogramm

Nachdem Sie die Regeln für die Korrelation von Ereignissen entworfen haben, können Sie mit Hilfe des ACT-Regelerstellungsprogramms die Regeln schreiben.

Die folgenden Schritte zählen zu den Hauptschritten beim Schreiben von Regeln mit dem Regelerstellungsprogramm.

1. Öffnen Sie die Eclipse-Workbench.
2. Legen Sie Ihre Anzeige in der Eclipse-Workbench fest.
3. Legen Sie Ihre Einstellungen für Active Correlation Technology fest, oder akzeptieren Sie die Standardeinstellungen.
4. Erstellen Sie ein Projekt, in dem die Regelsatzdatei gespeichert werden soll, oder verwenden Sie ein vorhandenes Projekt.
5. Erstellen Sie eine Regelsatzdatei, und speichern Sie sie in einem Projekt Ihrer Wahl.
6. Erstellen Sie mindestens einen Regelblock innerhalb des Regelsatzes. Darüber hinaus können Sie weitere Regelblöcke innerhalb des Regelsatzes erstellen, und Sie können Regelblöcke innerhalb von Regelblöcken erstellen.
7. Erstellen Sie Regeln innerhalb von Regelblöcken.
8. Prüfen Sie den Regelsatz.
9. Kompilieren Sie den Regelsatz.
10. Aktualisieren Sie den Regelsatz nach Bedarf.

In Ausdrücke innerhalb von Regeln können Sie auch Snippets aus der Anzeige **Snippet** in der Eclipse-Workbench einschließen.

Anzeige in der Eclipse-Workbench festlegen

Bevor Sie beginnen, sollten Sie Ihre Anzeige in der Eclipse-Workbench festlegen. In diesem Abschnitt wird beschrieben, wie die Anzeige des Regelerstellungsprogramms geöffnet wird.

Öffnen Sie zuerst die Eclipse-Workbench, wenn sie nicht bereits geöffnet ist.

Sie können zwar auch eine andere Anzeige als die Anzeige des Regelerstellungsprogramms auswählen, die Schritte für die Ausführung verschiedener Tasks variieren jedoch ein wenig abhängig von der gewählten Anzeige.

Wenn Sie die Anzeige des Regelerstellungsprogramms öffnen möchten, führen Sie die folgenden Schritte aus:

1. Klicken Sie in der Workbench auf **Window → Open Perspective → Other...**.
2. Klicken Sie auf **Rule Builder** und anschließend auf **OK**. Daraufhin wird die Anzeige des Regelerstellungsprogramms angezeigt.

Fahren Sie mit „Einstellungen festlegen" fort.

Einstellungen festlegen

Vor der Erstellung einer Regelsatzdatei sollten Sie sicherstellen, dass Ihre Einstellungen für Active Correlation Technology in der Eclipse-Workbench korrekt festgelegt sind. In diesem Abschnitt wird beschrieben, wie diese Einstellungen festgelegt werden.

Öffnen Sie zuerst die Eclipse-Workbench, wenn sie nicht bereits geöffnet ist.

Wenn Sie Ihre Einstellungen für Active Correlation Technology festlegen möchten, führen Sie die folgenden Schritte aus:

1. Klicken Sie in der Workbench auf **Fenster → Benutzervorgaben...**.
2. Klicken Sie auf **Active Correlation Technology**, und geben Sie auf der Seite für Active Correlation Technology an, ob Sie „act" als Präfix für neu erstellte Regeln und Regelblöcke im Regelerstellungsprogramm hinzufügen möchten. Standardmäßig wird „act" *nicht* als Präfix hinzugefügt.
3. Erweitern Sie **Active Correlation Technology**. Abhängig von Ihrer Anwendung können die folgenden zusätzlichen Elemente angezeigt werden. Klicken Sie auf diese Elemente, um die zugehörigen Einstellungen festzulegen.

Element	Zugehörige Einstellung
Common Base Event-Definitionsprovider	Sie können die XML-Dateien angeben, die die Struktur für mindestens einen Ereignistyp bereitstellen (einschließlich der Namen und Typen von Attributen, die ein vorgegebener Ereignistyp enthalten kann), der der Common Base Event-Spezifikation entspricht. Diese Ereignistypen und Attribute sind dann bei der Erstellung von Regeln verfügbar.

Element	Zugehörige Einstellung
Compiler	Sie können die folgenden primären Elemente angeben: • Ob kompilierte Regelsätze gespeichert werden sollen • Den Dateityp für kompilierte Regelsätze • Den Klassenpfad für Code, der zur Kompilierzeit verwendet werden soll Standardmäßig werden kompilierte Regelsätze gespeichert und in der Navigatoranzeige mit dem Dateityp `acts` angezeigt, der angibt, dass es sich bei der Compilerausgabe um einen serialisierten Regelsatz handelt.

Fahren Sie mit „Projekt für die Speicherung einer Regelsatzdatei erstellen" fort.

Projekt für die Speicherung einer Regelsatzdatei erstellen

Bei der Erstellung einer Regelsatzdatei müssen Sie in der Eclipse-Workbench das Projekt angeben, in dem die Datei gespeichert werden soll. Daher müssen Sie vor der Erstellung einer Regelsatzdatei ein Projekt erstellen oder ein vorhandenes Projekt verwenden. In diesem Abschnitt wird eine Möglichkeit beschrieben, ein Projekt in der Eclipse-Workbench zu erstellen.

Öffnen Sie zuerst die Eclipse-Workbench, wenn sie nicht bereits geöffnet ist. Öffnen Sie außerdem die Anzeige des Regelerstellungsprogramms.

Wenn Sie ein einfaches Projekt in der Workbench erstellen möchten, führen Sie die folgenden Schritte aus:

1. Klicken Sie auf **File** → **New** → **Project...**.
2. Klicken Sie im Assistenten für die Erstellung eines neuen Projekts auf **Simple** → **Project** und anschließend auf **Next**.
3. Geben Sie einen eindeutigen Namen für das Projekt in das Feld **Project name** ein. Verwenden Sie dann die Standardspeicherposition für das Projekt, oder wählen Sie eine andere Speicherposition aus.
4. Klicken Sie auf **Finish**. Das neue Projekt wird dann in der Navigatoranzeige der Anzeige des Regelerstellungsprogramms angezeigt.

Fahren Sie mit „Regelsatz erstellen" fort.

Regelsatz erstellen

In diesem Abschnitt wird beschrieben, wie ein Regelsatz erstellt wird.

Öffnen Sie zuerst die Eclipse-Workbench, wenn sie nicht bereits geöffnet ist. Öffnen Sie außerdem die Anzeige des Regelerstellungsprogramms.

Wenn Sie eine Regelsatzdatei erstellen möchten, führen Sie die folgenden Schritte aus:

1. Klicken Sie auf **File** → **New** → **Rule Set File**.
2. Klicken Sie im Assistenten für die Erstellung der neuen Regelsatzdatei auf den Namen des Ordners, der dem Projekt zugeordnet ist, in dem Sie die Regelsatzdatei speichern möchten. Dies ist vermutlich das Projekt, das Sie in

„Projekt für die Speicherung einer Regelsatzdatei erstellen" auf Seite 40 erstellt haben. Der Ordnername wird anschließend im ersten Feld angezeigt.

3. Geben Sie in das Feld **File name** einen Namen für die Regelsatzdatei ein. Der Dateityp muss 'actl' lauten.
4. Klicken Sie auf **Next**.
5. Geben Sie einen eindeutigen Namen und eine optionale Beschreibung für den Regelsatz ein. Wenn Sie für das Feld **XML encoding** nicht den Standardwert verwenden möchten, geben Sie auch die Codierungsdarstellung für die Regelsatzdatei an, die eine XML-Datei ist.
6. Klicken Sie auf **Finish**. Die Regelsatzdatei wird dann in der Navigatoranzeige in ihrem Projektordner angezeigt. Da die Datei beim Speichern validiert wird, wird das Wort „Geprüft" neben der Datei angezeigt. Die Regelsatzdatei wird zudem in der Modellstrukturanzeige angezeigt.

Fahren Sie mit „Regelblock erstellen" fort.

Regelblock erstellen

In diesem Abschnitt wird beschrieben, wie ein Regelblock innerhalb eines Regelsatzes oder innerhalb eines anderen Regelblocks erstellt wird.

Öffnen Sie zuerst die Eclipse-Workbench, wenn sie nicht bereits geöffnet ist. Öffnen Sie außerdem die Anzeige des Regelerstellungsprogramms.

Wenn Sie zum ersten Mal einen Regelsatz erstellen, müssen Sie vor der Erstellung von Regeln mindestens einen Regelblock innerhalb des Regelsatzes erstellen. Nachdem Sie diesen ersten Regelblock erstellt haben, können Sie weitere Regelblöcke erstellen, einschließlich Regelblöcke innerhalb von Regelblöcken.

Wenn Sie einen Regelblock erstellen möchten, führen Sie die folgenden Schritte aus:

1. Klicken Sie in der Navigatoranzeige doppelt auf den Namen der Regelsatzdatei, die Sie aktualisieren möchten. Die Datei wird anschließend in der Modellstrukturanzeige geöffnet. Wenn Sie in der Modellstrukturanzeige auf den Regelsatz klicken, werden die aktuellen Informationen für den Regelsatz im Editorbereich angezeigt.
2. Klicken Sie in der Modellstrukturanzeige mit der rechten Maustaste auf den Regelsatz.
3. Klicken Sie auf **New Child → Rule Block**. Anschließend wird ein Regelblock innerhalb des Regelsatzes in der Modellstrukturanzeige angezeigt, und die aktuellen Informationen für den Regelblock werden im Editorbereich angezeigt.

Sie können nun weitere Regelblöcke auf folgende Art und Weise erstellen:

- Wenn Sie einen Regelblock erstellen möchten, der einem vorhandenen Regelblock gleichgeordnet ist, klicken Sie mit der rechten Maustaste auf den vorhandenen Regelblock, und klicken Sie auf **Neues gleichgeordnetes Element → Regelblock**.
- Wenn Sie einen Regelblock innerhalb eines vorhandenen Regelblocks erstellen möchten, klicken Sie mit der rechten Maustaste auf den vorhandenen Regelblock, und klicken Sie auf **Neues untergeordnetes Element → Regelblock**.

Mit Hilfe des Editors können Sie zudem die Informationen für den Regelsatz und jeden einzelnen Regelblock aktualisieren. Fahren Sie mit „Regel erstellen" auf Seite 42 fort.

Regel erstellen

In diesem Abschnitt wird beschrieben, wie eine Regel erstellt wird.

Öffnen Sie zuerst die Eclipse-Workbench, wenn sie nicht bereits geöffnet ist. Öffnen Sie außerdem die Anzeige des Regelerstellungsprogramms.

Eine Regel muss innerhalb eines Regelblocks erstellt werden. Wenn Sie eine Regel erstellen möchten, führen Sie die folgenden Schritte aus:

1. Klicken Sie in der Modellstrukturanzeige mit der rechten Maustaste auf den Regelblock, den Sie aktualisieren möchten.
2. Klicken Sie auf **New Child** und auf den Typ der Regel, die Sie erstellen möchten. Die Regel wird dann innerhalb des Regelblocks in der Modellstrukturanzeige angezeigt, und die aktuellen Informationen für die Regel werden im Editorbereich angezeigt.

Sie können Regelblöcken auf dieselbe Art und Weise weitere Regeln hinzufügen. Mit Hilfe des Editors können Sie zudem die Informationen für die einzelnen Regeln aktualisieren. Fahren Sie mit „Regelsatz prüfen" fort.

Regelsatz prüfen

In diesem Abschnitt wird beschrieben, wie ein Regelsatz vor der Kompilierung geprüft wird.

Öffnen Sie zuerst die Eclipse-Workbench, wenn sie nicht bereits geöffnet ist. Öffnen Sie außerdem die Anzeige des Regelerstellungsprogramms.

Wenn Sie einen Regelsatz prüfen möchten, führen Sie die folgenden Schritte aus:

1. Klicken Sie in der Modellstrukturanzeige mit der rechten Maustaste auf eine Regel, einen Regelblock oder den Regelsatz.
2. Klicken Sie auf **Regelsatz prüfen**. Nach Beendigung der Prüfung wird ein Nachrichtenfenster mit möglichen Fehlern eingeblendet. Wenn die Prüfung erfolgreich abgeschlossen wurde, wird das Wort „Geprüft" rechts neben dem Dateinamen des Regelsatzes in der Navigatoranzeige angezeigt.

Wenn die Prüfung erfolgreich abgeschlossen wurde, fahren Sie mit „Regelsatz kompilieren" fort.

Regelsatz kompilieren

In diesem Abschnitt wird beschrieben, wie ein Regelsatz kompiliert wird.

Öffnen Sie zuerst die Eclipse-Workbench, wenn sie nicht bereits geöffnet ist. Öffnen Sie außerdem die Anzeige des Regelerstellungsprogramms.

Klicken Sie in der Navigatoranzeige oder in der Modellstrukturanzeige mit der rechten Maustaste auf den Regelsatz, den Sie kompilieren möchten, und klicken Sie auf **Regelsatz kompilieren**. Eventuelle Kompilierungsfehler werden in der Anzeige **Problems** angezeigt.

Wenn keine Kompilierungsfehler aufgetreten sind, wird der kompilierte Regelsatz standardmäßig in der Navigatoranzeige mit dem Standarddateityp `acts` angezeigt, der angibt, dass es sich um einen serialisierten Regelsatz handelt. In der Navigatoranzeige wird zudem das Wort „Kompiliert" rechts neben dem Dateinamen des Regelsatzes angezeigt.

Regelsatz aktualisieren

In diesem Abschnitt wird beschrieben, wie ein Regelsatz aktualisiert wird.

Öffnen Sie zuerst die Eclipse-Workbench, wenn sie nicht bereits geöffnet ist. Öffnen Sie außerdem die Anzeige des Regelerstellungsprogramms.

Klicken Sie in der Modellstrukturanzeige auf das Element (Regel, Regelblock oder Regelsatz), das Sie aktualisieren wollen. Die aktuellen Informationen für dieses Element werden im Editorbereich angezeigt, und Sie können diese Informationen mit dem Editor aktualisieren.

Gehen Sie wie folgt vor, um einen Peerregelblock oder eine Peerregel zu einem vorhandenen Regelblock bzw. zu einer vorhandenen Regel zu erstellen:

1. Klicken Sie mit der rechten Maustaste auf den vorhandenen Regelblock oder die vorhandene Regel.
2. Klicken Sie auf **New Sibling** und auf das Element (Regelblock oder Regeltyp), das Sie hinzufügen wollen.

Gehen Sie wie folgt vor, um einen Regelblock oder eine Regel in einem vorhandenen Regelblock zu erstellen:

1. Klicken Sie mit der rechten Maustaste auf den vorhandenen Regelblock.
2. Klicken Sie auf **New Child** und auf das Element (Regelblock oder Regeltyp), das Sie hinzufügen wollen.

Gehen Sie wie folgt vor, um auf andere Aktualisierungsfunktionen in der Modellstrukturanzeige zuzugreifen:

1. Klicken Sie mit der rechten Maustaste auf das Element (Regelblock oder Regel), das Sie aktualisieren wollen.
2. Klicken Sie auf den Menüpunkt für die Funktion wie z. B. **Delete**, **Copy** oder **Paste**.

Snippets in Ausdrücke innerhalb von Regeln einschließen

Sie können Snippets über die Anzeige **Snippet** in der Eclipse-Workbench in Ausdrücke innerhalb von Regeln einschließen.

Öffnen Sie zuerst die Eclipse-Workbench, wenn sie nicht bereits geöffnet ist. Öffnen Sie außerdem die Anzeige des Regelerstellungsprogramms.

Die Anzeige **Snippet** wird in der Anzeige des Regelerstellungsprogramms angezeigt. Snippets werden entsprechend ihrer Funktion in Kategorien zusammengefasst.

Wenn Sie ein Snippet aus der Anzeige **Snippet** einschließen möchten, führen Sie die folgenden Schritte aus:

1. Klicken Sie auf eine Snippetkategorie, um die Namen der Snippets in der Kategorie anzuzeigen.
2. Klicken Sie auf das Snippet, das Sie in einen Ausdruck einschließen möchten.
3. Ziehen Sie das Snippet in das entsprechende Feld für den Ausdruck. Der Code wird an der Cursorposition in das Feld für den Ausdruck eingefügt. Wenn der Code eine Eingabe durch den Regelautor erfordert, wie z. B. einen speziellen Nachrichtentext oder Variablenwerte, werden Sie aufgefordert, diese Eingabe vorzunehmen, bevor der Code in den Ausdruck eingeschlossen wird.

Fahren Sie mit „Regelsatz prüfen” auf Seite 42 fort.

Teil 2. Referenz für Regelautoren

Kapitel 4. Organisation eines Regelsatzes - Zusammenfassung

Diese Referenz listet alle Sprachelemente eines Regelsatzes, eines Regelblocks und jedes Regeltyps auf. Sie dient als Kurzübersicht zum Codieren eines Regelsatzes.

Tabelle 10 erläutert die Bedeutung der Schreibweise, die jedem Sprachelement folgt. *n* ist eine ungebundene Zahl.

Tabelle 10. Erläuterung der Schreibweise, die die Häufigkeit eines Sprachelements definiert

Schreibweise	Bedeutung
(0, 1)	0 gibt an, dass das Sprachelement optional ist. 1 gibt an, dass nur 1 Vorkommen zulässig ist, wenn es codiert ist.
(0, *n*)	0 gibt an, dass das Sprachelement optional ist. *n* gibt an, dass mehrere Vorkommen zulässig sind, wenn es codiert ist.
(1, 1)	Die erste 1 gibt an, dass das Sprachelement erforderlich ist. Die zweite 1 gibt an, dass nur 1 Vorkommen zulässig ist.
(1, *n*)	1 gibt an, dass das Sprachelement erforderlich ist. *n* gibt an, dass mehrere Vorkommen zulässig sind.
(2, *n*)	2 gibt an, dass 2 Vorkommen des Sprachelements erforderlich sind. *n* gibt an, dass weitere Vorkommen zulässig sind.

Die Elemente müssen in der angezeigten Reihenfolge codiert werden. Wenn ein Element optional ist, muss es nicht codiert werden. Alle codierten Elemente müssen jedoch die richtige Reihenfolge aufweisen.

Regelsatz - Zusammenfassung

In dieser Zusammenfassung werden die Sprachelemente eines Regelsatzes aufgelistet.

Regelsatzelemente

<ruleSet> enthält die folgenden Elemente:

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

  Zugehörige Verweise

  „Regelblock - Zusammenfassung" auf Seite 48
  In dieser Zusammenfassung werden die Sprachelemente eines Regelblocks aufgelistet.

Regelblock - Zusammenfassung

In dieser Zusammenfassung werden die Sprachelemente eines Regelblocks aufgelistet.

Regelblockelemente

<ruleBlock> enthält die folgenden Elemente.

Wenn sie codiert sind, müssen die Elemente <comment>, <import> und <variable> in der angezeigten Reihenfolge codiert werden. Die restlichen Elemente können in beliebiger Reihenfolge codiert werden.

- <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)

Zugehörige Verweise

„Datenerfassungsregel - Zusammenfassung" auf Seite 49
In dieser Zusammenfassung werden alle Sprachelemente der Datenerfassungsregel aufgeführt.

„Berechnungsregel - Zusammenfassung" auf Seite 50
In dieser Zusammenfassung werden alle Sprachelemente der Berechnungsregel aufgeführt.

„Duplikatregel - Zusammenfassung" auf Seite 51
In dieser Zusammenfassung werden alle Sprachelemente der Duplikatregel aufgeführt.

„Filterregel - Zusammenfassung" auf Seite 53
In dieser Zusammenfassung werden alle Sprachelemente der Filterregel aufgeführt.

„Sequenzregel - Zusammenfassung" auf Seite 54
In dieser Zusammenfassung werden alle Sprachelemente der Sequenzregel aufgeführt.

„Schwellenwertregel - Zusammenfassung" auf Seite 55
In dieser Zusammenfassung werden alle Sprachelemente der Schwellenwertregel aufgeführt.

„Zeitgeberregel - Zusammenfassung" auf Seite 57
In dieser Zusammenfassung werden alle Sprachelemente der Zeitgeberregel aufgeführt.

Datenerfassungsregel - Zusammenfassung

In dieser Zusammenfassung werden alle Sprachelemente der Datenerfassungsregel aufgeführt.

Regelelemente

<collectionRule> enthält die folgenden Elemente:

- <comment> (0, 1)
- <variable> (0, n)
  - <comment> (0, 1)
  - <varInitializer> (1, 1)
- <activationInterval> (0, 1)
  - <activationTime> (0, 1)
    - <start> (0, 1)
      - Eines der folgenden drei Elemente (1, 1):
        - <dateTime>
        - <whenLoaded>
        - <inactiveWhenLoaded>
    - <stop> (0, 1)
      - Eines der folgenden drei Elemente (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)
  - Die folgenden drei Elemente in beliebiger Reihenfolge (1, n):
    - <attributeAlias>
      - <eventAttribute> (2, n)
    - <attributeName>
    - <computedValue>
- <timeWindow> (1, 1)
  - Eines der beiden folgenden Elemente (1, 1):
    - <timeInterval>
    - <runUntilDeactivated>
- <onTimeWindowComplete> (0, 1)
  - <action> (0, n)

Berechnungsregel - Zusammenfassung

In dieser Zusammenfassung werden alle Sprachelemente der Berechnungsregel aufgeführt.

Regelelemente

<computationRule> enthält die folgenden Elemente:

- <comment> (0, 1)
- <variable> (0, n)
  - <comment> (0, 1)
  - <varInitializer> (1, 1)
- <activationInterval> (0, 1)
  - <activationTime> (0, 1)
    - <start> (0, 1)
      - Eines der folgenden drei Elemente (1, 1):
        - <dateTime>
        - <whenLoaded>
        - <inactiveWhenLoaded>
    - <stop> (0, 1)
      - Eines der folgenden drei Elemente (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)
  - Die folgenden drei Elemente in beliebiger Reihenfolge (1, n):
    - <attributeAlias>
      - <eventAttribute> (2, n)
    - <attributeName>
    - <computedValue>
- <computeFunction> (1, 1)
- <timeWindow> (1, 1)
  - Eines der beiden folgenden Elemente (1, 1):
    - <timeInterval>
    - <runUntilDeactivated>
- <onTimeWindowComplete> (0, 1)
  - <action> (0, n)

Duplikatregel - Zusammenfassung

In dieser Zusammenfassung werden alle Sprachelemente der Duplikatregel aufgeführt.

Regelelemente

<duplicateRule> enthält die folgenden Elemente:

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

- <activationTime> (0, 1)
  - <start> (0, 1)
    - Eines der folgenden drei Elemente (1, 1):
      - <dateTime>
      - <whenLoaded>
      - <inactiveWhenLoaded>
  - <stop> (0, 1)
    - Eines der folgenden drei Elemente (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)
  - Die folgenden drei Elemente in beliebiger Reihenfolge (1, n):
    - <attributeAlias>
      - <eventAttribute> (2, n)
    - <attributeName>
    - <computedValue>
- <timeWindow> (1, 1)
  - Eines der beiden folgenden Elemente (1, 1):

- <timeInterval>
- <runUntilDeactivated>

- <onDetection> (0, 1)
  - <action> (0, n)
- <onNextEvent> (0, 1)
  - <action> (0, n)
- <onTimeWindowComplete> (0, 1)
  - <action> (0, n)

Filterregel - Zusammenfassung

In dieser Zusammenfassung werden alle Sprachelemente der Filterregel aufgeführt.

Regelelemente

<filterRule> enthält die folgenden Elemente:

- <comment> (0, 1)
- <variable> (0, n)
  - <comment> (0, 1)
  - <varInitializer> (1, 1)
- <activationInterval> (0, 1)
  - <activationTime> (0, 1)
    - <start> (0, 1)
      - Eines der folgenden drei Elemente (1, 1):
        - <dateTime>
        - <whenLoaded>
        - <inactiveWhenLoaded>
    - <stop> (0, 1)
      - Eines der folgenden drei Elemente (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)

Sequenzregel - Zusammenfassung

In dieser Zusammenfassung werden alle Sprachelemente der Sequenzregel aufgeführt.

Regelelemente

<sequenceRule> enthält die folgenden Elemente:

- <comment> (0, 1)
- <variable> (0, n)
  - <comment> (0, 1)
  - <varInitializer> (1, 1)
- <activationInterval> (0, 1)
  - <activationTime> (0, 1)
    - <start> (0, 1)
      - Eines der folgenden drei Elemente (1, 1):
        - <dateTime>
        - <whenLoaded>
        - <inactiveWhenLoaded>
    - <stop> (0, 1)
      - Eines der folgenden drei Elemente (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)
  - Die folgenden drei Elemente in beliebiger Reihenfolge (1, n):
    - <attributeAlias>
      - <eventAttribute> (2, n)
    - <attributeName>
    - <computedValue>
- <timeWindow> (1, 1)
  - Eines der beiden folgenden Elemente (1, 1):
    - <timeInterval>
    - <runUntilDeactivated>
- <onDetection> (0, 1)
  - <action> (0, n)
- <onTimeOut> (0, 1)
  - <action> (0, n)

Schwellenwertregel - Zusammenfassung

In dieser Zusammenfassung werden alle Sprachelemente der Schwellenwertregel aufgeführt.

Regelelemente

<thresholdRule> enthält die folgenden Elemente:

- <comment> (0, 1)
- <variable> (0, n)
  - <comment> (0, 1)
  - <varInitializer> (1, 1)
- <activationInterval> (0, 1)
  - <activationTime> (0, 1)
    - <start> (0, 1)
      - Eines der folgenden drei Elemente (1, 1):
        - <dateTime>
        - <whenLoaded>
        - <inactiveWhenLoaded>
    - <stop> (0, 1)
      - Eines der folgenden drei Elemente (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)
  - Die folgenden drei Elemente in beliebiger Reihenfolge (1, n):
    - <attributeAlias>
      - <eventAttribute> (2, n)
    - <attributeName>
    - <computedValue>
- Eines der folgenden drei Elemente (1, 1):
  - <booleanThreshold>
  - <computedThreshold>
  - <eventCountThreshold>
- <timeWindow> (1, 1)
  - Eines der beiden folgenden Elemente (1, 1):
    - <timeInterval>
    - <runUntilDeactivated>
- <onDetection> (0, 1)
  - <action> (0, n)
- <onTimeOut> (0, 1)

– <action> (0, n)

Zeitgeberregel - Zusammenfassung

In dieser Zusammenfassung werden alle Sprachelemente der Zeitgeberregel aufgeführt.

Regelelemente

<timerRule> enthält die folgenden Elemente:

- <comment> (0, 1)
- <variable> (0, n)
  - <comment> (0, 1)
  - <varInitializer> (1, 1)
- <activationInterval> (0, 1)
  - <activationTime> (0, 1)
    - <start> (0, 1)
      - Eines der folgenden drei Elemente (1, 1):
        - <dateTime>
        - <whenLoaded>
        - <inactiveWhenLoaded>
    - <stop> (0, 1)
      - Eines der folgenden drei Elemente (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)
  - Eines der beiden folgenden Elemente (1, 1):
    - <timeInterval>
    - <runUntilDeactivated>
- <onTimeWindowComplete> (0, 1)
  - <action> (0, n)

Kapitel 5. Sprachelementreferenz

Diese Referenz beschreibt die Details der Sprachelemente im XML-Schema für die ACT-Regelsprache. Die Sprachelemente sind in alphabetischer Reihenfolge aufgeführt, und die Attribute, die für die einzelnen Elemente verfügbar sind, werden im Abschnitt für das jeweilige Element beschrieben.

In XML sowie in anderen Markup-Sprachen, wie z. B. SGML und HTML, stellt ein Element eine Basiseinheit dar, die aus einem Startbefehl, einem Endbefehl, zugeordneten Attributen und ihren Werten sowie aus Text besteht, der sich zwischen den Start- und Endbefehlen befindet. Ein Attribut ist ein Name/Wert-Paar, das mit einem Element codiert wird, um bestimmte Merkmale des Elements zu definieren. Ein Attribut weist einen Datentyp auf, der die Art der Informationen kennzeichnet, die durch den Attributwert bereitgestellt werden (beispielsweise numerische Informationen, Textinformationen oder Boolesche Informationen).

In XML ist ein Namespace ein Uniform-Resource-Identifier (URI), der einen eindeutigen Namen bereitstellt, der den Elementen und Typendefinitionen in einem Schema zugeordnet wird. Der URI gibt an, welches XML-Schema die Definition eines Elements enthält. Ein Namespace wird mit einer Präfixzeichenfolge gefolgt von einem Doppelpunkt angegeben. Das ACT-Regelsprachschema ist in drei verschiedenen Dateien definiert und verwendet die folgenden drei Namespaces:

xsd: Dieser Namespace gibt an, dass das Sprachelement im Standard-XML-Schema definiert ist, das unter http://www.w3.org beschrieben wird.

br: Dieser Namespace gibt an, dass das Sprachelement im Basisregelsatzschema von Active Correlation Technology definiert ist, das sich in der Datei ACTparser.jar im Unterverzeichnis com/ibm/correlation/ruleparser/xml/RuleSetBase.xsd befindet. `br:ruleSet` beispielsweise bezieht sich auf das Element 'ruleSet', das in der Datei RuleSetBase.xsd definiert ist.

act: Dieser Namespace gibt an, dass das Sprachelement im ACT-Regelsprachschema definiert ist, das sich in der Datei ACTparser.jar im Unterverzeichnis com/ibm/correlation/ruleparser/xml/ACTL.xsd befindet. `act:ruleSet` beispielsweise bezieht sich auf das Element 'ruleSet', das in der Datei ACTL.xsd definiert ist.

Im Regelsprachschema werden Sprachelemente als Elemente oder als komplexe Typen definiert. Beispiel:

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

Im Schema definieren die Attribute 'minOccurs' und 'maxOccurs' jeweils die minimale und maximale Anzahl Vorkommen für ein Sprachelement. Tabelle 11 auf Seite 60 beschreibt die Bedeutung verschiedener Werte für die Attribute 'minOccurs' und 'maxOccurs'.

Tabelle 11. Attribute im Schema, die die Häufigkeit für ein Sprachelement definieren

Attribut	Attributwert	Bedeutung
minOccurs	0	Das Sprachelement ist optional.
minOccurs	1	Das Sprachelement muss mindestens einmal auftreten. 1 ist der Standardwert für das Attribut 'minOccurs'.
minOccurs	2	Das Sprachelement muss mindestens zweimal auftreten.
maxOccurs	1	Das Sprachelement kann nicht mehr als einmal auftreten. 1 ist der Standardwert für das Attribut 'maxOccurs'.
maxOccurs	unbounded	Das Sprachelement kann beliebig oft auftreten.

Element 'action'

Das Element <action> enthält einen Ausdruck, der eine Regelantwortaktion oder eine Lebenszyklusaktion definiert.

Details

Informationen zu den Variablen, die in Ausdrücken verwendet werden können, finden Sie in „Variablen" auf Seite 25. Die Verwendung bestimmter Variablen hängt vom Kontext des Ausdrucks ab.

Attribute

<action> weist die folgenden Attribute auf:

Tabelle 12. Attribute des Elements <action>

Name	Beschreibung	Datentyp	Erforderlich?
expressionLanguage	Gibt die Programmiersprache an, in der der Ausdruck geschrieben wurde. Da die Programmiersprache Java die einzige unterstützte Sprache für Ausdrücke ist, ist nur der Wert `java` für dieses Attribut gültig.	xsd:NMTOKEN	Ja
name	Kennzeichnet die Aktion. Dieser Bezeichner kann für die Fehlerbehebung hilfreich sein, insbesondere wenn es sich um einen eindeutigen Namen unter allen <action>-Elementen handelt, die für eine bestimmte Regelantwort- oder Lebenszyklusaktion definiert sind.	xsd:NMTOKEN	Nein

Enthalten in

<action> ist in den folgenden Elementen enthalten:

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

- <onTimeWindowComplete>
- <onUnload>

Enthält

<action> enthält keine Elemente.

Zugehörige Konzepte

„Ausdrücke” auf Seite 20
Ein Ausdruck ist Code, der eine angepasste Logik enthält, die einer Regel hinzugefügt werden kann. Ausdrücke können auch auf Code außerhalb der ACT-Engine zugreifen. In der Regelsprache sind Ausdrücke nur in bestimmten Kontexten, oder Regelsprachelementen, gültig.

Element ’activateOnEvent’

Das Element <activateOnEvent> definiert die Ereignisse, die die Regel aktivieren können, oder eine Regelinstanz für Regeln, die mit einem <groupingKey>-Element definiert sind.

Im Folgenden werden die drei Möglichkeiten aufgeführt, Ereignisse auszuwählen:

- Die Verwendung mindestens eines <eventType>-Elements mit einem <filteringPredicate>-Element
- Die Verwendung mindestens eines <eventType>-Elements ohne <filteringPredicate>-Element
- Die Verwendung eines <filteringPredicate>-Elements ohne <eventType>-Elemente

Wenn die Regel inaktiv ist und kein <eventType>- oder <filteringPredicate>-Element codiert ist, wird jedes beliebige Ereignis ausgewählt, das auftritt.

Wenn Sie keine <eventType>-Elemente codieren, kann dies negative Auswirkungen auf die Systemleistung haben.

Angenommen, Sie möchten alle Ereignisse des Typs `Audit Failure` auswählen. Sie können ein Filterprädikat verwenden, um die Auswahlkriterien weiter einzugrenzen, damit nur die Ereignisse eingeschlossen werden, die ein Ereignisattribut mit einem bestimmten Wert aufweisen. Beispiel: Sie codieren ein <eventType>-Element, um alle Ereignisse des Typs `Audit Failure` auszuwählen und ein <filteringPredicate>-Element, um nur die Ereignisse auszuwählen, die ein Hostnamensattribut mit dem Wert `MyCriticalSystem` aufweisen.

Attribute

<activateOnEvent> weist keine Attribute auf.

Enthalten in

<activateOnEvent> ist in den folgenden Elementen enthalten:

- <activationInterval>
- <activationByGroupingKey>

Enthält

<activateOnEvent> enthält die folgenden Elemente.

Die Elemente müssen in der angezeigten Reihenfolge codiert werden. Wenn ein Element optional ist, muss es nicht codiert werden. Alle codierten Elemente müssen jedoch die richtige Reihenfolge aufweisen.

Tabelle 13. Im Element <activateOnEvent> enthaltene Elemente

Element	Erforderlich oder optional?
<eventType>	Optional. 0 oder mehr Vorkommen sind zulässig.
<filteringPredicate>	Optional. 0 oder 1 Vorkommen ist zulässig.
<stopAfter>	Dieses Element ist nur dann gültig, wenn das Element <activateOnEvent> im Element <activationByGroupingKey> enthalten ist. Optional. 0 oder 1 Vorkommen ist zulässig.

Element 'activationByGroupingKey'

Das Element <activationByGroupingKey> enthält Elemente, die die Ereignisse angeben, die eine Regelinstanz aktivieren und inaktivieren können, die durch das Element <groupingKey> definiert wird. Da das Element <groupingKey> für die Filter- und Zeitgeberregeln nicht gültig ist, wird das Element <activationByGroupingKey> für diese Regeln nicht angewendet.

Details

Die Funktion des Elements <activationByGroupingKey> dient zur Verwendung in Regeln, bei denen ein Gruppierungsschlüssel definiert wird. Es ermöglicht Ihnen, die Aktivierung und Inaktivierung von Regelinstanzen auf der Basis des Gruppierungsschlüssels zu steuern. Wenn Sie das Element <activationByGroupingKey> codieren, kann jede Regelinstanz individuell auf der Basis der <activateOnEvent>- und <deactivateOnEvent>-Bedingungen innerhalb von <activationByGroupingKey> aktiviert und inaktiviert werden.

Das folgende Beispiel zeigt die Verwendung des Elements <activationByGroupingKey> innerhalb einer Berechnungsregel.

- Die folgende Berechnungsregel akzeptiert Ereignisse des Typs `StockSharesTraded`. Diese Ereignisse geben die Anzahl Aktienanteile an, die für viele verschiedene Unternehmen gehandelt werden.
- Der Gruppierungsschlüssel ist das stockSymbol-Attribut eines Ereignisses. Der Wert des stockSymbol-Attributs ist der Name eines bestimmten Unternehmens.
- Der Wert des sharesTraded-Attributs eines Ereignisses ist die Anzahl Aktienanteile, die für das entsprechende Unternehmen gehandelt wurden (das Unternehmen wird durch den Wert des stockSymbol-Attributs angegeben).
- Für ein bestimmtes Unternehmen wird ein Bericht erstellt, der die Anzahl Aktienanteile angibt, die für dieses Unternehmen während eines Zeitraums von 10 Minuten gehandelt wurden. Bevor die Berechnungsregel jedoch diesen Bericht erstellen kann, muss sie ein Ereignis des Typs `StartReporting` mit dem Namen des entsprechenden Unternehmens als Wert des stockSymbol-Attributs empfangen.

```
<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>
```

Attribute

<activationByGroupingKey> weist keine Attribute auf.

Enthalten in

<activationByGroupingKey> ist im folgenden Element enthalten:

- <activationInterval>

Enthält

<activationByGroupingKey> enthält die folgenden Elemente.

Die Elemente müssen in der angezeigten Reihenfolge codiert werden. Wenn ein Element optional ist, muss es nicht codiert werden. Alle codierten Elemente müssen jedoch die richtige Reihenfolge aufweisen.

Tabelle 14. Im Element <activationByGroupingKey> enthaltene Elemente

Element	Erforderlich oder optional?
<activateOnEvent>	Optional. 0 oder 1 Vorkommen ist zulässig.
<deactivateOnEvent>	Optional. 0 oder 1 Vorkommen ist zulässig.

Beziehung zwischen den Elementen <activationInterval> und <activationByGroupingKey>

Die Elemente <activateOnEvent> und <deactivateOnEvent> sind in den beiden folgenden Elementen enthalten:

- <activationInterval>
- <activationByGroupingKey>, das auch in <activationInterval> enthalten ist

Das Regelverhalten kann sich basierend auf der aktuellen Regelaktivität und auf den Interaktionen zwischen den Definitionen für <activateOnEvent> und <deactivateOnEvent> innerhalb der Elemente <activationInterval> und <activationByGroupingKey> unterscheiden. Das folgende Beispiel zeigt, wie diese Definitionen interagieren können.

In diesem Beispiel wird eine Duplikatregel definiert, um Ereignisse von Systemen zu unterdrücken, die sich im Wartungsmodus befinden, und um am Ende der Wartungsperiode einen Ergebnisbericht über die Anzahl der unterdrückten Ereignisse bereitzustellen.

Standardmäßig lässt eine Regel, in der ein Gruppierungsschlüssel definiert ist, zu, dass alle Gruppierungsschlüsselwerte verarbeitet werden. Wenn Ereignisse die Ereignisauswahlkriterien für die Regel erfüllen, sind daher alle Regelinstanzen aktiv und bereit, diese Ereignisse mit beliebigen Werten des Gruppierungsschlüssels zu akzeptieren. Das Aktivierungsintervall für die Regel ist mit dem Intervall identisch, das vorliegen würde, wenn die Regel keinen Gruppierungsschlüssel aufweisen würde, da im Prinzip alle Ereignisse verarbeitet werden, die die Ereignisauswahlkriterien für die Regel erfüllen.

Im folgenden Beispiel lautet der Gruppierungsschlüssel `hostname`, und die Definitionen im Element <activationInterval> geben die folgenden Aktionen an:

1. Alle Regelinstanzen aktivieren, wenn ein Ereignis des Typs `StartMaintenanceModeAllHosts` empfangen wird.
2. Alle Regelinstanzen nach 2 Stunden inaktivieren, oder wenn ein Ereignis des Typs `StopMaintenanceModeAllHosts` empfangen wird.

```
<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>
```

In einigen Situationen möchten Sie möglicherweise steuern, welche Regelinstanzen aktiv werden und wann sie aktiv werden. Für diese Fälle sollten Sie das Element <activationByGroupingKey> codieren.

Das folgende Beispiel erweitert das vorherige Beispiel und zeigt, wie Sie mit dem Gruppierungsschlüsselwert auswählen können, welche Regelinstanzen verarbeitet werden sollen. Die Definitionen im Element <activationByGroupingKey> geben die folgenden Aktionen an:

1. Die Verarbeitung der Regelinstanzen für bestimmte Hostnamen zulassen, wenn Ereignisse des Typs `StartMaintenanceMode` für diese Hostnamen empfangen werden.
2. Diese Regelinstanzen 2 Stunden nach der Aktivierung inaktivieren, oder wenn ein Ereignis des Typs `StopMaintenanceMode` für den entsprechenden Hostnamen empfangen wird.

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

Die folgenden Aussagen fassen zusammen, was geschieht, wenn Sie das Element <activationByGroupingKey> codieren:

- Wenn das Element <activateOnEvent> im Element <activationByGroupingKey> codiert wird, können nur die Ereignisse verarbeitet werden, die denselben Gruppierungsschlüsselwert wie das Ereignis haben, das die Bedingung für <activateOnEvent> in <activationByGroupingKey> erfüllte.
- Wenn das Element <deactivateOnEvent> im Element <activationByGroupingKey> codiert wird, können die Ereignisse *nicht* verarbeitet werden, die denselben Gruppierungsschlüsselwert wie das Ereignis haben, das die Bedingung für <activationByGroupingKey> <deactivateOnEvent> erfüllte.

Auswirkungen verschiedener Aktivierungs- und Inaktivierungsdefinitionen auf den Regelstatus

Tabelle 15 auf Seite 66 und Tabelle 16 auf Seite 67 zeigen, wie der Status einer Regel von verschiedenen Aktivierungs- und Inaktivierungdefinitionen betroffen wird. In diesen Tabellen werden die folgenden Konventionen verwendet:

- *A* ist ein aktivierendes Ereignis.
- In der Notation „*A[x]*" steht *x* für den Gruppierungsschlüsselwert. Beispiel: *A[1]* ist ein aktivierendes Ereignis mit dem Gruppierungsschlüsselwert 1.
- *D* ist ein inaktivierendes Ereignis.
- In der Notation „*D[x]*" steht *x* für den Gruppierungsschlüsselwert. Beispiel: *D[1]* ist ein inaktivierendes Ereignis mit dem Gruppierungsschlüsselwert 1.

Tabelle 15. Regelstatusänderungen basierend auf verschiedenen Aktivierungsdefinitionen

Startstatus der Regel	**Mögliche Auswirkung auf Regelstatus**	**Endstatus der Regel**
Inaktiv	Definierte Zeit in <activationInterval> <activationTime> <start>	1. Die Regel wird aktiviert. 2. Die Aktionen <onActivation> werden ausgeführt. 3. Alle Gruppierungsschlüsselwerte sind zulässig.
	Methode activate()	
	Ereignis *A* (definiert in <activationInterval> <activateOnEvent>)	
	Ereignis *A[1]* (definiert in <activationByGroupingKey> <activateOnEvent> *ohne* <stopAfter>)	1. Die Regel wird aktiviert. 2. Die Aktionen <onActivation> werden ausgeführt. 3. Nur der Gruppierungsschlüsselwert 1 ist zulässig. Wenn das Regelmuster für diese Regelinstanz übereinstimmt, ist der Gruppierungsschlüsselwert 1 nicht mehr zulässig.
	Ereignis *A[2]* (definiert in <activateOnEvent> *mit* <stopAfter>)	1. Die Regel wird aktiviert. 2. Die Aktionen <onActivation> werden ausgeführt. 3. Nur der Gruppierungsschlüsselwert 2 ist zulässig. Er ist allerdings nur für die Dauer zulässig, die durch das Element <stopAfter> angegeben wird. Während dieser Zeitdauer kann das Regelmuster für diese Regelinstanz mehrfach übereinstimmen.
• Aktiv • Alle Gruppierungsschlüsselwerte sind zulässig	Definierte Zeit in <activationInterval> <activationTime> <start>	Der Regelstatus wurde nicht geändert. Er ist mit dem Startstatus der Regel identisch.
	Methode activate()	
	Ereignis *A* (definiert in <activationInterval> <activateOnEvent>)	
	Ereignis *A[1]* (definiert in <activationByGroupingKey> <activateOnEvent> *ohne* <stopAfter>)	
	Ereignis *A[2]* (definiert in <activateOnEvent> *mit* <stopAfter>)	

Tabelle 15. Regelstatusänderungen basierend auf verschiedenen Aktivierungsdefinitionen (Forts.)

Startstatus der Regel	Mögliche Auswirkung auf Regelstatus	Endstatus der Regel
• Aktiv • Es sind nur Gruppierungsschlüsselwerte zulässig, die basierend auf den Definitionen <activationByGroupingKey> <activateOnEvent> Regelinstanzen ausgelöst haben	Definierte Zeit in <activationInterval> <activationTime> <start>	Der Regelstatus wurde nicht geändert. Er ist mit dem Startstatus der Regel identisch.
	Methode activate()	
	Ereignis *A* (definiert in <activationInterval> <activateOnEvent>)	Alle Gruppierungsschlüsselwerte sind zulässig.
	Ereignis *A[1]* (definiert in <activationByGroupingKey> <activateOnEvent> *ohne* <stopAfter>)	• Der Gruppierungsschlüsselwert 1 ist jetzt zusätzlich zu den bereits gültigen Gruppierungsschlüsselwerten zulässig. • Wenn das Regelmuster für diese Regelinstanz übereinstimmt, ist der Gruppierungsschlüsselwert 1 nicht mehr zulässig.
	Ereignis *A[2]* (definiert in <activateOnEvent> *mit* <stopAfter>)	• Der Gruppierungsschlüsselwert 2 ist jetzt zusätzlich zu den bereits gültigen Gruppierungsschlüsselwerten zulässig. • Dieser Wert ist nur für die Dauer zulässig, die durch das Element <stopAfter> angegeben wird. • Während dieser Zeitdauer kann das Regelmuster für diese Regelinstanz mehrfach übereinstimmen.
• Aktiv • Alle Gruppierungsschlüsselwerte sind zulässig, mit Ausnahme der Werte, die basierend auf den Definitionen <activationByGroupingKey> <deactivateOnEvent> nicht zulässig sind	Definierte Zeit in <activationInterval> <activationTime> <start>	Der Regelstatus wurde nicht geändert. Er ist mit dem Startstatus der Regel identisch.
	Methode activate()	
	Ereignis *A* (definiert in <activationInterval> <activateOnEvent>)	Alle Gruppierungsschlüsselwerte sind zulässig.
	Ereignis *A[1]* (definiert in <activationByGroupingKey> <activateOnEvent> *ohne* <stopAfter>)	Der Gruppierungsschlüsselwert 1 ist jetzt zusätzlich zu den bereits gültigen Gruppierungsschlüsselwerten zulässig.
	Ereignis *A[2]* (definiert in <activateOnEvent> *mit* <stopAfter>)	Der Gruppierungsschlüsselwert 2 ist jetzt zusätzlich zu den bereits gültigen Gruppierungsschlüsselwerten zulässig.

Tabelle 16. Regelstatusänderungen basierend auf verschiedenen Inaktivierungsdefinitionen

Startstatus der Regel	Mögliche Auswirkung auf Regelstatus	Endstatus der Regel
Inaktiv	Definierte Zeit in <activationInterval> <activationTime> <stop>	Der Regelstatus wurde nicht geändert. Er ist mit dem Startstatus der Regel identisch.
	Methode deactivate()	
	Ereignis *D* (definiert in <activationInterval> <deactivateOnEvent>)	
	Ereignis *D[1]* (definiert in <activationByGroupingKey> <deactivateOnEvent>)	
	Definierte Dauer in <activationByGroupingKey> <activateOnEvent> <stopAfter> endet für Ereignis *A[2]*	

Tabelle 16. Regelstatusänderungen basierend auf verschiedenen Inaktivierungsdefinitionen (Forts.)

Startstatus der Regel	Mögliche Auswirkung auf Regelstatus	Endstatus der Regel
• Aktiv • Alle Gruppierungsschlüsselwerte sind zulässig	Definierte Zeit in <activationInterval> <activationTime> <stop>	1. Alle Regelinstanzen werden inaktiviert. 2. Die Aktionen <onDeactivation> werden ausgeführt. 3. Die Regel wird inaktiviert.
	Methode deactivate()	
	Ereignis *D* (definiert in <activationInterval> <deactivateOnEvent>)	
	Ereignis *D[1]* (definiert in <activationByGroupingKey> <deactivateOnEvent>)	• Der Gruppierungsschlüsselwert 1 ist nicht mehr zulässig. • Wenn eine Regelinstanz mit dem Gruppierungsschlüsselwert 1 aktiv ist, wird sie inaktiviert.
	Definierte Dauer in <activationByGroupingKey> <activateOnEvent> <stopAfter> endet für Ereignis *A[2]*	Eine Regelinstanz mit dem Gruppierungsschlüsselwert 2 wird inaktiviert.
• Aktiv • Es sind nur Gruppierungsschlüsselwerte zulässig, die basierend auf den Definitionen <activationByGroupingKey> <activateOnEvent> Regelinstanzen ausgelöst haben	Definierte Zeit in <activationInterval> <activationTime> <stop>	1. Alle Regelinstanzen werden inaktiviert. 2. Die Aktionen <onDeactivation> werden ausgeführt. 3. Die Regel wird inaktiviert.
	Methode deactivate()	
	Ereignis *D* (definiert in <activationInterval> <deactivateOnEvent>)	
	Ereignis *D[1]* (definiert in <activationByGroupingKey> <deactivateOnEvent>)	• Der Gruppierungsschlüsselwert 1 ist nicht mehr zulässig. • Wenn eine Regelinstanz mit dem Gruppierungsschlüsselwert 1 aktiv ist, wird sie inaktiviert.
	Definierte Dauer in <activationByGroupingKey> <activateOnEvent> <stopAfter> endet für Ereignis *A[2]*	• Der Gruppierungsschlüsselwert 2 ist nicht mehr zulässig. • Eine Regelinstanz mit dem Gruppierungsschlüsselwert 2 wird inaktiviert.
• Aktiv • Alle Gruppierungsschlüsselwerte sind zulässig, mit Ausnahme der Werte, die basierend auf den Definitionen <activationByGroupingKey> <deactivateOnEvent> nicht zulässig sind	Definierte Zeit in <activationInterval> <activationTime> <stop>	1. Alle Regelinstanzen werden inaktiviert. 2. Die Aktionen <onDeactivation> werden ausgeführt. 3. Die Regel wird inaktiviert.
	Methode deactivate()	
	Ereignis *D* (definiert in <activationInterval> <deactivateOnEvent>)	
	Ereignis *D[1]* (definiert in <activationByGroupingKey> <deactivateOnEvent>)	• Der Gruppierungsschlüsselwert 1 ist nicht mehr zulässig. • Wenn eine Regelinstanz mit dem Gruppierungsschlüsselwert 1 aktiv ist, wird sie inaktiviert.
	Definierte Dauer in <activationByGroupingKey> <activateOnEvent> <stopAfter> endet für Ereignis *A[2]*	Eine Regelinstanz mit dem Gruppierungsschlüsselwert 2 wird inaktiviert.

Element 'activationInterval'

Das Element <activationInterval> enthält Elemente, die definieren, wann eine Regel aktiv und inaktiv ist.

Details

Eine Regel kann zu einem diskreten Zeitpunkt oder durch ein bestimmtes Ereignis aktiviert oder inaktiviert werden.

Wenn Sie angeben, dass eine Regel zu einem diskreten Zeitpunkt *und* durch ein bestimmtes Ereignis aktiviert oder inaktiviert werden soll, wird die Regel aktiviert oder inaktiviert, sobald der Zeitpunkt erreicht oder das Ereignis empfangen wird, je nach dem, was zuerst eintritt. In diesem Fall wird die Regel während ihres Lebenszyklus jedoch möglicherweise durch eine Vielzahl Ereignisse aktiviert oder inaktiviert. Zum Beispiel wird eine Regel möglicherweise durch ein Ereignis aktiviert, inaktiviert, zu einem definierten Zeitpunkt aktiviert, erneut inaktiviert und durch ein weiteres Ereignis aktiviert.

In einem wirtschaftlichen Umfeld möchten Sie vielleicht, dass eine Regel aktiviert wird, wenn ein Ereignis empfangen wird, das angibt, dass die Börse für den Handel geöffnet wurde. In einer IT-Umgebung möchten Sie vielleicht, dass ein Wartungsfenster ab dem 29. Oktober 2006 um 06:00 Uhr beginnt und zu einem der folgenden Zeitpunkte endet, je nach dem, was zuerst eintritt:

- Am 30. Oktober 2006 um 11:30 Uhr
- Wenn ein Ereignis empfangen wird, das angibt, dass die Wartungsarbeiten abgeschlossen sind

Attribute

<activationInterval> weist keine Attribute auf.

Enthalten in

<activationInterval> ist in den folgenden Elementen enthalten:

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

Enthält

<activationInterval> enthält die folgenden Elemente.

Die Elemente müssen in der angezeigten Reihenfolge codiert werden. Wenn ein Element optional ist, muss es nicht codiert werden. Alle codierten Elemente müssen jedoch die richtige Reihenfolge aufweisen.

Tabelle 17. Im Element <activationInterval> enthaltene Elemente

Element	Erforderlich oder optional?
<activationTime>	Optional. 0 oder 1 Vorkommen ist zulässig.
<activateOnEvent>	Optional. 0 oder 1 Vorkommen ist zulässig.
<deactivateOnEvent>	Optional. 0 oder 1 Vorkommen ist zulässig.
<activationByGroupingKey>	Optional. 0 oder 1 Vorkommen ist zulässig.

Beziehungen zwischen enthaltenen Elementen

Die Elemente <start> und <stop>, die im Element <activationTime> enthalten sind, stellen eine statische Methode dar, eine Regel zu aktivieren und zu inaktivieren. Durch diese Elemente wird eine Regel zu einem diskreten Zeitpunkt aktiviert oder inaktiviert. Im Gegensatz dazu stellen die Elemente <activateOnEvent> und <deactivateOnEvent> eine dynamische Methode dar, eine Regel zu aktivieren und zu inaktivieren. Durch diese Elemente wird eine Regel aktiviert oder inaktiviert, wenn ein bestimmtes Ereignis auftritt. Beispiel: Eine Regel wird durch ein beliebiges Ereignis aktiviert, das die Bedingungen erfüllt, die für das Element <activateOnEvent> definiert sind, falls die Regel nicht bereits aktiv ist. Eine Regel wird durch ein beliebiges Ereignis inaktiviert, das die Bedingungen erfüllt, die für das Element <deactivateOnEvent> definiert sind, falls die Regel nicht bereits inaktiv ist. Daher können bestimmte Ereignisse die statische Definition ändern, wann eine Regel aktiviert oder inaktiviert wird.

Tabelle 18 beschreibt, wie und wann eine Regel auf der Basis bestimmter Kombinationen aktiviert oder inaktiviert wird, in denen die folgenden Elemente codiert sein können:

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

In Tabelle 18 steht *X* für den Namen eines Ereignisses, das die Regel aktiviert, und*Y* steht für den Namen eines Ereignisses, das die Regel inaktiviert.

Wenn das Element <start> überhaupt nicht codiert ist, entspricht die Standardstartzeit der Zeit, die durch das Element <whenLoaded> definiert wird.

Wenn das Element <stop> überhaupt nicht codiert ist, entspricht die Standardstoppzeit der Zeit, die durch das Element <never> definiert wird.

Tabelle 18. Regelaktivitäten basierend auf der Codierung verschiedener Kombinationen der Elemente, die im Element <activationInterval> enthalten sind

<activationTime>				
<start>	**<stop>**	**<activateOnEvent>**	**<deactivateOnEvent>**	**Regelaktivität**
<whenLoaded>	<never>			Die Regel ist aktiv, wenn sie geladen wird, und bleibt aktiv, während die ACT-Engine ausgeführt wird.
<whenLoaded>	<never>		*Y*	Wenn die Regel geladen wird, ist sie aktiv. Ereignis *Y* inaktiviert die Regel.
<whenLoaded>	<never>	*X*	*Y*	Wenn die Regel geladen wird, ist sie aktiv. Ereignis *Y* inaktiviert die Regel, und Ereignis *X* reaktiviert sie. Diese Inaktivierung und Reaktivierung kann mehrfach ausgeführt werden.
<whenLoaded>	<after>			Die Regel ist aktiv, wenn sie geladen wird, und wird nach einem angegebenen Zeitintervall inaktiviert.

Tabelle 18. Regelaktivitäten basierend auf der Codierung verschiedener Kombinationen der Elemente, die im Element <activationInterval> enthalten sind (Forts.)

<activationTime>				
<start>	**<stop>**	**<activateOnEvent>**	**<deactivateOnEvent>**	**Regelaktivität**
<whenLoaded>	<dateTime>			Die Regel ist aktiv, wenn sie geladen wird, und wird an einem angegebenen Datum und zu einer angegebenen Zeit inaktiviert.
<inactiveWhenLoaded>	<never>	*X*		Wenn die Regel geladen wird, ist sie inaktiv. Das Ereignis *X* aktiviert die Regel, und die Regel bleibt aktiv, während die ACT-Engine ausgeführt wird.
<inactiveWhenLoaded>	<never>	*X*	*Y*	Wenn die Regel geladen wird, ist sie inaktiv. Ereignis *X* aktiviert die Regel, und Ereignis *Y* inaktiviert sie. Diese Aktivierung und Inaktivierung kann mehrfach ausgeführt werden.
<dateTime>	<dateTime>			Die Regel wird an einem angegebenen Datum und zu einer angegebenen Zeit aktiviert und an einem angegebenen Datum und zu einer angegebenen Zeit inaktiviert.
<dateTime>	<dateTime>	*X*	*Y*	Die Regel wird an einem angegebenen Datum und zu einer angegebenen Zeit aktiviert und an einem angegebenen Datum und zu einer angegebenen Zeit inaktiviert. Ereignis *X* aktiviert die Regel, und Ereignis *Y* inaktiviert sie. Die Ereignisse *X* und *Y* können die Regel mehrmals aktivieren bzw. inaktivieren.
<dateTime>	<never>			Die Regel wird an einem angegebenen Datum und zu einer angegebenen Zeit aktiviert und bleibt aktiv, während die ACT-Engine ausgeführt wird.
<dateTime>	<never>		*Y*	Die Regel wird an einem angegebenen Datum und zu einer angegebenen Zeit aktiviert. Ereignis *Y* inaktiviert die Regel.
<dateTime>	<never>	*X*	*Y*	Die Regel wird an einem angegebenen Datum und zu einer angegebenen Zeit aktiviert. Ereignis *Y* inaktiviert die Regel, und Ereignis *X* reaktiviert sie. Diese Inaktivierung und Reaktivierung kann mehrfach ausgeführt werden.
<dateTime>	<after>			Die Regel wird an einem angegebenen Datum und zu einer angegebenen Zeit aktiviert und nach einem angegebenen Zeitintervall inaktiviert.
<dateTime>	<after>	*X*	*Y*	Die Regel wird an einem angegebenen Datum und zu einer angegebenen Zeit aktiviert und nach einem angegebenen Zeitintervall inaktiviert. Ereignis *X* aktiviert die Regel, und Ereignis *Y* inaktiviert sie. Diese Aktivierung und Inaktivierung kann mehrfach ausgeführt werden.

Element 'activationTime'

Das Element <activationTime> definiert diskrete Zeitpunkte, wann eine Regel aktiviert oder inaktiviert wird.

Attribute

<activationTime> weist keine Attribute auf.

Enthalten in

<activationTime> ist im folgenden Element enthalten:

- <activationInterval>

Enthält

<activationTime> enthält die folgenden Elemente.

Die Elemente müssen in der angezeigten Reihenfolge codiert werden. Wenn ein Element optional ist, muss es nicht codiert werden. Alle codierten Elemente müssen jedoch die richtige Reihenfolge aufweisen.

Tabelle 19. Im Element <activationTime> enthaltene Elemente

Element	Erforderlich oder optional?
<start>	Optional. 0 oder 1 Vorkommen ist zulässig.
<stop>	Optional. 0 oder 1 Vorkommen ist zulässig.

Element 'after'

Das Element <after> gibt die Zeitdauer an, für die die Regel aktiv bleiben soll, nachdem sie aktiv wurde. Nach dieser Zeitdauer soll die Regel inaktiviert werden.

Attribute

<after> weist die folgenden Attribute auf:

Tabelle 20. Attribute des Elements <after>

Name	Beschreibung	Datentyp	Erforderlich?
duration	Gibt die Zeitdauer an. Der Datentyp dieses Attributs hängt vom Wert des Attributs für die Einheit ab.	• Wenn der Wert des Attributs für die Einheit `ISO-8601` lautet, lautet der Datentyp 'xsd:duration'. • Wenn der Wert des Attributs für die Einheit `milliseconds` lautet, lautet der Datentyp 'xsd:positiveInteger'.	Ja
unit	Gibt die zu verwendende Zeiteinheit an. Für dieses Attribut sind folgende Werte gültig: • `ISO-8601` • `milliseconds`	xsd:string	Ja

Verwendung des ISO-Standards 8601 für die Dauer

Die Codierung von `ISO-8601` als Wert für das Attribut für die Einheit bedeutet, dass der Wert des Attributs für die Dauer entsprechend des ISO-Standards 8601 für die Angabe einer Dauer als Zeichenfolge codiert wird. Die Standard-XML-Schemadatentypspezifikation verwendet ISO 8601, um einen Datentyp mit dem Namen `duration` bereitzustellen. Dieser Datentyp wird unter http://www.w3.org/TR/xmlschema-2/#duration genauer beschrieben.

Das Format für den Datentyp `duration` im Standard-XML-Schema ist die folgende Zeichenfolge:

`PnYnMnDTnHnMnS`

- `P` ist das Zeichen, mit dem die Zeichenfolge immer beginnt.
- *n*`Y` stellt die Anzahl Jahre dar. Ein Jahr entspricht 365 Tagen. Daher entspricht die Codierung `1Y` der Codierung `365D`.

- *n*M stellt die Anzahl Monate dar. Ein Monat entspricht 30 Tagen. Daher entspricht die Codierung 1M der Codierung 30D.
- *n*D stellt die Anzahl Tage dar.
- T ist ein Trennzeichen, das Tageseinheiten (Jahre, Monate und Tage) von Zeiteinheiten (Stunden, Minuten und Sekunden) trennt. Zeiteinheiten folgen immer auf T.
- *n*H stellt die Anzahl Stunden dar.
- *n*M stellt die Anzahl Minuten dar.
- *n*S stellt die Anzahl Sekunden dar.

Im Folgenden finden Sie Beispiele zum Format:

- P5DT12H entspricht 5,5 Tagen.
- PT59M59S entspricht 59 Minuten und 59 Sekunden.
- P1M entspricht 1 Monat.

Enthalten in

<after> ist im folgenden Element enthalten:

- <stop>

Enthält

<after> enthält keine Elemente.

Element 'attributeAlias'

Das Element <attributeAlias> stellt einen Aliasnamen bereit, der Ereignisattribute verknüpft, die dieselbe Bedeutung, jedoch unterschiedliche Namen in unterschiedlichen Ereignissen haben. Drei verschiedene Ereignisse können beispielsweise die folgenden drei unterschiedlichen Namen für ein Ereignisattribut verwenden, das den Namen des Systems angibt, von dem das Ereignis stammt: Host, Hostname und Quelle. Das Element <attributeAlias> enthält die <eventAttribute>-Elemente, die die einzelnen Ereignisattribute beschreiben, die als ein Ereignisattribut für den Gruppierungsschlüssel verknüpft werden müssen.

Details

Das Element <attributeAlias> und sein aliasName-Attribut sind nur im Kontext eines Gruppierungsschlüssels gültig. Auf dieses Element und sein Attribut kann in keinem Ausdruck verwiesen werden, einschließlich eines Ausdrucks im Element <computedValue>.

Attribute

<attributeAlias> weist das folgende Attribut auf:

Tabelle 21. Attribute des Elements <attributeAlias>

Name	Beschreibung	Datentyp	Erforderlich?
aliasName	Definiert den Namen für die Ereignisattribute, die in den <eventAttribute>-Elementen beschrieben sind und die als ein Ereignisattribut für den Gruppierungsschlüssel verknüpft werden sollen. Dieser Name muss innerhalb der Regel eindeutig sein.	xsd:NMTOKEN	Ja

Enthalten in

<attributeAlias> ist im folgenden Element enthalten:

- <groupingKey>

Enthält

<attributeAlias> enthält das folgende Element:

Tabelle 22. Im Element <attributeAlias> enthaltene Elemente

Element	Erforderlich oder optional?
<eventAttribute>	2 Vorkommen dieses Elements sind erforderlich. Zusätzliche Vorkommen sind zulässig.

Element 'attributeName'

Das Element <attributeName> enthält den Namen eines bestimmten Ereignisattributs, das Teil des Gruppierungsschlüssels ist. Dieser Name muss dem Namen entsprechen, der in dem Methodenaufruf 'getAttribute' für die Variable 'act_event' verwendet wird.

Attribute

<attributeName> weist keine Attribute auf.

Enthalten in

<attributeName> ist im folgenden Element enthalten:

- <groupingKey>

Enthält

<attributeName> enthält keine Elemente.

Element 'booleanThreshold'

Das Element <booleanThreshold> ist nur für die Schwellenwertregel gültig. Es enthält einen Ausdruck, der aufgerufen wird, wenn die einzelnen Ereignisse empfangen werden. Der Ausdruck berechnet oder vergleicht den Schwellenwert auf der Basis des aktuellen Ereignisses und aller übrigen Ereignisse, die von der Regel akzeptiert wurden. Der Ausdruck gibt den Booleschen Wert `wahr` oder `falsch` zurück, um anzugeben, ob der Schwellenwert erreicht wurde.

Details

Informationen zu den Variablen, die in Ausdrücken verwendet werden können, finden Sie in „Variablen" auf Seite 25. Die Verwendung bestimmter Variablen hängt vom Kontext des Ausdrucks ab.

Attribute

<booleanThreshold> weist das folgende Attribut auf:

Tabelle 23. Attribute des Elements <booleanThreshold>

Name	Beschreibung	Datentyp	Erforderlich?
expressionLanguage	Gibt die Programmiersprache an, in der der Ausdruck geschrieben wurde. Da die Programmiersprache Java die einzige unterstützte Sprache für Ausdrücke ist, ist nur der Wert `java` für dieses Attribut gültig.	xsd:NMTOKEN	Ja

Enthalten in

<booleanThreshold> ist im folgenden Element enthalten:

- <thresholdRule>

Enthält

<booleanThreshold> enthält keine Elemente.

Zugehörige Konzepte

„Ausdrücke" auf Seite 20
Ein Ausdruck ist Code, der eine angepasste Logik enthält, die einer Regel hinzugefügt werden kann. Ausdrücke können auch auf Code außerhalb der ACT-Engine zugreifen. In der Regelsprache sind Ausdrücke nur in bestimmten Kontexten, oder Regelsprachelementen, gültig.

Element 'collectionRule'

Das Element <collectionRule> definiert eine Regel entsprechend des Erfassungsmusters.

Attribute

<collectionRule> weist die folgenden Attribute auf:

Tabelle 24. Attribute des Elements <collectionRule>

Name	Beschreibung	Datentyp	Erforderlich?
name	Kennzeichnet die Regel. Dieser Bezeichner muss innerhalb des Regelblocks, der diese Regel enthält, eindeutig sein. Er darf keinen Punkt enthalten.	xsd:NMTOKEN	Ja
processOnlyForwardedEvents	Bestimmt, ob die Regel alle Ereignisse oder nur die Ereignisse empfängt, die von anderen Regeln weitergeleitet wurden. Der Standardwert lautet `false`, was bedeutet, dass die Regel alle Ereignisse empfängt, einschließlich der Ereignisse, die von anderen Regeln weitergeleitet wurden.	xsd:boolean	Nein

Enthalten in

<collectionRule> ist im folgenden Element enthalten:

- <ruleBlock>

Enthält

<collectionRule> enthält die folgenden Elemente.

Die Elemente müssen in der angezeigten Reihenfolge codiert werden. Wenn ein Element optional ist, muss es nicht codiert werden. Alle codierten Elemente müssen jedoch die richtige Reihenfolge aufweisen.

Tabelle 25. Im Element <collectionRule> enthaltene Elemente

Element	Erforderlich oder optional?
<comment>	Optional. 0 oder 1 Vorkommen ist zulässig.
<variable>	Optional. 0 oder mehr Vorkommen sind zulässig.
<activationInterval>	Optional. 0 oder 1 Vorkommen ist zulässig.
<lifeCycleActions>	Optional. 0 oder 1 Vorkommen ist zulässig.
<eventSelector>	Optional. 0 oder 1 Vorkommen ist zulässig.
<groupingKey>	Optional. 0 oder 1 Vorkommen ist zulässig.
<timeWindow>	Erforderlich. Nur 1 Vorkommen ist zulässig.
<onTimeWindowComplete>	Optional. 0 oder 1 Vorkommen ist zulässig.

Zugehörige Konzepte

„Erfassungsmuster" auf Seite 11
Eine Datenerfassungsregel wird durch das Erfassungsmuster definiert. Sie erfasst eine Gruppe von ausgewählten Ereignissen innerhalb eines Zeitintervalls. Sie ist eine statusabhängige Regel.

Element 'comment'

Das Element <comment> kann eine Beschreibung der Funktion und des Zwecks seines übergeordneten Regelsatzes bzw. Regelblocks sowie seiner übergeordneten Regel oder Variablen enthalten.

Attribute

<comment> weist keine Attribute auf.

Enthalten in

<comment> ist in den folgenden Elementen enthalten:

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

Enthält

<comment> enthält keine Elemente.

Element 'computationRule'

Das Element <computationRule> definiert eine Regel entsprechend des Berechnungsmusters.

Attribute

<computationRule> weist die folgenden Attribute auf:

Tabelle 26. Attribute des Elements <computationRule>

Name	Beschreibung	Datentyp	Erforderlich?
name	Kennzeichnet die Regel. Dieser Bezeichner muss innerhalb des Regelblocks, der diese Regel enthält, eindeutig sein. Er darf keinen Punkt enthalten.	xsd:NMTOKEN	Ja

Tabelle 26. Attribute des Elements <computationRule> (Forts.)

Name	Beschreibung	Datentyp	Erforderlich?
processOnlyForwardedEvents	Bestimmt, ob die Regel alle Ereignisse oder nur die Ereignisse empfängt, die von anderen Regeln weitergeleitet wurden. Der Standardwert lautet `false`, was bedeutet, dass die Regel alle Ereignisse empfängt, einschließlich der Ereignisse, die von anderen Regeln weitergeleitet wurden.	xsd:boolean	Nein

Enthalten in

<computationRule> ist im folgenden Element enthalten:

- <ruleBlock>

Enthält

<computationRule> enthält die folgenden Elemente.

Die Elemente müssen in der angezeigten Reihenfolge codiert werden. Wenn ein Element optional ist, muss es nicht codiert werden. Alle codierten Elemente müssen jedoch die richtige Reihenfolge aufweisen.

Tabelle 27. Im Element <computationRule> enthaltene Elemente

Element	Erforderlich oder optional?
<comment>	Optional. 0 oder 1 Vorkommen ist zulässig.
<variable>	Optional. 0 oder mehr Vorkommen sind zulässig.
<activationInterval>	Optional. 0 oder 1 Vorkommen ist zulässig.
<lifeCycleActions>	Optional. 0 oder 1 Vorkommen ist zulässig.
<eventSelector>	Optional. 0 oder 1 Vorkommen ist zulässig.
<groupingKey>	Optional. 0 oder 1 Vorkommen ist zulässig.
<computeFunction>	Erforderlich. Nur 1 Vorkommen ist zulässig.
<timeWindow>	Erforderlich. Nur 1 Vorkommen ist zulässig.
<onTimeWindowComplete>	Optional. 0 oder 1 Vorkommen ist zulässig.

Zugehörige Konzepte

„Berechnungsmuster" auf Seite 12
Eine Berechnungsregel wird durch das Berechnungsmuster definiert. Es wendet eine Berechnung (durch einen Ausdruck) auf erfasste Ereignisse an, während die einzelnen Ereignisse innerhalb eines Zeitintervalls empfangen werden. Es ist eine statusabhängige Regel.

Element 'computedThreshold'

Das Element <computedThreshold> ist nur für die Schwellenwertregel gültig. Es enthält einen Ausdruck, der aufgerufen wird, wenn die einzelnen Ereignisse empfangen werden, und der den Schwellenwert auf der Grundlage des aktuellen Ereignisses und aller übrigen Ereignisse berechnet, die die Ereignisauswahlkriterien

für die Regel erfüllten. Der Ausdruck gibt den berechneten Schwellenwert zurück, damit dieser in einer für die Regel definierten Variablen gespeichert wird. Die Regel vergleicht dann den berechneten Schwellenwert mit dem definierten Schwellenwert.

Details

Informationen zu den Variablen, die in Ausdrücken verwendet werden können, finden Sie in „Variablen" auf Seite 25. Die Verwendung bestimmter Variablen hängt vom Kontext des Ausdrucks ab.

Attribute

<computedThreshold> weist die folgenden Attribute auf:

Tabelle 28. Attribute des Elements <computedThreshold>

Name	Beschreibung	Datentyp	Erforderlich?
expressionLanguage	Gibt die Programmiersprache an, in der der Ausdruck geschrieben wurde. Da die Programmiersprache Java die einzige unterstützte Sprache für Ausdrücke ist, ist nur der Wert `java` für dieses Attribut gültig.	xsd:NMTOKEN	Ja
threshold	Definiert den zu erreichenden Schwellenwert. Dieser definierte Schwellenwert muss eine Zeichenfolgedarstellung eines numerischen Wertes sein, die in einen Datentyp umgewandelt werden kann, der für die Regelvariable gültig ist.	xsd:string	Ja
assignTo	Gibt den Namen der Variablen an, die den berechneten Schwellenwert enthält, der von diesem Ausdruck zurückgegeben wird. Diese Variable muss bereits mit dem Element <variable> (auf der Ebene des Regelsatzes, des Regelblocks oder der Regel) für die Regel definiert sein. Sie muss als einer der folgenden numerischen Datentypen definiert sein: • java.lang.Double • java.lang.Float • java.lang.Integer • java.lang.Long • java.lang.String Wenn die Variable auf der Ebene des Regelsatzes oder Regelblocks definiert ist, wird sie nicht reinitialisiert, nachdem das Regelmuster abgeglichen wurde.	xsd:NMTOKEN	Ja

Tabelle 28. Attribute des Elements <computedThreshold> (Forts.)

Name	Beschreibung	Datentyp	Erforderlich?
thresholdComparison	Definiert den Operator für den Vergleich des berechneten Schwellenwerts mit dem definierten Schwellenwert. Die gültigen Werte für diesen Operator lauten wie folgt: • `lessThan` • `lessThanOrEqualTo` • `greaterThan` • `greaterThanOrEqualTo`	xsd:string	Ja

Enthalten in

<computedThreshold> ist im folgenden Element enthalten:

- <thresholdRule>

Enthält

<computedThreshold> enthält keine Elemente.

Zugehörige Konzepte

„Ausdrücke" auf Seite 20
Ein Ausdruck ist Code, der eine angepasste Logik enthält, die einer Regel hinzugefügt werden kann. Ausdrücke können auch auf Code außerhalb der ACT-Engine zugreifen. In der Regelsprache sind Ausdrücke nur in bestimmten Kontexten, oder Regelsprachelementen, gültig.

Element 'computedValue'

Das Element <computedValue> enthält einen Ausdruck, der ausgeführt wird, wenn die Regel ein Ereignis empfängt, um einen Zeichenfolgewert zu erstellen, der auf dem Wert mindestens eines Attributs des Ereignisses basiert. Dieser Zeichenfolgewert kann anschließend im Gruppierungsschlüssel verwendet werden.

Details

Es kommt vor, dass der Autor einer Regel im Gruppierungsschlüssel Einträge wie die folgenden verwenden möchte:

- Eine Unterzeichenfolge eines Ereignisattributwerts. Beispiel: Wenn ein Ereignisattributwert eine eingebettete IP-Adresse enthält, könnte der Ausdruck im Element <computedValue> diese IP-Adresse als eindeutigen Wert für die Verwendung im Gruppierungsschlüssel extrahieren.
- Unterzeichenfolgen der Werte mehrerer verschiedener Ereignisattribute. Beispiel: Der Ausdruck im Element <computedValue> könnte die Unterzeichenfolgen extrahieren und sie kombinieren, um einen eindeutigen Wert für die Verwendung im Gruppierungsschlüssel zu erstellen.

Wenn der Ausdruck im Element <computedValue> einen Nullwert zurückgibt, behandelt die Regel diesen Nullwert als fehlenden Attributwert.

Informationen zu den Variablen, die in Ausdrücken verwendet werden können, finden Sie in „Variablen" auf Seite 25. Die Verwendung bestimmter Variablen hängt vom Kontext des Ausdrucks ab.

Attribute

<computedValue> weist das folgende Attribut auf:

Tabelle 29. Attribute des Elements <computedValue>

Name	Beschreibung	Datentyp	Erforderlich?
expressionLanguage	Gibt die Programmiersprache an, in der der Ausdruck geschrieben wurde. Da die Programmiersprache Java die einzige unterstützte Sprache für Ausdrücke ist, ist nur der Wert `java` für dieses Attribut gültig.	xsd:NMTOKEN	Ja

Enthalten in

<computedValue> ist im folgenden Element enthalten:

- <groupingKey>

Enthält

<computedValue> enthält keine Elemente.

Zugehörige Konzepte

„Ausdrücke" auf Seite 20
Ein Ausdruck ist Code, der eine angepasste Logik enthält, die einer Regel hinzugefügt werden kann. Ausdrücke können auch auf Code außerhalb der ACT-Engine zugreifen. In der Regelsprache sind Ausdrücke nur in bestimmten Kontexten, oder Regelsprachelementen, gültig.

Element 'computeFunction'

Das Element <computeFunction> ist nur für die Berechnungsregel gültig. Es enthält einen Ausdruck, der aufgerufen wird, wenn die einzelnen Ereignisse empfangen werden, und der einen Wert zurückgibt, der in einer für die Regel definierten Variablen gespeichert wird. Der von diesem Ausdruck zurückgegebene Wert muss dem Datentyp der Variablen entsprechen, die im Attribut 'assignTo' des Elements <computeFunction> angegeben ist.

Details

Informationen zu den Variablen, die in Ausdrücken verwendet werden können, finden Sie in „Variablen" auf Seite 25. Die Verwendung bestimmter Variablen hängt vom Kontext des Ausdrucks ab.

Attribute

<computeFunction> weist die folgenden Attribute auf:

Tabelle 30. Attribute des Elements <computeFunction>

Name	Beschreibung	Datentyp	Erforderlich?
expressionLanguage	Gibt die Programmiersprache an, in der der Ausdruck geschrieben wurde. Da die Programmiersprache Java die einzige unterstützte Sprache für Ausdrücke ist, ist nur der Wert `java` für dieses Attribut gültig.	xsd:NMTOKEN	Ja
assignTo	Gibt den Namen der Variablen an, die den Wert enthält, der von diesem Ausdruck zurückgegeben wird. Diese Variable muss bereits mit dem Element <variable> (auf der Ebene des Regelsatzes, des Regelblocks oder der Regel) für die Regel definiert sein. Wenn die Variable auf der Ebene des Regelsatzes oder Regelblocks definiert ist, wird sie nicht reinitialisiert, nachdem das Regelmuster abgeglichen wurde.	xsd:NMTOKEN	Ja

Enthalten in

<computeFunction> ist im folgenden Element enthalten:

- <computationRule>

Enthält

<computeFunction> enthält keine Elemente.

Zugehörige Konzepte

„Ausdrücke” auf Seite 20
Ein Ausdruck ist Code, der eine angepasste Logik enthält, die einer Regel hinzugefügt werden kann. Ausdrücke können auch auf Code außerhalb der ACT-Engine zugreifen. In der Regelsprache sind Ausdrücke nur in bestimmten Kontexten, oder Regelsprachelementen, gültig.

Element ’dateTime’

Das Element <dateTime> gibt das Datum und die Zeit an, wann eine Regel aktiviert oder inaktiviert wird. Die Regel wird jedoch nur aktiviert oder inaktiviert, wenn die Regel vor diesem angegebenen Zeitpunkt in eine aktive ACT-Engine geladen wurde.

Details

Wenn die Regel vor dem angegebenen Zeitpunkt für die Aktivierung nicht in eine aktive ACT-Engine geladen wurde, wird die Regel niemals aktiviert. Wenn die Regel vor dem angegebenen Zeitpunkt für die Inaktivierung nicht in eine aktive ACT-Engine geladen wurde, wird die Regel in den Status gesetzt, der durch das Element <start> definiert ist, und die Regel wird niemals durch das Element <stop> inaktiviert.

Der Inhalt des Elements <dateTime> muss eine Zeichenfolge sein, die das Format für den Datentyp dateTime im Standard-XML-Schema aufweist. dateTime besteht beispielsweise aus Zeichenfolgen mit begrenzter Länge im folgenden Format:

jjjj '-' *mm* '-' *tt* 'T' *hh* ':' *mm* ':' *ss* ('.' *s*+)? (*zzzzzz*)?

- *jjjj* ist ein Numeral bestehend aus mindestens vier Stellen und stellt das Jahr dar. Wenn das Numeral aus mehr als vier Stellen besteht, sind führende Nullen untersagt, und 0000 ist ebenfalls untersagt.
- Die verbleibenden Bindestriche ('-') sind Trennzeichen zwischen Teilen des Datums.
- Das erste Vorkommen von *mm* ist ein zweistelliges Numeral, das den Monat beginnend mit 01 darstellt.
- *tt* ist ein zweistelliges Numeral, das den Tag des Monats beginnend mit 01 darstellt.
- T ist ein Trennzeichen, das angibt, dass die Uhrzeit folgt.
- *hh* ist ein zweistelliges Numeral, das die Stunde im 24-Stunden-System darstellt, beginnend mit 00 und endend mit 23.
- : ist ein Trennzeichen zwischen Teilen der Uhrzeit.
- Das zweite Vorkommen von *mm* ist ein zweistelliges Numeral, das die Minute beginnend mit 00 und endend mit 59 darstellt.
- *ss* ist ein zweistelliges Numeral, das ganze Sekunden beginnend mit 00 und endend mit 59 darstellt.
- '.' *s*+, falls vorhanden, stellt den Bruchteil von Sekunden dar.
- *zzzzzz*, falls vorhanden, stellt die Zeitzone dar. Die Zeitzone besteht aus Zeichenfolgen mit begrenzter Länge im Format (('+' | '-') *hh* ':' *mm*) | 'Z', wobei Folgendes gilt:
  - '+', falls vorhanden, stellt eine nicht negative Dauer dar, und '-' darf nicht vorhanden sein.
  - '-', falls vorhanden, stellt eine nicht positive Dauer dar, und '+' darf nicht vorhanden sein.
  - *hh* ist ein zweistelliges Numeral, das die Stunden beginnend mit 00 und endend mit 14 darstellt.
  - *mm* ist ein zweistelliges Numeral, das die Minuten beginnend mit 00 und endend mit 59 darstellt. Wenn der Stundenwert jedoch 14 beträgt, muss der Minutenwert 00 lauten.
  - Z ist die Kurzschreibweise für UTC (Universal Time Coordinated - Koordinierte Weltzeit) (+00:00 oder -00:00), und somit muss kein weiteres Zeitzonenelement vorhanden sein.

Im Folgenden finden Sie zwei Beispiele für den Inhalt des Elements <dateTime>:

- 2005-06-01T13:05:06.07 bedeutet 6 Sekunden und 7 Hundertstelsekunden nach 13:05 Uhr Ortszeit am 1. Juni 2005.
- 2005-06-01T13:05:06.07Z bedeutet 6 Sekunden und 7 Hundertstelsekunden nach 13:05 Uhr UTC-Zeit 1. Juni 2005, und somit 6 Sekunden und 7 Hundertstelsekunden nach 9:05 Uhr EDT (Eastern Daylight Saving Time - Ost-Sommerzeit) 1. Juni 2005 (oder 2005-06-01T09:05:06.07-04:00).

Attribute

<dateTime> weist keine Attribute auf.

Enthalten in

<dateTime> ist in den folgenden Elementen enthalten:

- <start>
- <stop>

Enthält

<dateTime> enthält keine Elemente.

Element 'deactivateOnEvent'

Das Element <deactivateOnEvent> definiert die Ereignisse, die die Regel (oder für Regeln, die mit dem Element <groupingKey> definiert sind, eine Regelinstanz) inaktivieren können.

Im Folgenden werden die drei Möglichkeiten aufgeführt, Ereignisse auszuwählen:

- Die Verwendung mindestens eines <eventType>-Elements mit einem <filteringPredicate>-Element
- Die Verwendung mindestens eines <eventType>-Elements ohne <filteringPredicate>-Element
- Die Verwendung eines <filteringPredicate>-Elements ohne <eventType>-Elemente

Wenn die Regel aktiv ist und kein <eventType>- oder <filteringPredicate>-Element codiert ist, wird jedes Ereignis ausgewählt, das auftritt.

Wenn Sie keine <eventType>-Elemente codieren, kann dies negative Auswirkungen auf die Systemleistung haben.

Angenommen, Sie möchten alle Ereignisse des Typs `Audit Failure` auswählen. Sie können ein Filterprädikat verwenden, um die Auswahlkriterien weiter einzugrenzen, damit nur die Ereignisse eingeschlossen werden, die ein Ereignisattribut mit einem bestimmten Wert aufweisen. Beispiel: Sie codieren ein <eventType>-Element, um alle Ereignisse des Typs `Audit Failure` auszuwählen und ein <filteringPredicate>-Element, um nur die Ereignisse auszuwählen, die ein Hostnamensattribut mit dem Wert `MyCriticalSystem` aufweisen.

Attribute

<deactivateOnEvent> weist keine Attribute auf.

Enthalten in

<deactivateOnEvent> ist in den folgenden Elementen enthalten:

- <activationInterval>
- <activationByGroupingKey>

Enthält

<deactivateOnEvent> enthält die folgenden Elemente.

Die Elemente müssen in der angezeigten Reihenfolge codiert werden. Wenn ein Element optional ist, muss es nicht codiert werden. Alle codierten Elemente

müssen jedoch die richtige Reihenfolge aufweisen.

Tabelle 31. Im Element <deactivateOnEvent> enthaltene Elemente

Element	Erforderlich oder optional?
<eventType>	Optional. 0 oder mehr Vorkommen sind zulässig.
<filteringPredicate>	Optional. 0 oder 1 Vorkommen ist zulässig.

Element 'duplicateRule'

Das Element <duplicateRule> definiert eine Regel entsprechend dem Duplikatmuster.

Attribute

<duplicateRule> weist die folgenden Attribute auf:

Tabelle 32. Attribute des Elements <duplicateRule>

Name	Beschreibung	Datentyp	Erforderlich?
name	Kennzeichnet die Regel. Dieser Bezeichner muss innerhalb des Regelblocks, der diese Regel enthält, eindeutig sein. Er darf keinen Punkt enthalten.	xsd:NMTOKEN	Ja
processOnlyForwardedEvents	Bestimmt, ob die Regel alle Ereignisse oder nur die Ereignisse empfängt, die von anderen Regeln weitergeleitet wurden. Der Standardwert lautet `false`, was bedeutet, dass die Regel alle Ereignisse empfängt, einschließlich der Ereignisse, die von anderen Regeln weitergeleitet wurden.	xsd:boolean	Nein

Enthalten in

<duplicateRule> ist im folgenden Element enthalten:

- <ruleBlock>

Enthält

<duplicateRule> enthält die folgenden Elemente.

Die Elemente müssen in der angezeigten Reihenfolge codiert werden. Wenn ein Element optional ist, muss es nicht codiert werden. Alle codierten Elemente müssen jedoch die richtige Reihenfolge aufweisen.

Tabelle 33. Im Element <duplicateRule> enthaltene Elemente

Element	Erforderlich oder optional?
<comment>	Optional. 0 oder 1 Vorkommen ist zulässig.
<variable>	Optional. 0 oder mehr Vorkommen sind zulässig.
<activationInterval>	Optional. 0 oder 1 Vorkommen ist zulässig.

Tabelle 33. Im Element <duplicateRule> enthaltene Elemente (Forts.)

Element	Erforderlich oder optional?
<lifeCycleActions>	Optional. 0 oder 1 Vorkommen ist zulässig.
<eventSelector>	Optional. 0 oder 1 Vorkommen ist zulässig.
<groupingKey>	Optional. 0 oder 1 Vorkommen ist zulässig.
<timeWindow>	Erforderlich. Nur 1 Vorkommen ist zulässig.
<onDetection>	Optional. 0 oder 1 Vorkommen ist zulässig.
<onNextEvent>	Optional. 0 oder 1 Vorkommen ist zulässig.
<onTimeWindowComplete>	Optional. 0 oder 1 Vorkommen ist zulässig.

Zugehörige Konzepte

„Duplikatmuster" auf Seite 12
Eine Duplikatregel wird durch das Duplikatmuster definiert. Es zählt das zweite Ereignis sowie nachfolgende Ereignisse, die innerhalb eines angegebenen Zeitintervalls akzeptiert werden, überspringt jedoch die Regelsatzverarbeitung für diese Ereignisse. Es ist eine statusabhängige Regel.

Element 'eventAttribute'

Das Element <eventAttribute> bietet eine Möglichkeit, einen Ereignistyp und ein Ereignisattribut als Teil des Attributaliasnamens zu verknüpfen, der durch das Element <attributeAlias> definiert ist.

Attribute

<eventAttribute> weist die folgenden Attribute auf:

Tabelle 34. Attribute des Elements <eventAttribute>

Name	Beschreibung	Datentyp	Erforderlich?
type	Definiert den Namen des Ereignistyps. Dies ist derselbe Name, der für das Attribut 'type' im Element <eventType> verwendet wird.	xsd:NMTOKEN	Ja
attributeName	Gibt den vollständig qualifizierten Namen des Ereignisattributs an, das mit anderen Ereignisattributen durch den Attributaliasnamen verknüpft wird. Dieser Name muss dem Namen entsprechen, der in der Variablen 'act_event' für den Aufruf der Methode 'getAttribute' verwendet wird.	xsd:string	Ja

Enthalten in

<eventAttribute> ist im folgenden Element enthalten:

- <attributeAlias>

Enthält

<eventAttribute> enthält keine Elemente.

Element 'eventCountThreshold'

Das Element <eventCountThreshold> gilt nur für die Schwellenwertregel. Es definiert die Anzahl Ereignisse, die die Ereignisauswahlkriterien in einem bestimmten Zeitraum erfüllen müssen. Das Element <eventCountThreshold> gibt zudem einen von zwei möglichen Zeitintervallmodi (festgelegtes Intervall oder Schiebeintervall) für das Zeitfenster an.

Details

Festgelegtes Intervall

Ein festgelegtes Intervall beginnt, wenn das erste Ereignis empfangen wird, das die Ereignisauswahlkriterien erfüllt, und endet, wenn eine der folgenden Aussagen eintritt:

- Der Schwellenwert der Regel wird im angegebenen Zeitraum erreicht.
- Die angegebene Dauer ist abgelaufen.

Schiebeintervall

Ein Schiebeintervall beginnt, wenn das erste Ereignis empfangen wird, das die Ereignisauswahlkriterien erfüllt. Wenn die Regel ihren Schwellenwert jedoch nicht erreicht hat und die angegebene Dauer abgelaufen ist, passt das Zeitfenster die Anfangszeit an die Ereignisempfangszeit für ein neues „erstes" Ereignis an, das normalerweise das nächste Ereignis ist, das akzeptiert wird. Die Anfangszeit wird somit geschoben. Das Schiebeintervall fährt mit der Anpassung auf diese Weise fort, bis eine der folgenden Situationen eintritt:

- Der Schwellenwert der Regel wird im angegebenen Zeitraum erreicht.
- Nachdem das Ereignis empfangen wird, das das Zeitfenster startet, werden innerhalb der angegebenen Dauer keine folgenden Ereignisse empfangen.

Das Ereignis, das das Zeitfenster startet (das neue „erste" Ereignis), ist das Ereignis mit einer Empfangszeit, die die folgende Bedingung erfüllt: Die Empfangszeit, in Addition mit der Zeitintervalldauer für die Regel, ist größer als die aktuelle Uhrzeit. Die Bedingung kann auch in Form einer Gleichung ausgedrückt werden:

```
ereignisempfangszeit + zeitintervalldauer für regel > aktuelle uhrzeit
```

Wenn ein solches Ereignis nicht vorhanden ist, kann das Schiebeintervall die Zeit nicht weiter nach vorne schieben, und das Intervall endet.

Die Schwellenwertregel zählt jedes akzeptierte Ereignis, bis der Schwellenwert erreicht ist oder der Zeitraum endet. Anschließend werden die Aktionen ausgeführt, die im Element <onDetection> bzw. im Element <onTimeOut> definiert sind.

<onDetection>-Aktionen

Diese Aktionen werden ausgeführt, wenn die Zahl der Ereignisse dem Wert entspricht, der durch das Schwellenwertattribut des Elements <eventCountThreshold> definiert ist, was bedeutet, dass der Schwellenwert erreicht ist.

<onTimeOut>-Aktionen

Wann diese Aktionen ausgeführt werden, hängt davon ab, ob der Zeitintervallmodus 'fixed' oder 'sliding' lautet.

Modus 'fixed'
Im Modus 'fixed' werden diese Aktionen ausgeführt, wenn das Zeitfenster beendet ist.

Modus 'sliding'
Im Modus 'sliding' werden diese Aktionen ausgeführt, wenn nach Empfang des Ereignisses, das das Zeitfenster startet, innerhalb der angegebenen Dauer keine folgenden Ereignisse empfangen werden. Anders gesagt, es wird kein Ereignis mit einer Empfangszeit empfangen, die größer als die aktuelle Uhrzeit ist, wenn die Empfangszeit zur Zeitintervalldauer für die Regel hinzuaddiert wird.

Der Zeitintervallmodus für das Zeitfenster wird durch das Attribut 'timeIntervalMode' des Elements <eventCountThreshold> definiert. Das folgende Szenario veranschaulicht das Verhalten der beiden möglichen Zeitintervallmodi und deren Unterschiede.

Szenario zur Veranschaulichung der Modi 'fixed' und 'sliding'

Angenommen, die Regel empfängt vier Ereignisse, die die Ereignisauswahlkriterien erfüllen; jeweils ein Ereignis zu den folgenden Uhrzeiten: 8:00 Uhr, 8:04 Uhr, 8:06 Uhr und 8:07 Uhr. Der Schwellenwert des Ereigniszählers ist 3, und die Dauer des Zeitfensters ist 5 Minuten.

Regelverhalten im Modus `fixed`
Bei diesem Zeitintervallmodus beginnt die Schwellenwertregel um 8:00 Uhr mit der Verarbeitung. Sie führt die <onTimeOut>-Aktionen um 8:05 Uhr aus, da sie nur 2 Ereignisse in 5 Minuten empfängt. Daher wird ihr Schwellenwert innerhalb des Zeitfensters nicht erreicht. Wenn das dritte Ereignis um 8:06 Uhr empfangen wird, beginnt die Schwellenwertregel erneut mit der Verarbeitung und führt die <onTimeOut>-Aktionen um 8:11 Uhr aus, da sie nur 2 Ereignisse in 5 Minuten empfängt.

Der Modus 'fixed' ist statisch.

Regelverhalten im Modus `sliding`
Bei diesem Zeitintervallmodus beginnt die Schwellenwertregel um 8:00 Uhr mit der Verarbeitung. Um 8:05 Uhr (die terminierte Beendigung des Zeitfensters) stellt die Regel fest, dass sie nur zwei Ereignisse empfangen hat. Die Regel verwirft dann das Ereignis, das sie um 8:00 Uhr empfangen hat, und berechnet die Dauer mit dem Ende 8:09 Uhr erneut (weil das erste Ereignis nun das Ereignis ist, das sie um 8:04 Uhr empfangen hat). Wenn die Regel das Ereignis um 8:07 Uhr empfängt, führt sie die <onDetection>-Aktionen aus, da sie nun ihren Schwellenwert (3 Ereignisse um 8:04, 8:06 und 8:07 Uhr) innerhalb des letzten Zeitfensters (8:04 – 8:09) erreicht hat.

Der Modus 'sliding' ist in der Hinsicht dynamisch, dass er die Anpassung (das Schieben) der Anfangszeit fortführt, um den Schwellenwert innerhalb des Zeitfensters zu erreichen.

Setzen wir nun voraus, dass die Regel 4 Ereignisse empfängt, die die Ereignisauswahlkriterien erfüllen; jeweils ein Ereignis zu den folgenden Uhrzeiten: 8:00 Uhr, 8:04 Uhr, 8:06 Uhr und 8:10 Uhr. Der Schwellenwert des Ereigniszählers ist 3, und die Dauer des Zeitfensters ist 5 Minuten.

Regelverhalten im Modus `sliding`
In diesem Fall beginnt die Schwellenwertregel um 8:00 Uhr mit der

Verarbeitung. Um 8:05 Uhr (die terminierte Beendigung des Zeitfensters) stellt die Regel fest, dass sie nur zwei Ereignisse empfangen hat. Die Regel verwirft dann das Ereignis, das sie um 8:00 Uhr empfangen hat, und berechnet die Dauer mit dem Ende 8:09 Uhr erneut (weil das erste Ereignis nun das Ereignis ist, das sie um 8:04 Uhr empfangen hat).

Um 8:09 Uhr (die neu terminierte Beendigung des Zeitfensters) stellt die Regel fest, dass sie nur zwei Ereignisse empfangen hat. Die Regel verwirft dann das Ereignis, das sie um 8:04 Uhr empfangen hat, und berechnet die Dauer erneut, so dass sie um 8:11 Uhr endet (da das erste Ereignis nun das Ereignis ist, das sie um 8:06 Uhr empfangen hat).

Um 8:11 Uhr (die neu terminierte Beendigung des Zeitfensters) stellt die Regel fest, dass sie nur zwei Ereignisse empfangen hat. Die Regel verwirft dann das Ereignis, das sie um 8:06 Uhr empfangen hat, und berechnet die Dauer erneut, so dass sie um 8:15 Uhr endet (da das erste Ereignis nun das Ereignis ist, das sie um 8:10 Uhr empfangen hat).

Um 8:15 Uhr (die neu terminierte Beendigung des Zeitfensters) stellt die Regel fest, dass sie seit dem Ereignis um 8:10 Uhr, das das Zeitfenster gestartet hat, keine Ereignisse empfangen hat. Die Regel führt dann die <onTimeOut>-Aktionen aus.

Attribute

<eventCountThreshold> weist die folgenden Attribute auf:

Tabelle 35. Attribute des Elements <eventCountThreshold>

Name	Beschreibung	Datentyp	Erforderlich?
threshold	Definiert die Anzahl Ereignisse, die die Ereignisauswahlkriterien innerhalb eines bestimmten Zeitraums erfüllen müssen. Dies ist der Schwellenwert des Ereigniszählers, der erreicht werden soll. Dieser Wert muss eine positive Ganzzahl sein.	xsd:positiveInteger	Ja
timeIntervalMode	Definiert, ob das Zeitintervall für das Zeitfenster ein festgelegtes Intervall oder ein Schiebeintervall ist. Für dieses Attribut sind folgende Werte gültig: • `fixed` (der Standardwert) • `sliding`	xsd:string	Nein

Enthalten in

<eventCountThreshold> ist im folgenden Element enthalten:

- <thresholdRule>

Enthält

<eventCountThreshold> enthält keine Elemente.

Element 'eventSelector'

Das Element <eventSelector> definiert die Ereignisse, die für die Verarbeitung durch eine Regel ausgewählt werden.

Details

Im Folgenden werden die drei Möglichkeiten aufgeführt, Ereignisse auszuwählen:

- Die Verwendung mindestens eines <eventType>-Elements mit einem <filteringPredicate>-Element
- Die Verwendung mindestens eines <eventType>-Elements ohne <filteringPredicate>-Element
- Die Verwendung eines <filteringPredicate>-Elements ohne <eventType>-Elemente

In besonderen Fällen, in denen Sie möchten, dass eine Regel alle Ereignisse verarbeitet, haben Sie die folgenden Optionen:

- Codieren Sie kein <eventSelector>-Element.
- Codieren Sie ein <eventSelector>-Element, das keine Elemente enthält.

Wenn Sie keine <eventType>-Elemente codieren, kann dies negative Auswirkungen auf die Systemleistung haben.

Angenommen, Sie möchten alle Ereignisse des Typs `Audit Failure` auswählen. Sie können ein Filterprädikat verwenden, um die Auswahlkriterien weiter einzugrenzen, damit nur die Ereignisse eingeschlossen werden, die ein Ereignisattribut mit einem bestimmten Wert aufweisen. Beispiel: Sie codieren ein <eventType>-Element, um alle Ereignisse des Typs `Audit Failure` auszuwählen und ein <filteringPredicate>-Element, um nur die Ereignisse auszuwählen, die ein Hostnamensattribut mit dem Wert `MyCriticalSystem` aufweisen.

Attribute

<eventSelector> weist das folgende Attribut auf:

Tabelle 36. Attribute des Elements <eventSelector>

Name	Beschreibung	Datentyp	Erforderlich?
alias	Dieses Attribut ist nur in einer Sequenzregel gültig, da sie die einzige Regel ist, die mehrere <eventSelector>-Elemente hat. Dieses Attribut gibt einem Ereignis, das durch einen bestimmten Ereignisselektor in der Sequenzregel ausgewählt wird, einen eindeutigen Namen. Filterprädikate und Aktionen können diesen Aliasnamen dann verwenden, um auf dieses Ereignis zuzugreifen.	xsd:NMTOKEN	Nein

Enthalten in

<eventSelector> ist in den folgenden Elementen enthalten:

- <collectionRule>

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

Enthält

<eventSelector> enthält die folgenden Elemente.

Die Elemente müssen in der angezeigten Reihenfolge codiert werden. Wenn ein Element optional ist, muss es nicht codiert werden. Alle codierten Elemente müssen jedoch die richtige Reihenfolge aufweisen.

Tabelle 37. Im Element <eventSelector> enthaltene Elemente

Element	Erforderlich oder optional?
<eventType>	Optional. 0 oder mehr Vorkommen sind zulässig.
<filteringPredicate>	Optional. 0 oder 1 Vorkommen ist zulässig.

Element 'eventType'

Das Element <eventType> definiert den Typ des Ereignisses, das für die Verarbeitung durch eine Regel ausgewählt wird oder das die Regel aktiviert oder inaktiviert.

Attribute

<eventType> weist das folgende Attribut auf:

Tabelle 38. Attribute des Elements <eventType>

Name	Beschreibung	Datentyp	Erforderlich?
type	Definiert den Ereignistyp. Bei Ereignissen, die der Common Base Event-Spezifikation entsprechen, ist dieser Name der Wert des Attributs 'extensionName'. Für Ereignisse in IBM Tivoli Enterprise Console ist dieser Name der Ereignisklassenname, der in der BAROC-Datei definiert ist. Ereignisse, die auf anderen Formaten basieren, verwenden für die Angabe des Ereignistyps möglicherweise ein anderes Attribut.	xsd:NMTOKEN	Ja

Enthalten in

<eventType> ist in den folgenden Elementen enthalten:

- <activateOnEvent>
- <deactivateOnEvent>

- <eventSelector>

Enthält

<eventType> enthält keine Elemente.

Element 'filteringPredicate'

Das Element <filteringPredicate> enthält einen Ausdruck, der weiter einschränkt, welche Ereignisse für die Verarbeitung durch die Regel ausgewählt werden oder welche Ereignisse ausgewählt werden, um die Regel zu aktivieren oder zu inaktivieren. Auf diese Weise können Ereignisse noch umfassender gefiltert werden als lediglich mit Hilfe des Ereignistyps durch das Element <eventType>.

Details

Der Ausdruck definiert eine Bedingung und gibt einen Booleschen Wert zurück, entweder `true` (die Bedingung wird erfüllt) oder `false` (die Bedingung wird nicht erfüllt).

Informationen zu den Variablen, die in Ausdrücken verwendet werden können, finden Sie in „Variablen" auf Seite 25. Die Verwendung bestimmter Variablen hängt vom Kontext des Ausdrucks ab.

Attribute

<filteringPredicate> weist das folgende Attribut auf:

Tabelle 39. Attribute des Elements <filteringPredicate>

Name	Beschreibung	Datentyp	Erforderlich?
expressionLanguage	Gibt die Programmiersprache an, in der der Ausdruck geschrieben wurde. Da die Programmiersprache Java die einzige unterstützte Sprache für Ausdrücke ist, ist nur der Wert `java` für dieses Attribut gültig.	xsd:NMTOKEN	Ja

Enthalten in

<filteringPredicate> ist in den folgenden Elementen enthalten:

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

Enthält

<filteringPredicate> enthält keine Elemente.

Zugehörige Konzepte

„Ausdrücke" auf Seite 20
Ein Ausdruck ist Code, der eine angepasste Logik enthält, die einer Regel hinzugefügt werden kann. Ausdrücke können auch auf Code außerhalb der ACT-Engine zugreifen. In der Regelsprache sind Ausdrücke nur in bestimmten Kontexten, oder Regelsprachelementen, gültig.

Element 'filterRule'

Das Element <filterRule> definiert eine Regel entsprechend des Filtermusters.

Attribute

<filterRule> weist die folgenden Attribute auf:

Tabelle 40. Attribute des Elements <filterRule>

Name	Beschreibung	Datentyp	Erforderlich?
name	Kennzeichnet die Regel. Dieser Bezeichner muss innerhalb des Regelblocks, der diese Regel enthält, eindeutig sein. Er darf keinen Punkt enthalten.	xsd:NMTOKEN	Ja
processOnlyForwardedEvents	Bestimmt, ob die Regel alle Ereignisse oder nur die Ereignisse empfängt, die von anderen Regeln weitergeleitet wurden. Der Standardwert lautet `false`, was bedeutet, dass die Regel alle Ereignisse empfängt, einschließlich der Ereignisse, die von anderen Regeln weitergeleitet wurden.	xsd:boolean	Nein

Enthalten in

<filterRule> ist im folgenden Element enthalten:

- <ruleBlock>

Enthält

<filterRule> enthält die folgenden Elemente.

Die Elemente müssen in der angezeigten Reihenfolge codiert werden. Wenn ein Element optional ist, muss es nicht codiert werden. Alle codierten Elemente müssen jedoch die richtige Reihenfolge aufweisen.

Tabelle 41. Im Element <filterRule> enthaltene Elemente

Element	Erforderlich oder optional?
<comment>	Optional. 0 oder 1 Vorkommen ist zulässig.
<variable>	Optional. 0 oder mehr Vorkommen sind zulässig.
<activationInterval>	Optional. 0 oder 1 Vorkommen ist zulässig.
<lifeCycleActions>	Optional. 0 oder 1 Vorkommen ist zulässig.
<eventSelector>	Optional. 0 oder 1 Vorkommen ist zulässig.
<onDetection>	Optional. 0 oder 1 Vorkommen ist zulässig.

Zugehörige Konzepte

„Filtermuster" auf Seite 13
Eine Filterregel wird durch das Filtermuster definiert. Sie führt eine bestimmte

Aktion aus, wenn sie ein Ereignis akzeptiert. Sie wird nur für ein einzelnes Ereignis ausgeführt und ist somit eine statusunabhängige Regel.

Element 'groupingKey'

Normalerweise verfügt jede aktive Regel über eine Regelinstanz (oder Kopie), die in der ACT-Engine ausgeführt wird. Manchmal ist allerdings dieselbe Regel für verschiedene Ereignisgruppen erforderlich, die oft zu verschiedenen Ressourcengruppen gehören. Der Gruppierungsschlüssel besteht aus mindestens einem Ereignisattribut oder Teilen von Ereignisattributen, das bzw. die verwendet werden können, um die ausgewählten Ereignisse für eine einheitliche Verarbeitung als Gruppe in unterschiedliche Gruppen zu teilen. Das Element <groupingKey> definiert den Gruppierungsschlüssel für eine Regel. Das Element <groupingKey> weist die Regel an, für jede Gruppe von Ereignissen, die allgemeine Merkmale gemeinsam nutzen (die durch die Werte ihrer Attribute definiert sind, die den Gruppierungsschlüssel bilden), eine getrennte Regelinstanz (oder eine Kopie) zu erstellen.

Details

Die folgenden beiden Szenarios verdeutlichen die Bedeutung des Gruppierungsschlüssels.

Szenario 1:
Die beiden Ereignisse „DB2down" und „DB2up" treten auf. DB2 wird auf drei Computern mit den Namen A, B und C ausgeführt. Eine Sequenzregel wird definiert, um ein Ereignis „DB2down" mit einem Ereignis „DB2up" zu korrelieren und den Bediener zu benachrichtigen, wenn DB2 stoppt und nicht erneut startet.

Wenn die Sequenzregel ohne den Gruppierungsschlüssel definiert wird und ein Ereignis „DB2down" von Computer A empfangen wird, würde ein Ereignis „DB2up" von einem beliebigen Computer die Sequenz beenden, was aber nicht beabsichtigt wäre. Wenn die Gruppierungsschlüssel jedoch als Attribut „hostname" definiert wurden, würde für jeden eindeutigen Wert des Attributs „hostname" in den ausgewählten Ereignissen eine eindeutige Kopie oder eine Instanz der Regel erzeugt. Die ACT-Engine würde jedes Ereignis an die richtige Regelinstanz senden (die Regelinstanz für den Wert „hostname" des Ereignisses). Wird also ein Ereignis „DB2down" von Computer A empfangen, würde die ACT-Engine eine Regelinstanz für Computer A erzeugen. Wenn ein Ereignis „DB2down" von Computer B empfangen wird, würde die ACT-Engine eine zweite Regelinstanz für Computer B erzeugen. Wenn ein Ereignis „DB2up" von Computer B empfangen wird, verarbeitet die ACT-Engine dieses Ereignis in der zweiten Regelinstanz. Die Sequenz ist vollständig, und der Operator wird benachrichtigt, weil die Ereignisse „DB2down" und „DB2up" von Computer B richtig korreliert werden.

Szenario 2:
Ein Ereignis für eine Nachricht über einen ungültigen Anmeldeversuch tritt auf allen Computern in einer bestimmten Umgebung auf. Das Ereignis enthält eine Benutzer-ID. Eine Schwellenwertregel ist definiert, um eine Warnung an den Operator auszugeben, wenn dieses Ereignis mehr als zehn Mal in fünf Minuten auftritt.

Ein Gruppierungsschlüssel könnte als Benutzer-ID definiert werden. Anschließend würde für jede eindeutige Benutzer-ID eine neue Regelinstanz erzeugt. Jeder Anmeldeversuch eines Benutzers würde in einer eindeutigen Schwellenwertregelinstanz protokolliert, wobei jede Instanz über einen getrennten

Zähler mit der Anzahl Anmeldeversuche dieses Benutzers verfügt. Der Operator würde eine Warnung erhalten, wenn eine beliebige Benutzer-ID zehn ungültige Anmeldeversuche in fünf Minuten überschreitet.

Mögliche Variationen dieses Szenarios:

- Der Gruppierungsschlüssel könnte als Hostname anstatt als Benutzer-ID definiert werden. Mit dieser Option könnte eine große Anzahl ungültiger Anmeldeversuche auf einem einzelnen Computer ermittelt werden.
- Der Gruppierungsschlüssel könnte auch als Kombination von Hostname und Benutzer-ID definiert werden. Mit dieser Option könnte ein möglicher Hackerangriff einer bestimmten Benutzer-ID auf einen bestimmten Computer festgestellt werden.

Wenn dasselbe Attribut in allen Ereignistypen vorhanden ist, die für eine Regel angegeben werden, ist es am einfachsten und gebräuchlichsten, einen Gruppierungsschlüssel mit dem Element <attributeName> zu definieren.

Attribute

<groupingKey> hat das folgende Attribut:

Tabelle 42. Attribute des Elements <groupingKey>

Name	Beschreibung	Datentyp	Erforderlich?
missingAttributeHandling	Definiert die Aktion, die die Regel unter einer der folgenden Bedingungen ausführen muss: • Ein ausgewähltes Ereignis verfügt über ein Attribut, das Teil des Gruppierungsschlüssels ist, aber der Wert für dieses Attribut fehlt. • Der Ausdruck im Element <computedValue> übergibt einen Nullwert. Die Regel behandelt diesen Nullwert als fehlenden Attributwert. Gültige Werte für das Attribut „missingAttributeHandling”: • `ignoreEvent` (Standardwert) - Die Regel ignoriert das Ereignis und führt keine Aktion für das Ereignis aus. • `ignoreAttribute` - Die Regel akzeptiert das Ereignis, sie ignoriert aber das Attribut mit dem fehlenden Wert. Die ACT-Engine fügt anschließend einen Ersatzwert für das Attribut ein.	xsd:string	Nein

Enthalten in

<groupingKey> ist in den folgenden Elementen enthalten:

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

Enthält

<groupingKey> enthält die folgenden Elemente.

Tabelle 43. Im Element <groupingKey> enthaltene Elemente

Element	Erforderlich oder optional?
<attributeAlias>	Eines dieser Elemente ist erforderlich. Die Codierung mehrerer Elemente ist optional. Mehrere Vorkommen aller drei Elemente sind zulässig. Diese Elemente können in beliebiger Reihenfolge codiert werden.
<attributeName>	
<computedValue>	

Element 'import'

Das Element <import> enthält einen Ausdruck, der die externen Module (wie z. B. Java-Klassen) angibt, die zur Verwendung in anderen Ausdrücken in Regeln importiert werden sollen.

Details

Der Ausdruckscode ist eine Zeichenfolge im Element <import>. Der ACT-Compiler verwendet die Importanweisungen, die durch die Elemente <import> bereitgestellt werden, zum Kompilieren von Ausdruckscode in Regeln, die externe Methoden aufrufen.

Attribute

<import> hat das folgende Attribut:

Tabelle 44. Attribute des Elements <import>

Name	Beschreibung	Datentyp	Erforderlich?
expressionLanguage	Gibt die Programmiersprache an, in der der Ausdruck geschrieben wurde. Da die Programmiersprache Java die einzige unterstützte Sprache für Ausdrücke ist, ist nur der Wert `java` für dieses Attribut gültig.	xsd:NMTOKEN	Ja

Enthalten in

<import> ist in den folgenden Elementen enthalten:

- <ruleSet>
- <ruleBlock>

Enthält

<import> enthält keine Elemente.

Zugehörige Konzepte

„Externe Module und Objekte importieren und darauf zugreifen" auf Seite 22
In diesem Beispiel wird gezeigt, wie Sie externen Code (wie z. B. Java-Klassen) sowie externe Objekte für Ausdrücke zugänglich machen können. Ein externes Objekt ist ein Objekt, das von einer Anwendung erstellt wird, um mit Ausdrücken zu kommunizieren.

Element 'inactiveWhenLoaded'

Das Element <inactiveWhenLoaded> gibt an, dass eine Regel inaktiv ist, wenn sie von der ACT-Engine geladen wird. Die Regel bleibt inaktiv, bis sie auf eine andere Art aktiviert wird.

Attribute

<inactiveWhenLoaded> hat keine Attribute.

Enthalten in

<inactiveWhenLoaded> ist im folgenden Element enthalten:

- <start>

Enthält

<inactiveWhenLoaded> enthält keine Elemente.

Element 'lifeCycleActions'

Das Element <lifeCycleActions> enthält Elemente, die Aktionen definieren, die in den vier primären Phasen im Lebenszyklus einer Regel ausgeführt werden sollen.

Details

Die für das Laden und die Aktivierung definierten Aktionen werden aufgerufen, wenn die Regel tatsächlich geladen oder aktiviert wird, und bevor die Regel mit der Verarbeitung beginnt. Die für die Inaktivierung und das Entladen definierten Aktionen werden unmittelbar vor dem tatsächlichen Inaktivieren und Entladen der Regel aufgerufen.

Attribute

<lifeCycleActions> hat keine Attribute.

Enthalten in

<lifeCycleActions> ist in den folgenden Elementen enthalten:

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

- <thresholdRule>
- <timerRule>

Enthält

<lifeCycleActions> enthält die folgenden Elemente.

Die Elemente müssen in der angezeigten Reihenfolge codiert werden. Wenn ein Element optional ist, muss es nicht codiert werden. Alle codierten Elemente müssen jedoch die richtige Reihenfolge aufweisen.

Tabelle 45. Im Element <lifeCycleActions> enthaltenen Elemente

Element	Erforderlich oder optional?
<onLoad>	Optional. 0 oder 1 Vorkommen ist zulässig.
<onActivation>	Optional. 0 oder 1 Vorkommen ist zulässig.
<onDeactivation>	Optional. 0 oder 1 Vorkommen ist zulässig.
<onUnload>	Optional. 0 oder 1 Vorkommen ist zulässig.

Element 'never'

Das Element <never> gibt an, dass eine Regel nie zu einer bestimmten Zeit inaktiviert wird. Eine Regel kann immer noch durch ein Ereignis oder auf eine andere Weise inaktiviert werden.

Attribute

<never> hat keine Attribute.

Enthalten in

<never> ist im folgenden Element enthalten:

- <stop>

Enthält

<never> enthält keine Elemente.

Element 'onActivation'

Das Element <onActivation> gibt die Aktion bzw. die Aktionen an, die ausgeführt werden, wenn die Regel aktiviert wird. Die <onActivation>-Aktion wird nach der Aktivierung der Regel aufgerufen, aber bevor die Regel mit der Verarbeitung beginnt.

Details

Wenn der Regelsatz mehrere Regeln enthält, die zum gleichen Datum und zur gleichen Uhrzeit oder durch das gleiche Ereignis aktiviert wurden und die das gleiche Zeitfenster haben, werden die folgenden Aktionen für diese Regeln nicht exakt zum gleichen Zeitpunkt ausgeführt:

- Regelantwortaktionen in den Elementen <onTimeOut> und <onTimeWindowComplete>
- Lebenszyklusaktionen in den Elementen <onActivation> und <onDeactivation>

Diese Aktionen werden nacheinander in beliebiger Reihenfolge ausgeführt. Sie müssen nicht zwingend in der Reihenfolge ausgeführt werden, in der sie im Regelsatz codiert wurden. Da jede Aktion beendet werden muss, bevor die nächste Aktion in der Sequenz beginnt, werden die Aktionen nicht gleichzeitig ausgeführt.

Attribute

<onActivation> hat keine Attribute.

Enthalten in

<onActivation> ist im folgenden Element enthalten:

- <lifeCycleActions>

Enthält

<onActivation> enthält das folgende Element:

Tabelle 46. Im Element <onActivation> enthaltenen Elemente

Element	Erforderlich oder optional?
<action>	Optional. 0 oder mehr Vorkommen sind zulässig.

Element 'onDeactivation'

Das Element <onDeactivation> gibt die Aktion bzw. die Aktionen an, die ausgeführt werden, wenn die Regel inaktiviert wird. Die Aktion <onDeactivation> wird unmittelbar vor der Inaktivierung der Regel aufgerufen.

Details

Wenn der Regelsatz mehrere Regeln enthält, die zum gleichen Datum und zur gleichen Uhrzeit oder durch das gleiche Ereignis aktiviert wurden und die das gleiche Zeitfenster haben, werden die folgenden Aktionen für diese Regeln nicht exakt zum gleichen Zeitpunkt ausgeführt:

- Regelantwortaktionen in den Elementen <onTimeOut> und <onTimeWindowComplete>
- Lebenszyklusaktionen in den Elementen <onActivation> und <onDeactivation>

Diese Aktionen werden nacheinander in beliebiger Reihenfolge ausgeführt. Sie müssen nicht zwingend in der Reihenfolge ausgeführt werden, in der sie im Regelsatz codiert wurden. Da jede Aktion beendet werden muss, bevor die nächste Aktion in der Sequenz beginnt, werden die Aktionen nicht gleichzeitig ausgeführt.

Attribute

<onDeactivation> hat keine Attribute.

Enthalten in

<onDeactivation> ist im folgenden Element enthalten:

- <lifeCycleActions>

Enthält

<onDeactivation> enthält das folgende Element:

Tabelle 47. Im Element <onDeactivation> enthaltene Elemente

Element	Erforderlich oder optional?
<action>	Optional. 0 oder mehr Vorkommen sind zulässig.

Element 'onDetection'

Das Element <onDetection> ist nur für die Duplikat-, Filter-, Sequenz- und Schwellenwertregeln gültig. Es gibt die Aktion bzw. die Aktionen an, die ausgeführt werden, wenn das Regelmuster ermittelt wird.

Details

Tabelle 48 beschreibt, wie das Regelmuster für jeden Regeltyp ermittelt wird, für den die Aktion <onDetection> gültig ist.

Tabelle 48. Ermitteln eines Regelmusters basierend auf dem Regeltyp

Regeltyp	Ermitteln eines Regelmusters
Duplikat	Dieses Regelmuster wird ermittelt, wenn das erste Ereignis empfangen wird, das den Auswahlkriterien für Ereignisse entspricht.
Filter	Dieses Regelmuster wird ermittelt, wenn ein beliebiges Ereignis empfangen wird, das den Auswahlkriterien für Ereignisse entspricht.
Sequenz	Dieses Regelmuster wird ermittelt, wenn eine Folge von Ereignissen, die den Auswahlkriterien für Ereignisse entsprechen, in der richtigen Reihenfolge und im Zeitfenster empfangen wird.
Schwellenwert	Dieses Regelmuster wird ermittelt, wenn Ereignisse, die den Auswahlkriterien für Ereignisse entsprechen, im Zeitfenster empfangen werden und den Schwellenwert erfüllen.

Attribute

<onDetection> hat keine Attribute.

Enthalten in

<onDetection> ist in den folgenden Elementen enthalten:

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

Enthält

<onDetection> enthält das folgende Element:

Tabelle 49. Im Element <onDetection> enthaltene Elemente

Element	Erforderlich oder optional?
<action>	Optional. 0 oder mehr Vorkommen sind zulässig.

Element 'onLoad'

Das Element <onLoad> gibt die Aktion bzw. die Aktionen an, die ausgeführt werden, wenn die Regel in die aktive ACT-Engine geladen (oder implementiert) wird. Die <onLoad>-Aktion wird nach dem Laden der Regel aufgerufen, aber bevor die Regel mit der Verarbeitung beginnt.

Attribute

<onLoad> hat keine Attribute.

Enthalten in

<onLoad> ist im folgenden Element enthalten:

- <lifeCycleActions>

Enthält

<onLoad> enthält das folgende Element:

Tabelle 50. Im Element <onLoad> enthaltene Elemente

Element	Erforderlich oder optional?
<action>	Optional. 0 oder mehr Vorkommen sind zulässig.

Element 'onNextEvent'

Das Element <onNextEvent> ist nur für die Duplikatregel gültig. Es gibt die Aktion bzw. die Aktionen an, die ausgeführt werden, wenn die Duplikatregel das zweite und jedes nachfolgende Ereignis empfängt, das den Auswahlkriterien für Ereignisse im angegebenen Zeitfenster entspricht.

Details

Bei Duplikatregeln überspringt die ACT-Engine die Verarbeitung der Regelsätze für das zweite und jedes nachfolgende Ereignis, das den Auswahlkriterien für Ereignisse im angegebenen Zeitfenster entspricht. Die <onNextEvent>-Aktion muss daher nur codiert werden, um eine alternative Verarbeitung des zweiten und jedes nachfolgenden Ereignisses anzugeben.

Attribute

<onNextEvent> hat keine Attribute.

Enthalten in

<onNextEvent> ist im folgenden Element enthalten:

- <duplicateRule>

Enthält

<onNextEvent> enthält das folgende Element:

Tabelle 51. Im Element <onNextEvent> enthaltene Elemente

Element	Erforderlich oder optional?
<action>	Optional. 0 oder mehr Vorkommen sind zulässig.

Element 'onTimeOut'

Das Element <onTimeOut> ist nur für Sequenz- und Schwellenwertregeln gültig. Es gibt die Aktion bzw. die Aktionen an, die ausgeführt werden, wenn das Zeitfenster für die Regel abläuft.

Details

Tabelle 52 beschreibt, wie das Zeitfenster für jeden Regeltyp mit einer gültigen <onTimeOut>-Aktion abläuft.

Tabelle 52. Ablaufen des Zeitfensters basierend auf dem Regeltyp

Regeltyp	Ablaufen des Zeitfensters
Sequenz	Das Zeitfenster läuft ab, wenn mindestens ein Ereignis akzeptiert wurde, die vollständige Sequenz der Ereignisse aber nicht im Zeitfenster empfangen wurde.
Schwellenwert	Das Zeitfenster läuft ab, wenn mindestens ein Ereignis akzeptiert wurde, der Schwellenwert aber nicht im Zeitfenster erreicht wurde.

Wenn der Regelsatz mehrere Regeln enthält, die zum gleichen Datum und zur gleichen Uhrzeit oder durch das gleiche Ereignis aktiviert wurden und die das gleiche Zeitfenster haben, werden die folgenden Aktionen für diese Regeln nicht exakt zum gleichen Zeitpunkt ausgeführt:

- Regelantwortaktionen in den Elementen <onTimeOut> und <onTimeWindowComplete>
- Lebenszyklusaktionen in den Elementen <onActivation> und <onDeactivation>

Diese Aktionen werden nacheinander in beliebiger Reihenfolge ausgeführt. Sie müssen nicht zwingend in der Reihenfolge ausgeführt werden, in der sie im Regelsatz codiert wurden. Da jede Aktion beendet werden muss, bevor die nächste Aktion in der Sequenz beginnt, werden die Aktionen nicht gleichzeitig ausgeführt.

Attribute

<onTimeOut> hat keine Attribute.

Enthalten in

<onTimeOut> ist in den folgenden Elementen enthalten:

- <sequenceRule>
- <thresholdRule>

Enthält

<onTimeOut> enthält das folgende Element:

Tabelle 53. Im Element <onTimeOut> enthaltene Elemente

Element	Erforderlich oder optional?
<action>	Optional. 0 oder mehr Vorkommen sind zulässig.

Element 'onTimeWindowComplete'

Das Element <onTimeWindowComplete> ist nur für Datenerfassungs-, Berechnungs-, Duplikat- und Zeitgeberregeln gültig. Es gibt die Aktion bzw. die Aktionen an, die ausgeführt werden, wenn das Zeitfenster für die Regel abgelaufen ist.

Details

Wenn der Regelsatz mehrere Regeln enthält, die zum gleichen Datum und zur gleichen Uhrzeit oder durch das gleiche Ereignis aktiviert wurden und die das gleiche Zeitfenster haben, werden die folgenden Aktionen für diese Regeln nicht exakt zum gleichen Zeitpunkt ausgeführt:

- Regelantwortaktionen in den Elementen <onTimeOut> und <onTimeWindowComplete>
- Lebenszyklusaktionen in den Elementen <onActivation> und <onDeactivation>

Diese Aktionen werden nacheinander in beliebiger Reihenfolge ausgeführt. Sie müssen nicht zwingend in der Reihenfolge ausgeführt werden, in der sie im Regelsatz codiert wurden. Da jede Aktion beendet werden muss, bevor die nächste Aktion in der Sequenz beginnt, werden die Aktionen nicht gleichzeitig ausgeführt.

Attribute

<onTimeWindowComplete> hat keine Attribute.

Enthalten in

<onTimeWindowComplete> ist in den folgenden Elementen enthalten:

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

Enthält

<onTimeWindowComplete> enthält das folgende Element:

Tabelle 54. Im Element <onTimeWindowComplete> enthaltene Elemente

Element	Erforderlich oder optional?
<action>	Optional. 0 oder mehr Vorkommen sind zulässig.

Element 'onUnload'

Das Element <onUnload> gibt die Aktion bzw. die Aktionen an, die ausgeführt werden, wenn die Regel aus der aktiven ACT-Engine entladen oder entfernt wird. Die <onUnload>-Aktion wird unmittelbar vor dem Entladen der Regel aufgerufen.

Attribute

<onUnload> hat keine Attribute.

Enthalten in

<onUnload> ist im folgenden Element enthalten:

- <lifeCycleActions>

Enthält

<onUnload> enthält das folgende Element:

Tabelle 55. Im Element <onUnload> enthaltene Elemente

Element	Erforderlich oder optional?
<action>	Optional. 0 oder mehr Vorkommen sind zulässig.

Element 'ruleBlock'

Das Element <ruleBlock> ermöglicht es, zusammengehörige Regeln zu gruppieren und Regeln in einer Hierarchie zu organisieren.

Attribute

<ruleBlock> hat das folgende Attribut:

Tabelle 56. Attribute des Elements <ruleBlock>

Name	Beschreibung	Datentyp	Erforderlich?
name	Gibt den Regelblock an. Diese Kennung muss im Regelsatz oder Regelblock, der diesen Regelblock enthält, eindeutig sein. Er darf keinen Punkt enthalten.	xsd:NMTOKEN	Ja

Enthalten in

<ruleBlock> ist in den folgenden Elementen enthalten:

- <ruleSet>
- <ruleBlock>

Enthält

<ruleBlock> enthält die folgenden Elemente.

Wenn sie codiert sind, müssen die Elemente <comment>, <import> und <variable> in der angezeigten Reihenfolge codiert werden. Die restlichen Elemente können in beliebiger Reihenfolge codiert werden.

Tabelle 57. Im Element <ruleBlock> enthaltene Elemente

Element	Erforderlich oder optional?
<comment>	Optional. 0 oder 1 Vorkommen ist zulässig.
<import>	Optional. 0 oder mehr Vorkommen sind zulässig.
<variable>	Optional. 0 oder mehr Vorkommen sind zulässig.
<ruleBlock>	Optional. 0 oder mehr Vorkommen sind zulässig.
<collectionRule>	Optional. 0 oder mehr Vorkommen sind zulässig.
<computationRule>	Optional. 0 oder mehr Vorkommen sind zulässig.
<duplicateRule>	Optional. 0 oder mehr Vorkommen sind zulässig.
<filterRule>	Optional. 0 oder mehr Vorkommen sind zulässig.
<sequenceRule>	Optional. 0 oder mehr Vorkommen sind zulässig.
<thresholdRule>	Optional. 0 oder mehr Vorkommen sind zulässig.
<timerRule>	Optional. 0 oder mehr Vorkommen sind zulässig.

Element 'ruleSet'

Das Element <ruleSet>, das durch `act:ruleSet` definiert ist, ist das Stammelement für die ACT-Regelsprache. Alle anderen Elemente sind in diesem Element <ruleSet> enthalten.

Details

Die Elemente <ruleSet>, die durch das ACT-Sprachschema (`act:ruleSet`) und durch das ACT-Basisregelsatzschema (`br:ruleSet`) definiert werden, sind Duplikate. Wenn Sie jedoch einen Regelsatz erstellen, müssen Sie den folgenden Namensbereich für das Element <ruleSet> angeben: `act:ruleSet`.

Attribute

<ruleSet> hat das folgende Attribut:

Tabelle 58. Attribute des Elements <ruleSet>

Name	Beschreibung	Datentyp	Erforderlich?
name	Gibt den Regelsatz an. Diese ID muss eindeutig sein. Sie darf keinen Punkt enthalten.	xsd:NMTOKEN	Ja

Enthalten in

Da <ruleSet> das Stammelement für die Regelsprache ist, ist es in keinem Element enthalten.

Enthält

<ruleSet> enthält die folgenden Elemente.

Die Elemente müssen in der angezeigten Reihenfolge codiert werden. Wenn ein Element optional ist, muss es nicht codiert werden. Alle codierten Elemente müssen jedoch die richtige Reihenfolge aufweisen.

Tabelle 59. Im Element <ruleSet> enthaltene Elemente

Element	Erforderlich oder optional?
<comment>	Optional. 0 oder 1 Vorkommen ist zulässig.
<import>	Optional. 0 oder mehr Vorkommen sind zulässig.
<variable>	Optional. 0 oder mehr Vorkommen sind zulässig.
<ruleBlock>	Optional. 0 oder mehr Vorkommen sind zulässig.

Element 'runUntilDeactivated'

Das Element <runUntilDeactivated> gibt an, dass das Zeitfenster aktiv bleiben soll, bis die Regel inaktiviert wird. Das Zeitfenster für diese Regel startet also, wenn die Regel die Verarbeitung beginnt, und wird erst gestoppt, wenn die Regel inaktiviert oder aus dem Regelsatz gelöscht wird oder wenn die ACT-Engine heruntergefahren wird.

Details

Das spezifische Verhalten einer Regel mit dem Element <runUntilDeactivated> hängt vom Regeltyp ab. Tabelle 60 beschreibt das Regelverhalten für jeden Regeltyp, für den das Element <timeWindow> gültig ist und der das Element <runUntilDeactivated> enthält.

Tabelle 60. Regelverhalten, wenn <runUntilDeactivated> codiert ist

Regeltyp	Regelverhalten, wenn <runUntilDeactivated> codiert ist
Datenerfassung	Die Datenerfassungsregel akzeptiert das erste Ereignis, das ihren Auswahlkriterien für Ereignisse entspricht. Sie akzeptiert und verarbeitet weiterhin Ereignisse, bis die Regel inaktiviert wird. Zu diesem Zeitpunkt werden die Aktionen ausgeführt, die im Element <onTimeWindowComplete> definiert sind, unmittelbar gefolgt von den Aktionen, die im Element <onDeactivation> definiert sind.
Berechnung	Die Berechnungsregel akzeptiert das erste Ereignis, das ihren Auswahlkriterien für Ereignisse entspricht. Sie akzeptiert und verarbeitet weiterhin Ereignisse, bis die Regel inaktiviert wird. Zu diesem Zeitpunkt werden die Aktionen ausgeführt, die im Element <onTimeWindowComplete> definiert sind, unmittelbar gefolgt von den Aktionen, die im Element <onDeactivation> definiert sind.
Duplikat	Die Duplikatregel akzeptiert das erste Ereignis, das ihren Auswahlkriterien für Ereignisse entspricht. Sie akzeptiert und verarbeitet weiterhin Ereignisse, bis die Regel inaktiviert wird. Zu diesem Zeitpunkt werden die Aktionen ausgeführt, die im Element <onTimeWindowComplete> definiert sind, unmittelbar gefolgt von den Aktionen, die im Element <onDeactivation> definiert sind.

Tabelle 60. Regelverhalten, wenn <runUntilDeactivated> codiert ist (Forts.)

Regeltyp	Regelverhalten, wenn <runUntilDeactivated> codiert ist
Sequenz	Die Sequenzregel akzeptiert das erste Ereignis, das ihren Auswahlkriterien für Ereignisse entspricht, und sie akzeptiert und verarbeitet weiterhin Ereignisse, bis eine der folgenden Bedingungen auftritt: • Das Sequenzmuster wird ermittelt. Wenn dies auftritt, werden die Aktionen ausgeführt, die im Element <onDetection> definiert sind, und die Regel kehrt in ihren Anfangsstatus zurück. Die Ereignisverarbeitung durch diese Regel beginnt erneut, und der Prozess kann viele Male wiederholt werden, bis die Regel inaktiviert wird. • Die Regel wird inaktiviert, während sie Ereignisse verarbeitet. Wenn dies auftritt, werden die Aktionen ausgeführt, die im Element <onTimeOut> definiert sind, gefolgt von den Aktionen, die im Element <onDeactivation> definiert sind.
Schwellenwert	Die Schwellenwertregel akzeptiert das erste Ereignis, das ihren Auswahlkriterien für Ereignisse entspricht, und sie akzeptiert und verarbeitet weiterhin Ereignisse, bis eine der folgenden Bedingungen auftritt: • Das Schwellenwertmuster wird ermittelt. Wenn dies auftritt, werden die Aktionen ausgeführt, die im Element <onDetection> definiert sind, und die Regel kehrt in ihren Anfangsstatus zurück. Die Ereignisverarbeitung durch diese Regel beginnt erneut, und der Prozess kann viele Male wiederholt werden, bis die Regel inaktiviert wird. • Die Regel wird inaktiviert, während sie Ereignisse verarbeitet. Wenn dies auftritt, werden die Aktionen ausgeführt, die im Element <onTimeOut> definiert sind, gefolgt von den Aktionen, die im Element <onDeactivation> definiert sind.
Zeitgeber	Nachdem die Zeitgeberregel aktiviert wurde, führt sie keine Aktionen aus, bis sie inaktiviert wird. Zu diesem Zeitpunkt werden die Aktionen ausgeführt, die im Element <onTimeWindowComplete> definiert sind, unmittelbar gefolgt von den Aktionen, die im Element <onDeactivation> definiert sind. Das Wiederholungsattribut im Element <timerRule> wird ignoriert.

Attribute

<runUntilDeactivated> hat keine Attribute.

Enthalten in

<runUntilDeactivated> ist im folgenden Element enthalten:

- <timeWindow>

Enthält

<runUntilDeactivated> enthält keine Elemente.

Element 'sequenceRule'

Das Element <sequenceRule> definiert eine Regel nach dem Sequenzmuster. Die Sequenzregel ist die einzige Regel, die mehrere Ereignisselektoren zulässt. Es sind außerdem mindestens zwei Ereignisselektoren erforderlich.

Attribute

<sequenceRule> hat die folgenden Attribute:

Tabelle 61. Attribute des Elements <sequenceRule>

Name	Beschreibung	Datentyp	Erforderlich?
name	Kennzeichnet die Regel. Dieser Bezeichner muss innerhalb des Regelblocks, der diese Regel enthält, eindeutig sein. Er darf keinen Punkt enthalten.	xsd:NMTOKEN	Ja
processOnlyForwardedEvents	Bestimmt, ob die Regel alle Ereignisse oder nur die Ereignisse empfängt, die von anderen Regeln weitergeleitet wurden. Der Standardwert lautet `false`, was bedeutet, dass die Regel alle Ereignisse empfängt, einschließlich der Ereignisse, die von anderen Regeln weitergeleitet wurden.	xsd:boolean	Nein
arrivalOrder	Definiert, ob die Ereignisse in der Reihenfolge empfangen werden müssen, in der die Elemente <eventSelector> für die Regel codiert sind. Die folgenden Werte sind gültig: • `inOrder` (Standardwert) • `randomOrder`	xsd:string	Nein

Wenn der Wert des Attributs 'arrivalOrder' `randomOrder` ist, ist die Reihenfolge wichtig, in der die Elemente <eventSelector> codiert sind. Die Elemente <eventSelector> mit den genauesten Auswahlkriterien für Ereignisse sollten vor den Elementen <eventSelector> mit ungenaueren Auswahlkriterien für Ereignisse codiert werden. Andernfalls wird die Sequenz nicht erkannt, wenn sie erkannt werden sollte.

Setzen Sie z. B. folgende Umstände voraus:

- Drei Elemente <eventSelector> sind definiert.
- Das erste Element <eventSelector> überprüft das Vorhandensein von Ereignis `eventA`.
- Das zweite Element <eventSelector> überprüft das Vorhandensein eines beliebigen Ereignisses.
- Das dritte Element <eventSelector> überprüft das Vorhandensein von Ereignis `eventB`.
- Die folgenden Ereignisse werden im angegebenen Zeitfenster an das System übergeben: `eventA`, `eventB`, `eventC`.

Die Regel verhält sich wie folgt, und die Sequenz wird nicht erkannt, wenn sie erkannt werden sollte:

1. Das erste Ereignis, `eventA`, wird vom ersten Element <eventSelector> akzeptiert.
2. Das zweite Ereignis, `eventB`, wird vom zweiten Element <eventSelector> akzeptiert.

3. Das dritte Ereignis, `eventC`, wird ignoriert.

Setzen Sie die folgenden Umstände voraus, in denen die Elemente <eventSelector> korrekt codiert wurden, d. h. zuerst die genauesten und anschließend die ungenaueren Auswahlkriterien für Ereignisse:

- Das erste Element <eventSelector> überprüft das Vorhandensein von Ereignis `eventA`.
- Das zweite Element <eventSelector> überprüft das Vorhandensein von Ereignis `eventB`.
- Das dritte Element <eventSelector> überprüft das Vorhandensein eines beliebigen Ereignisses.

Die Regel verhält sich wie folgt, und die Sequenz wird erkannt:

1. Das erste Ereignis, `eventA`, wird vom ersten Element <eventSelector> akzeptiert.
2. Das zweite Ereignis, `eventB`, wird vom zweiten Element <eventSelector> akzeptiert.
3. Das dritte Ereignis, `eventC`, wird vom dritten Element <eventSelector> akzeptiert.

Enthalten in

<sequenceRule> ist im folgenden Element enthalten:

- <ruleBlock>

Enthält

<sequenceRule> enthält die folgenden Elemente.

Die Elemente müssen in der angezeigten Reihenfolge codiert werden. Wenn ein Element optional ist, muss es nicht codiert werden. Alle codierten Elemente müssen jedoch die richtige Reihenfolge aufweisen.

Tabelle 62. Im Element <sequenceRule> enthaltene Elemente

Element	Erforderlich oder optional?
<comment>	Optional. 0 oder 1 Vorkommen ist zulässig.
<variable>	Optional. 0 oder mehr Vorkommen sind zulässig.
<activationInterval>	Optional. 0 oder 1 Vorkommen ist zulässig.
<lifeCycleActions>	Optional. 0 oder 1 Vorkommen ist zulässig.
<eventSelector>	Für die Sequenzregel sind zwei Vorkommen dieses Elements erforderlich. Zusätzliche Vorkommen sind zulässig.
<groupingKey>	Optional. 0 oder 1 Vorkommen ist zulässig.
<timeWindow>	Erforderlich. Nur 1 Vorkommen ist zulässig.
<onDetection>	Optional. 0 oder 1 Vorkommen ist zulässig.
<onTimeOut>	Optional. 0 oder 1 Vorkommen ist zulässig.

Zugehörige Konzepte

„Sequenzmuster" auf Seite 14
Eine Sequenzregel wird durch das Sequenzmuster definiert. Es ermittelt, ob eine gewisse Sequenz von Ereignissen in einem Zeitintervall ankommt. Die Reihenfolge kann sortiert oder zufällig sein. Eine Sequenzregel ist eine statusabhängige Regel.

Element 'start'

Das Element <start> definiert, ob eine Regel zu einer bestimmten Zeit an einem bestimmten Datum oder beim Laden der Regel durch die ACT-Engine aktiviert wird.

Details

Wenn das Element <start> überhaupt nicht codiert ist, entspricht die Standardstartzeit der Zeit, die durch das Element <whenLoaded> definiert wird.

Attribute

<start> hat keine Attribute.

Enthalten in

<start> ist im folgenden Element enthalten:

- <activationTime>

Enthält

<start> enthält die folgenden Elemente:

Tabelle 63. Im Element <start> enthaltene Elemente

Element	Erforderlich oder optional?
<dateTime>	Eines dieser Elemente ist erforderlich, und nur ein Vorkommen der ausgewählten Elemente ist zulässig.
<whenLoaded>	
<inactiveWhenLoaded>	

Element 'stop'

Das Element <stop> definiert, ob eine Regel zu einer bestimmten Zeit an einem bestimmten Datum, nach einer bestimmten Dauer oder nie zu einem bestimmten Zeitpunkt inaktiviert wird.

Details

Wenn das Element <stop> überhaupt nicht codiert ist, entspricht die Standardstoppzeit der Zeit, die durch das Element <never> definiert wird.

Attribute

Das Element <stop> hat keine Attribute.

Enthalten in

Das Element <stop> ist im folgenden Element enthalten:

- <activationTime>

Enthält

Das Element <stop> enthält die folgenden Elemente:

Tabelle 64. Im Element <stop> enthaltene Elemente

Element	Erforderlich oder optional?
<dateTime>	Eines dieser Elemente ist erforderlich, und nur ein Vorkommen der ausgewählten Elemente ist zulässig.
<never>	
<after>	

Element 'stopAfter'

Das Element <stopAfter> gibt die Zeitdauer an, die eine Regelinstanz, die durch das Element <groupingKey> definiert wird, nach dem Aktivieren aktiv bleiben soll. Nach dieser Zeitdauer soll die Regelinstanz inaktiviert werden.

Attribute

Das Element <stopAfter> hat die folgenden Attribute:

Tabelle 65. Attribute des Elements <stopAfter>

Name	Beschreibung	Datentyp	Erforderlich?
duration	Gibt die Zeitdauer an. Der Datentyp dieses Attributs hängt vom Wert des Attributs für die Einheit ab.	• Wenn der Wert des Attributs für die Einheit `ISO-8601` lautet, lautet der Datentyp 'xsd:duration'. • Wenn der Wert des Attributs für die Einheit `milliseconds` lautet, lautet der Datentyp 'xsd:positiveInteger'.	Ja
unit	Gibt die zu verwendende Zeiteinheit an. Für dieses Attribut sind folgende Werte gültig: • `ISO-8601` • `milliseconds`	xsd:string	Ja

Verwendung des ISO-Standards 8601 für die Dauer

Die Codierung von `ISO-8601` als Wert für das Attribut für die Einheit bedeutet, dass der Wert des Attributs für die Dauer entsprechend des ISO-Standards 8601 für die Angabe einer Dauer als Zeichenfolge codiert wird. Die Standard-XML-Schemadatentypspezifikation verwendet ISO 8601, um einen Datentyp mit dem Namen `duration` bereitzustellen. Dieser Datentyp wird unter http://www.w3.org/TR/xmlschema-2/#duration genauer beschrieben.

Das Format für den Datentyp `duration` im Standard-XML-Schema ist die folgende Zeichenfolge:

`PnYnMnDTnHnMnS`

- `P` ist das Zeichen, mit dem die Zeichenfolge immer beginnt.

- *n*Y stellt die Anzahl Jahre dar. Ein Jahr entspricht 365 Tagen. Daher entspricht die Codierung 1Y der Codierung 365D.
- *n*M stellt die Anzahl Monate dar. Ein Monat entspricht 30 Tagen. Daher entspricht die Codierung 1M der Codierung 30D.
- *n*D stellt die Anzahl Tage dar.
- T ist ein Trennzeichen, das Tageseinheiten (Jahre, Monate und Tage) von Zeiteinheiten (Stunden, Minuten und Sekunden) trennt. Zeiteinheiten folgen immer auf T.
- *n*H stellt die Anzahl Stunden dar.
- *n*M stellt die Anzahl Minuten dar.
- *n*S stellt die Anzahl Sekunden dar.

Im Folgenden finden Sie Beispiele zum Format:

- P5DT12H entspricht 5,5 Tagen.
- PT59M59S entspricht 59 Minuten und 59 Sekunden.
- P1M entspricht 1 Monat.

Enthalten in

<stopAfter> ist im Element <activateOnEvent> enthalten, aber nur, wenn <activateOnEvent> im Element <activationByGroupingKey> codiert ist.

Enthält

<stopAfter> enthält keine Elemente.

Element 'thresholdRule'

Das Element <thresholdRule> definiert eine Regel nach dem Schwellenwertmuster.

Attribute

<thresholdRule> hat die folgenden Attribute:

Tabelle 66. Attribute des Elements <thresholdRule>

Name	Beschreibung	Datentyp	Erforderlich?
name	Kennzeichnet die Regel. Dieser Bezeichner muss innerhalb des Regelblocks, der diese Regel enthält, eindeutig sein. Er darf keinen Punkt enthalten.	xsd:NMTOKEN	Ja
processOnlyForwardedEvents	Bestimmt, ob die Regel alle Ereignisse oder nur die Ereignisse empfängt, die von anderen Regeln weitergeleitet wurden. Der Standardwert lautet false, was bedeutet, dass die Regel alle Ereignisse empfängt, einschließlich der Ereignisse, die von anderen Regeln weitergeleitet wurden.	xsd:boolean	Nein

Enthalten in

<thresholdRule> ist im folgenden Element enthalten:

- <ruleBlock>

Enthält

<thresholdRule> enthält die folgenden Elemente.

Die Elemente müssen in der angezeigten Reihenfolge codiert werden. Wenn ein Element optional ist, muss es nicht codiert werden. Alle codierten Elemente müssen jedoch die richtige Reihenfolge aufweisen.

Tabelle 67. Im Element <thresholdRule> enthaltene Elemente

Element	**Erforderlich oder optional?**
<comment>	Optional. 0 oder 1 Vorkommen ist zulässig.
<variable>	Optional. 0 oder mehr Vorkommen sind zulässig.
<activationInterval>	Optional. 0 oder 1 Vorkommen ist zulässig.
<lifeCycleActions>	Optional. 0 oder 1 Vorkommen ist zulässig.
<eventSelector>	Optional. 0 oder 1 Vorkommen ist zulässig.
<groupingKey>	Optional. 0 oder 1 Vorkommen ist zulässig.
<booleanThreshold>	Eines dieser Elemente ist erforderlich, und nur ein Vorkommen der ausgewählten Elemente ist zulässig.
<computedThreshold>	
<eventCountThreshold>	
<timeWindow>	Erforderlich. Nur 1 Vorkommen ist zulässig.
<onDetection>	Optional. 0 oder 1 Vorkommen ist zulässig.
<onTimeOut>	Optional. 0 oder 1 Vorkommen ist zulässig.

Zugehörige Konzepte

„Schwellenwertmuster" auf Seite 17
Eine Schwellenwertregel wird durch das Schwellenwertmuster definiert. Es erfasst eine Gruppe ausgewählter Ereignisse in einem Zeitintervall und ermittelt, nachdem alle Ereignisse empfangen wurden, ob eine Schwellenwertbedingung zutrifft. Es ist eine statusabhängige Regel.

Element 'timeInterval'

Das Element <timeInterval> gibt die Dauer für das Zeitfenster an.

Attribute

<timeInterval> hat die folgenden Attribute:

Tabelle 68. Attribute des Elements <timeInterval>

Name	Beschreibung	Datentyp	Erforderlich?
duration	Gibt die Zeitdauer an. Der Datentyp dieses Attributs hängt vom Wert des Attributs für die Einheit ab.	• Wenn der Wert des Attributs für die Einheit `ISO-8601` lautet, lautet der Datentyp 'xsd:duration'. • Wenn der Wert des Attributs für die Einheit `milliseconds` lautet, lautet der Datentyp 'xsd:positiveInteger'.	Ja
unit	Gibt die zu verwendende Zeiteinheit an. Für dieses Attribut sind folgende Werte gültig: • `ISO-8601` • `milliseconds`	xsd:string	Ja

Verwendung des ISO-Standards 8601 für die Dauer

Die Codierung von `ISO-8601` als Wert für das Attribut für die Einheit bedeutet, dass der Wert des Attributs für die Dauer entsprechend des ISO-Standards 8601 für die Angabe einer Dauer als Zeichenfolge codiert wird. Die Standard-XML-Schemadatentypspezifikation verwendet ISO 8601, um einen Datentyp mit dem Namen `duration` bereitzustellen. Dieser Datentyp wird unter http://www.w3.org/TR/xmlschema-2/#duration genauer beschrieben.

Das Format für den Datentyp `duration` im Standard-XML-Schema ist die folgende Zeichenfolge:

`PnYnMnDTnHnMnS`

- `P` ist das Zeichen, mit dem die Zeichenfolge immer beginnt.
- *n*`Y` stellt die Anzahl Jahre dar. Ein Jahr entspricht 365 Tagen. Daher entspricht die Codierung `1Y` der Codierung `365D`.
- *n*`M` stellt die Anzahl Monate dar. Ein Monat entspricht 30 Tagen. Daher entspricht die Codierung `1M` der Codierung `30D`.
- *n*`D` stellt die Anzahl Tage dar.
- `T` ist ein Trennzeichen, das Tageseinheiten (Jahre, Monate und Tage) von Zeiteinheiten (Stunden, Minuten und Sekunden) trennt. Zeiteinheiten folgen immer auf `T`.
- *n*`H` stellt die Anzahl Stunden dar.
- *n*`M` stellt die Anzahl Minuten dar.
- *n*`S` stellt die Anzahl Sekunden dar.

Im Folgenden finden Sie Beispiele zum Format:

- `P5DT12H` entspricht 5,5 Tagen.
- `PT59M59S` entspricht 59 Minuten und 59 Sekunden.
- `P1M` entspricht 1 Monat.

Enthalten in

<timeInterval> ist im folgenden Element enthalten:

- <timeWindow>

Enthält

<timeInterval> enthält keine Elemente.

Element 'timerRule'

Das Element <timerRule> definiert eine Regel nach dem Zeitgebermuster.

Attribute

<timerRule> hat die folgenden Attribute:

Tabelle 69. Attribute des Elements <timerRule>

Name	Beschreibung	Datentyp	Erforderlich?
name	Kennzeichnet die Regel. Dieser Bezeichner muss innerhalb des Regelblocks, der diese Regel enthält, eindeutig sein. Er darf keinen Punkt enthalten.	xsd:NMTOKEN	Ja
processOnlyForwardedEvents	Dieses Attribut wird ignoriert, da die Zeitgeberregel keine Ereignisse verarbeitet.	xsd:boolean	Nein
repeat	Definiert, ob eine Zeitgeberregel wiederholt ausgeführt wird, bis sie inaktiviert wird. Die folgenden Werte sind gültig: • `true` (Standardwert) • `false` Wenn der Wert `false` festgelegt ist, wird die Regel im Zeitintervall nur einmal ausgeführt. Die Regelantwortaktion wird ausgeführt, wenn das zugehörige Zeitfenster beendet ist, und anschließend wird die Regel gestoppt. Wenn das Element <timeWindow> für die Zeitgeberregel das Element <runUntilDeactivated> enthält, wird das Attribut „repeat" ignoriert.	xsd:boolean	Nein

Enthalten in

<timerRule> ist im folgenden Element enthalten:

- <ruleBlock>

Enthält

<timerRule> enthält die folgenden Elemente.

Die Elemente müssen in der angezeigten Reihenfolge codiert werden. Wenn ein Element optional ist, muss es nicht codiert werden. Alle codierten Elemente müssen jedoch die richtige Reihenfolge aufweisen.

Tabelle 70. Im Element <timerRule> enthaltene Elemente

Element	Erforderlich oder optional?
<comment>	Optional. 0 oder 1 Vorkommen ist zulässig.
<variable>	Optional. 0 oder mehr Vorkommen sind zulässig.
<activationInterval>	Optional. 0 oder 1 Vorkommen ist zulässig.
<lifeCycleActions>	Optional. 0 oder 1 Vorkommen ist zulässig.
<timeWindow>	Erforderlich. Nur 1 Vorkommen ist zulässig.
<onTimeWindowComplete>	Optional. 0 oder 1 Vorkommen ist zulässig.

Zugehörige Konzepte

„Zeitgebermuster" auf Seite 19
Eine Zeitgeberregel wird durch das Zeitgebermuster definiert. Sie leitet Aktionen in regelmäßigen Intervallen ein. Es ist eine statusabhängige Regel. Obwohl eine Zeitgeberregel keine Ereignisse verarbeitet, kann sie durch ein Ereignis aktiviert oder inaktiviert werden.

Element 'timeWindow'

Das Element <timeWindow> enthält Elemente, die das Zeitintervall definieren, während dessen die Regel verarbeitet wird.

Details

Das Zeitfenster für eine Duplikatregel definiert beispielsweise, wie lange die Regel auf Ereignisse überprüfen muss, die Duplikate des ersten akzeptierten Ereignisses sind. Wenn das Zeitfenster 30 Sekunden beträgt, verarbeitet die Duplikatregel alle doppelten Ereignisse, die während der 30 Sekunden nach dem ersten akzeptierten Ereignis auftreten.

Attribute

<timeWindow> hat keine Attribute.

Enthalten in

<timeWindow> ist in den folgenden Elementen enthalten:

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

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

Enthält

<timeWindow> enthält die folgenden Elemente:

Tabelle 71. Im Element <timeWindow> enthaltene Elemente

Element	Erforderlich oder optional?
<timeInterval>	Eines dieser Elemente ist erforderlich, und nur ein Vorkommen der ausgewählten Elemente ist zulässig.
<runUntilDeactivated>	

Element 'variable'

Das Element <variable> definiert eine Variable. Es enthält Informationen in einem Format, auf das Ausdrücke verweisen können. Eine Variable kann auf der Ebene des Regelsatzes, eines Regelblocks oder einer Regel definiert werden.

Details

Regelsatzvariable
Sie wird global auf den Regelsatz angewendet. Jeder Ausdruck in diesem Regelsatz kann auf sie verweisen.

Regelblockvariable
Sie kann nur im Regelblock (und beliebigen untergeordneten Regelblöcken) angewendet werden. Jeder Ausdruck in diesem Regelblock kann auf sie verweisen.

Regelvariable
Sie wird nur auf Ausdrücke in dieser Regel angewendet.

Variablen können den gleichen Namen auf unterschiedlichen Ebenen in der Regelhierarchie haben. Beim Zugriff auf eine Variable wird die erste Definition einer Variablen ausgehend von der lokalen Ebene verwendet. Wird eine Variable z. B. auf der Ebene des Regelsatzes, des Regelblocks und der Regel mit dem gleichen Namen definiert, wird die Variablendefinition auf der Ebene der Regel von den Ausdrücken in dieser Regel verwendet.

Wenn Variablen auf der Ebene des Regelsatzes oder Regelblocks definiert werden, werden sie zu unterschiedlichen Zeiten von mehreren Regeln abgerufen und festgelegt. Daher sollten die Interaktionen zwischen den Variablen im Regelsatz sorgfältig codiert werden, damit sichergestellt ist, dass die Variablenwerte korrekt verwaltet werden.

Wenn die Variable auf der Ebene des Regelsatzes oder Regelblocks definiert ist, wird sie nicht reinitialisiert, nachdem das Regelmuster abgeglichen wurde.

Unter den beiden folgenden Bedingungen müssen Sperren für das Abrufen und Festlegen von Regelsatz- und Regelblockvariablen verwendet werden, um zu verhindern, dass die Variablen falsch festgelegt werden:

- Wenn eine Zeitgeberregel eine Variable während einer <onTimeOut>-Aktion abruft oder festlegt

- Wenn es sich bei der Anwendung, in die die ACT-Engine eingebettet ist, um eine Multithread-Anwendung handelt

Wenn eine Regel mit einem Gruppierungsschlüssel definiert wird, sind Regelvariablen, die durch das Element <variable> definiert werden, in Lebenszyklusaktionen oder im Element <filteringPredicate> nicht gültig, das in einem Element<activateOnEvent> oder <deactivateOnEvent> enthalten ist, das wiederum in einem Element <activationInterval> enhalten ist. Dies ist der Fall, weil die Regelvariablen in diesem Fall nur auf eine Regelinstanz angewendet werden und Regelinstanzen bei der Ausführung dieser Ausdrücke nicht vorhanden sind.

Attribute

<variable> hat die folgenden Attribute:

Tabelle 72. Attribute des Elements <variable>

Name	Beschreibung	Datentyp	Erforderlich?
name	Gibt eine bestimmte Variable an. Auf eine Variable wird durch ihren Namen verwiesen.	xsd:NMTOKEN	Ja
dataType	Gibt den in der Variablen enthaltenen Informationstyp an. Es muss sich um einen vollständig qualifizierten Datentyp wie z. B. java.lang.String handeln.	xsd:NMTOKEN	Ja

Namenseinschränkungen für Variablen

Variablennamen unterliegen gewissen Einschränkungen. Der Wert des Namensattributs im Element <variable> unterliegt den folgenden Einschränkungen:

- Er darf nur die folgenden Zeichen enthalten:
  - Buchstaben A-Z in Großschreibung (ASCII, lateinischer Zeichensatz). Die Unicode-Darstellung ist `\u0041-\u005a`.
  - Buchstaben a-z in Kleinschreibung (ASCII, lateinischer Zeichensatz). Die Unicode-Darstellung ist `\u0061-\u007a`.
  - Das ASCII-Unterstreichungszeichen (_). Die Unicode-Darstellung ist `\u005f`.
  - Das Dollarzeichen ($). Die Unicode-Darstellung ist `\u0024`.
  - Die ASCII-Ziffern 0 – 9. Die Unicode-Darstellung ist `\u0030-\u0039`.
- Er darf nicht null sein.
- Er darf keine leere Zeichenfolge sein.
- Er darf keine Leerstellen enthalten.
- Er darf keinen Punkt enthalten.
- Er darf nicht mit `act_` in beliebigem Format (keine Großschreibung, Kleinschreibung oder Groß-/Kleinschreibung) beginnen.

Enthalten in

<variable> ist in den folgenden Elementen enthalten:

- <ruleSet>
- <ruleBlock>

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

Enthält

<variable> enthält die folgenden Elemente.

Die Elemente müssen in der angezeigten Reihenfolge codiert werden. Wenn ein Element optional ist, muss es nicht codiert werden. Alle codierten Elemente müssen jedoch die richtige Reihenfolge aufweisen.

Tabelle 73. Im Element <variable> enthaltene Elemente

Element	Erforderlich oder optional?
<comment>	Optional. 0 oder 1 Vorkommen ist zulässig.
<varInitializer>	Erforderlich. 1 Vorkommen ist zulässig.

Zugehörige Konzepte

„Variablen" auf Seite 25
In der Regelsprache werden bestimmte Variablen zum Speichern ereignisbezogener Informationen in unterschiedliche Ereignisvorkommen oder Regeln verwendet. Ausdrücke in Regeln können anschließend auf diese ereignisbezogenen Informationen zugreifen. Manche Variablentypen werden vom Autor der Regel definiert, andere werden von Active Correlation Technology bereitgestellt. Auf einige Typen kann direkt in einem Ausdruck zugegriffen werden, auf andere kann nur mit den von Active Correlation Technology bereitgestellten Methoden zugegriffen werden.

Element 'varInitializer'

Das Element <varInitializer> enthält einen Ausdruck, der den Anfangswert für die Variable bereitstellt, die im zugeordneten Element <variable> definiert ist.

Details

Da es sich um einen beliebigen Variablentyp handelt, kann der Ausdruckscode ein Bereichsobjekt oder ein beliebiges anderes komplexes, auf die Implementierung bezogenes Objekt zurückgeben, das von der ACT-Engine gespeichert wird.

Informationen zu den Variablen, die in Ausdrücken verwendet werden können, finden Sie in „Variablen" auf Seite 25. Die Verwendung bestimmter Variablen hängt vom Kontext des Ausdrucks ab.

Attribute

<varInitializer> hat das folgende Attribut:

Tabelle 74. Attribute des Elements <varInitializer>

Name	Beschreibung	Datentyp	Erforderlich?
expressionLanguage	Gibt die Programmiersprache an, in der der Ausdruck geschrieben wurde. Da die Programmiersprache Java die einzige unterstützte Sprache für Ausdrücke ist, ist nur der Wert `java` für dieses Attribut gültig.	xsd:NMTOKEN	Ja

Enthalten in

<varInitializer> ist im folgenden Element enthalten:

- <variable>

Enthält

<varInitializer> enthält keine Elemente.

Zugehörige Konzepte

„Ausdrücke" auf Seite 20
Ein Ausdruck ist Code, der eine angepasste Logik enthält, die einer Regel hinzugefügt werden kann. Ausdrücke können auch auf Code außerhalb der ACT-Engine zugreifen. In der Regelsprache sind Ausdrücke nur in bestimmten Kontexten, oder Regelsprachelementen, gültig.

Element 'whenLoaded'

Das Element <whenLoaded> gibt an, dass eine Regel aktiviert wird, wenn sie von der ACT-Engine geladen wird.

Attribute

<whenLoaded> hat keine Attribute.

Enthalten in

<whenLoaded> ist im folgenden Element enthalten:

- <start>

Enthält

<whenLoaded> enthält keine Elemente.

Kapitel 6. Glossar

Dieses Glossar enthält die Begriffe und Definitionen für wichtige Konzepte in Active Correlation Technology.

ACT Siehe Active Correlation Technology.

ACT-Compiler (Active Correlation Technology Compiler)
Die Komponente von Active Correlation Technology, die einen Regelsatz und darin enthaltenen Code syntaktisch analysiert, um die internen Datenstrukturen zu generieren, die für die ACT-Engine erforderlich sind.

ACT-Engine (Active Correlation Technology Engine)
Die Komponente von Active Correlation Technology, die Ereignisse entsprechend der Ausgabe des ACT-Compilers verarbeitet.

Active Correlation Technology
Eine IBM Technologie, die eine Ereigniskorrelation durch Regeln zur Verfügung stellt.

ACT-Laufzeitumgebung (Active Correlation Technology Runtime Environment)
Eine Anwendung, in der die ACT-Engine mit oder ohne Compiler eingebettet ist.

ACT-Regelerstellungsprogramm (Active Correlation Technology Rule Builder)
Eine GUI für das Schreiben von Korrelationsregeln in der ACT-Regelsprache.

ACT-Regelsprache (Active Correlation Technology Rule Language)
Eine XML-basierte Sprache für das Schreiben von Regeln zur Korrelation von Ereignissen. Diese Regeln können anschließend in einer ACT-Laufzeitumgebung eingesetzt werden.

Aktion (Action)
Ein Ausdruck, der als Teil einer Regelantwort oder beim Laden, Entladen, Aktivieren oder Inaktivieren einer Regel ausgeführt wird.

Antwort (Response)
Siehe Regelantwort.

Ausdruck (Expression)
Code, der angepasste Logik enthält, die einer Regel hinzugefügt werden kann. Autoren von Regeln können Ausdrücke für verschiedene Zwecke verwenden, wie z. B. die Initialisierung von Variablen, die Definition von Ereignisauswahlkriterien oder die Spezifikation von Regelantwortaktionen und Lebenszyklusaktionen.

Ausdruckssprache (Expression Language)
Die Programmiersprache, in der ein Ausdruck geschrieben wird.

Berechnungsmuster (Computation Pattern)
Ein Regelmuster, das eine Regel definiert, die eine Berechnung (durch einen Ausdruck) auf erfasste Ereignisse anwendet, während die einzelnen Ereignisse innerhalb eines Zeitintervalls empfangen werden. Eine Regel, die durch das Berechnungsmuster definiert ist, ist eine statusabhängige Regel.

Domäne (Domain)
Die Kategorie, für die eine Gruppe von Regeln basierend auf der Funktion

der Regeln gilt. Eine Domäne kann z. B. ein bestimmtes geographisches Gebiet, einen IT-Managementfachbereich (wie z. B. Sicherheitserkennung oder Netzereigniskorrelation) oder eine Geschäftsorganisation (wie z. B. ein bestimmtes Unternehmen oder eine Abteilung in einem Unternehmen) darstellen.

Duplikatmuster (Duplicate Pattern)
Ein Regelmuster, das eine Regel definiert, die das zweite Ereignis und nachfolgende Ereignisse zählt, die innerhalb des angegebenen Zeitintervalls akzeptiert werden, und die die Regelsatzverarbeitung für diese Ereignisse überspringt. Eine Regel, die durch das Duplikatmuster definiert ist, ist eine statusabhängige Regel.

Ereignisquelle (Event Provider)
Jede Software, die Ereignisse generiert, die von Active Correlation Technology verarbeitet werden.

Ereignisselektor (Event Selector)
Die Bedingungen für die Ereignisauswahl. Diese Bedingungen bestimmen, welche Ereignisse für die Verarbeitung durch eine Regel akzeptiert werden. Der Ereignisselektor umfasst den Ereignistyp und das Filterprädikat.

Erfassungsmuster (Collection Pattern)
Ein Regelmuster, das eine Regel definiert, die eine Gruppe von ausgewählten Ereignissen innerhalb eines Zeitintervalls erfasst. Eine Regel, die durch das Erfassungsmuster definiert ist, ist eine statusabhängige Regel.

Externes Ereignis (External Event)
Ein Ereignis, das die ACT-Engine von einer externen Quelle empfängt.

Externes Objekt (External Object)
Ein Objekt, das von einer Anwendung erstellt wird, um mit Ausdrücken zu kommunizieren.

Filtermuster (Filter Pattern)
Ein Regelmuster, das eine Regel definiert, die eine bestimmte Aktion ausführt, wenn sie ein Ereignis akzeptiert. Eine Regel, die durch das Filtermuster definiert wird, wird nur für ein einzelnes Ereignis ausgeführt und ist somit eine statusunabhängige Regel.

Filterprädikat (Filtering Predicate)
Ein Ausdruck, der die Bedingung definiert, unter der ein Ereignis für die Verarbeitung durch eine Regel akzeptiert wird. Das Filterprädikat ist Teil eines Ereignisselektors. Ein Filterprädikat gibt einen Booleschen Wert zurück.

Gruppierungsschlüssel (Grouping Key)
Eine Methode, die Regel anzuweisen, eine separate Regelinstanz (oder Kopie) für jede Gruppe von Ereignissen zu erstellen, die allgemeine Merkmale gemeinsam verwenden.

Import
Eine programmiersprachenspezifische Möglichkeit, externen Code für Ausdrücke zugänglich zu machen.

Internes Ereignis (Internal Event)
Ein Ereignis, das von einer Regel erstellt wird, die in der ACT-Engine ausgeführt wird. Dieses Ereignis kann an andere Regeln weitergeleitet werden.

Knoten (Node)
Ein Objekt innerhalb der Regelhierarchie, das innerhalb eines Regelsatz individuell und unabhängig hinzugefügt, entfernt oder ersetzt werden kann. Insbesondere die folgenden Objekte sind Knoten:

- Regeln
- Regelblöcke
- Regelblockvariablen
- Regelsatzvariablen

Weil für ein Objekt keine individuelle und unabhängige Verarbeitung unterhalb der Regelebene ausgeführt werden kann, ist eine Regelvariable kein Knoten.

Lebenszyklusaktion (Life Cycle Action)
Ein Ausdruck, der beim Laden, Entladen, Aktivieren oder Inaktivieren einer Regel ausgeführt wird.

Prädikat (Predicate)
Siehe Filterprädikat.

Regelantwortaktion (Rule Response Action)
Siehe Aktion.

Regelantwort (Rule Response)
Ein Ausdruck, der ausgeführt wird, wenn die ACT-Engine erkennt, dass eine Regelbedingung erfüllt wurde. Eine Regelantwort besteht aus mindestens einer Aktion.

Regelblock (Rule Block)
Die Organisationseinheit für die Gruppierung von Regeln nach Funktion in Domänen innerhalb des Regelsatzes. Ein Regelblock kann nicht nur Regeln, sondern auch andere Regelblöcke enthalten.

Regelinstanz (Rule Instance)
Die Kopie einer Regel im Kontext des Gruppierungsschlüssels.

Regelmuster (Rule Pattern)
Die Darstellung einer Ereigniskorrelationssituation (beispielsweise eine Schwellenwertbedingung oder eine Erkennung von duplizierten Ereignissen). Die ACT-Regelsprache umfasst die folgenden Regelmuster: Erfassungsmuster, Berechnungsmuster, Duplikatmuster, Filtermuster, Sequenzmuster, Schwellenwertmuster und Zeitgebermuster. Das Muster einer Regel stimmt überein, wenn die durch die Regel definierte Situation eintritt. Wenn das Muster übereinstimmt, schließt die Regel ihre Verarbeitung ab, indem Sie die entsprechenden Regelantwortaktionen ausführt. Während eine Regel aktiv ist, kann das Regelmuster mehrfach abgeglichen werden.

Regel (Rule)
Die Korrelationseinheit, die zur Erkennung von Beziehungen zwischen Ereignissen und zur Ausführung der geeigneten Regelantworten verwendet wird. Eine Regel ist eine Implementierung eines von sieben Regelmustern und befindet sich entsprechend ihrer Funktion in einem Regelblock, der Teil eines Regelsatzes ist. Eine Regel akzeptiert ein Ereignis für die Verarbeitung, wenn das Ereignis die Ereignisauswahlkriterien erfüllt.

Regelsatz (Rule Set)
Die Regelausführungseinheit für die ACT-Regelsprache. Der Regelsatz enthält die in Regelblöcken organisierten Regeln, die von der ACT-Engine

ausgeführt werden sollen. Die ACT-Engine wird zu einer vorgegebenen Zeit nur für jeweils einen Regelsatz ausgeführt.

Schwellenwertmuster (Threshold Pattern)
Ein Regelmuster, das eine Regel definiert, die eine Gruppe von ausgewählten Ereignissen innerhalb eines Zeitintervalls erfasst und jeweils nach dem Empfang der einzelnen Ereignisse bestimmt, ob eine Schwellenwertbedingung erfüllt wurde. Eine Regel, die durch das Schwellenwertmuster definiert ist, ist eine statusabhängige Regel.

Sequenzmuster (Sequence Pattern)
Ein Regelmuster, das eine Regel definiert, die das Vorhandensein oder Nichtvorhandensein einer bestimmten Reihenfolge von Ereignissen innerhalb eines Zeitintervalls erkennt. Die Reihenfolge kann sortiert oder zufällig sein. Eine Regel, die durch das Sequenzmuster definiert ist, ist eine statusabhängige Regel.

Snippet
Ein Auszug des Quellcodes.

Statusabhängige Regel (Stateful Rule)
Eine Regel, die Statusinformationen festhält, die Informationen zu Merkmalen einer Regelinstanz darstellen, um für eine Gruppe von Ereignissen über einen Zeitraum ausgeführt zu werden. Regeln, die durch eines der folgenden Regelmuster definiert sind, sind statusabhängige Regeln: Erfassungsmuster, Berechnungsmuster, Duplikatmuster, Sequenzmuster, Schwellenwertmuster oder Zeitgebermuster.

Statusunabhängige Regel (Stateless Rule)
Eine Regel, die keine Statusinformationen festhält und somit nur für jeweils ein Ereignis gleichzeitig ausgeführt werden kann. Eine Regel, die durch das Filtermuster definiert ist, ist eine statusunabhängige Regel.

Zeitgebermuster (Timer Pattern)
Ein Regelmuster, das eine Regel definiert, die Aktionen zu regelmäßigen Intervallen einleitet. Eine Regel, die durch das Zeitgebermuster definiert ist, ist eine statusabhängige Regel. Obwohl eine Zeitgeberregel keine Ereignisse verarbeitet, kann sie durch ein Ereignis aktiviert oder inaktiviert werden.

Anhang. Bemerkungen

Die vorliegenden Informationen wurden für Produkte und Services entwickelt, die auf dem deutschen Markt angeboten werden. Möglicherweise bietet IBM die in dieser Dokumentation beschriebenen Produkte, Services oder Funktionen in anderen Ländern nicht an. Informationen über die gegenwärtig im jeweiligen Land verfügbaren Produkte und Services sind beim IBM Ansprechpartner erhältlich. Hinweise auf IBM Lizenzprogramme oder andere IBM Produkte bedeuten nicht, dass nur Programme, Produkte oder Services von IBM verwendet werden können. An Stelle der IBM Produkte, Programme oder Services können auch andere ihnen äquivalente Produkte, Programme oder Services verwendet werden, solange diese keine gewerblichen oder anderen Schutzrechte von IBM verletzen. Die Verantwortung für den Betrieb von Fremdprodukten, Fremdprogrammen und Fremdservices liegt beim Kunden.

Für in diesem Handbuch beschriebene Erzeugnisse und Verfahren kann es IBM Patente oder Patentanmeldungen geben. Mit der Auslieferung dieses Handbuchs ist keine Lizenzierung dieser Patente verbunden. Lizenzanforderungen sind schriftlich an folgende Adresse zu richten (Anfragen an diese Adresse müssen auf Englisch formuliert werden):

IBM Europe
Director of Licensing
92066 Paris La Defense Cedex
France

Trotz sorgfältiger Bearbeitung können technische Ungenauigkeiten oder Druckfehler in dieser Veröffentlichung nicht ausgeschlossen werden. Die Angaben in diesem Handbuch werden in regelmäßigen Zeitabständen aktualisiert. Die Änderungen werden in Überarbeitungen oder in Technical News Letters (TNLs) bekannt gegeben. IBM kann ohne weitere Mitteilung jederzeit Verbesserungen und/oder Änderungen an den in dieser Veröffentlichung beschriebenen Produkten und/oder Programmen vornehmen.

Verweise in diesen Informationen auf Websites anderer Anbieter dienen lediglich als Benutzerinformationen und stellen keinerlei Billigung des Inhalts dieser Websites dar. Das über diese Websites verfügbare Material ist nicht Bestandteil des Materials für dieses IBM Produkt. Die Verwendung dieser Websites geschieht auf eigene Verantwortung.

Werden an IBM Informationen eingesandt, können diese beliebig verwendet werden, ohne dass eine Verpflichtung gegenüber dem Einsender entsteht.

Lizenznehmer des Programms, die Informationen zu diesem Produkt wünschen mit der Zielsetzung: (i) den Austausch von Informationen zwischen unabhängigen, erstellten Programmen und anderen Programmen (einschließlich des vorliegenden Programms) sowie (ii) die gemeinsame Nutzung der ausgetauschten Informationen zu ermöglichen, wenden sich an folgende Adresse:

Director of Licensing
2Z4A/101
11400 Burnet Road
Austin, TX 78758 U.S.A.

Die Bereitstellung dieser Informationen kann unter Umständen von bestimmten Bedingungen - in einigen Fällen auch von der Zahlung einer Gebühr - abhängig sein.

Die Lieferung des im Dokument aufgeführten Lizenzprogramms sowie des zugehörigen Lizenzmaterials erfolgt auf der Basis der IBM Rahmenvereinbarung sowie der Allgemeinen Geschäftsbedingungen von IBM, der IBM Internationalen Nutzungsbedingungen für Programmpakete oder einer äquivalenten Vereinbarung.

Marken

DB2, IBM, das IBM Logo, Tivoli, das Tivoli-Logo, Tivoli Enterprise Console und WebSphere sind in gewissen Ländern Marken oder eingetragene Marken der International Business Machines Corporation.

Java und alle Java-basierten Marken und Logos sind in gewissen Ländern Marken oder eingetragene Marken von Sun Microsystems, Inc.

Andere Namen von Unternehmen, Produkten und Services können Marken oder Servicemarken anderer Unternehmen sein.

Gedruckt in Deutschland