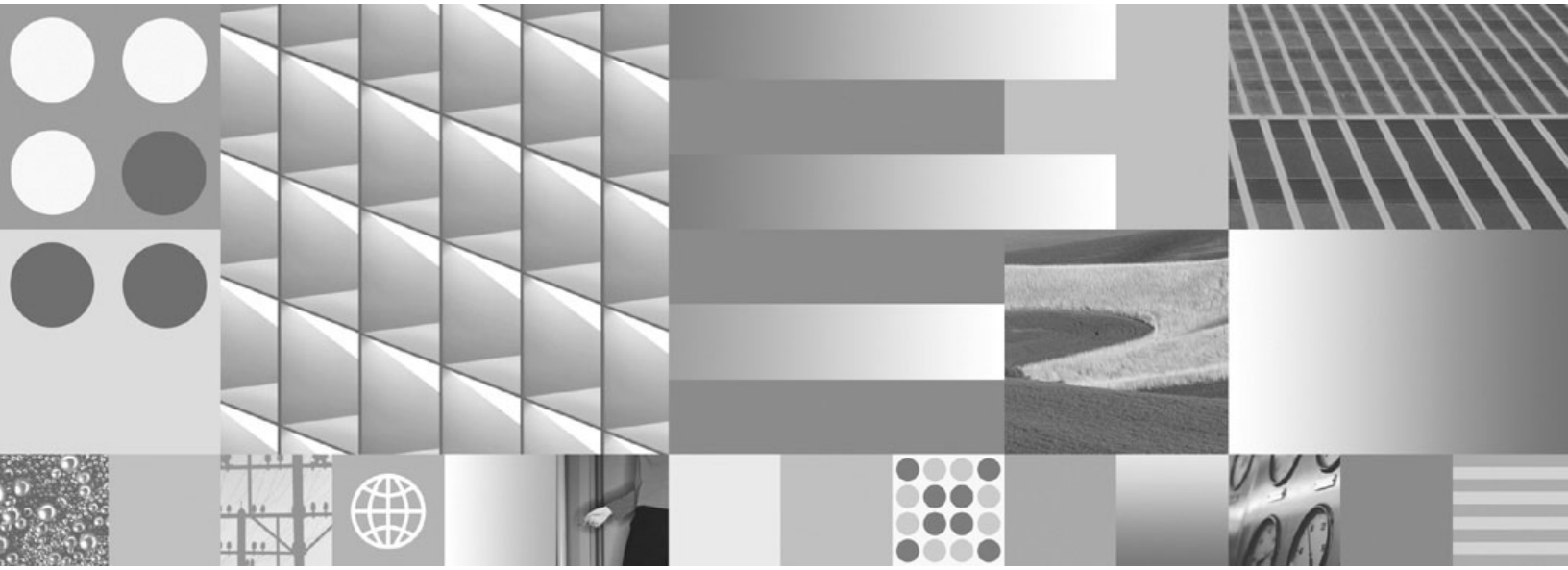
Active Correlation Technology

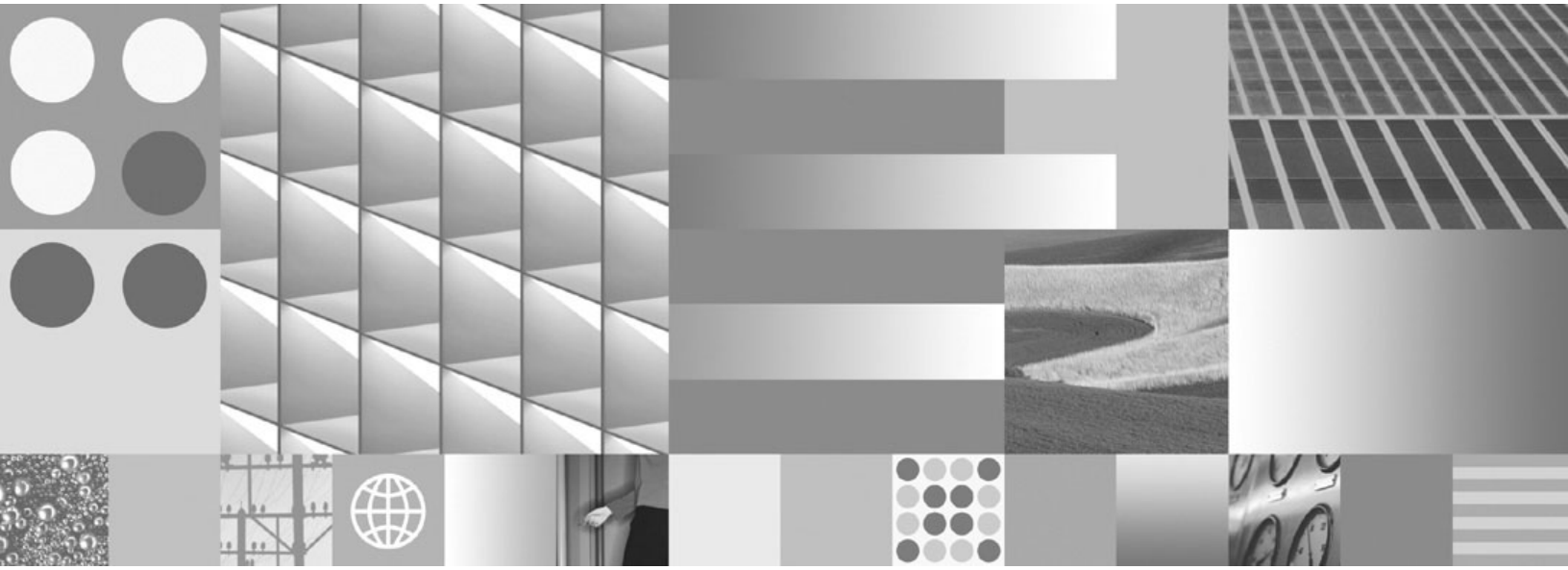
Version 6.0



Guide et document de référence pour le rédacteur de règle

Active Correlation Technology

Version 6.0



Guide et document de référence pour le rédacteur de règle

Remarque

Avant d'utiliser les informations suivantes et le produit qu'elles documentent, prenez connaissance des informations contenues dans «Remarques», à la page 127.

Première édition (juin 2006)

A propos de ces informations

Ces informations donnent une présentation générale de la technologie IBM Active Correlation Technology et de son rôle dans le traitement d'événements complexes. Le langage de règle Active Correlation Technology est un langage XML destiné à l'écriture de règles de corrélation d'événements. Ces informations sont destinées aux rédacteurs de règle qui doivent comprendre comment rédiger des règles pour corréler des événements dans le cadre de leur organisation.

Table des matières

Partie 1. Guide du rédacteur de règle

Chapitre 1. Introduction

Cette introduction donne une description succincte du traitement d'événement complexe (également connu sous le nom de CEP) et présente la technologie Active Correlation Technology (ACT) et son rôle dans ce type de traitement.

L'environnement métier aujourd'hui

Aujourd'hui, les organisations commerciales et gouvernementales sont dépendantes du traitement d'informations électroniques par le biais des réseaux informatiques, et plus particulièrement d'Internet. Avec les nouvelles technologies, telles que l'informatique distribuée, les organisations exécutent des applications vitales à toute heure et en tout lieu dans le monde. Les processus, activité et infrastructure métier, et donc notre société au niveau mondial, sont dépendants de la couche technologie de l'information (IT) des organisations.

Ces organisations ont besoin de connaître et comprendre à tout moment l'évolution de leurs affaires. Par exemple, elles ont besoin de savoir si les applications vitales sont disponibles et fonctionnent correctement, et comment détecter et éviter une crise potentielle dans les processus, activité ou infrastructure métier. Si une crise se produit, elles doivent immédiatement comprendre le problème, savoir comment le résoudre et d'où il provient.

L'importance de la plupart des événements concernant les processus, activité et infrastructure métier n'est jamais reconnue ou comprise du fait de la quantité d'informations trop importante et trop difficile à traiter, ces informations existant sous forme d'éléments individuels et non liés entre eux. Cependant, si les événements sont agrégés et corrélés de manière à faciliter la compréhension de leurs relations, ils peuvent fournir une mine d'informations.

L'objectif du traitement d'événement complexe est d'obtenir de meilleures informations sur les événements en temps réel.

Traitement d'événement complexe

Un *événement* est une simple notification à propos de ce qui a eu lieu.

Le *traitement d'événement complexe* consiste à faire dériver des événements généraux à partir de l'analyse, de la corrélation et du regroupement d'événements particuliers dans des systèmes commandés par les événements. Ces événements généraux, appelés événements complexes, sont adaptés pour notifier aux entreprises des opportunités commerciales ou des incidents dans des termes facilement compréhensibles, ou pour déclencher des processus automatisés. Les organisations peuvent alors fonctionner de manière plus efficace, grâce à la notification préalable de potentiels incidents ou opportunités, ainsi qu'à une meilleure compréhension des causes profondes qui modifient les conditions dans leurs processus, activité ou infrastructure métier.

La *corrélation d'événements* est un processus de définition et détection de modèles dans des flots d'événements en temps réel, et d'implémentation d'actions en réponse à des événements associés. Elle permet d'identifier un problème en fonction de ses symptômes détectés. Les événements peuvent être corrélés par cause, par heure, par appartenance ou par la combinaison des trois. La corrélation

d'événements fait partie intégrante du traitement d'événement complexe.

Technologie ACT

La technologie ACT utilise des règles pour détecter des modèles dans des flots d'événements en temps réel. Cette technologie repose sur la prise de conscience suivante : dans de nombreux cas, les actions de réponse ne doivent pas être déclenchées par un seul événement particulier, mais plutôt par une combinaison complexe d'événements ayant lieu à des moments différents dans des contextes différents. La technologie ACT exploite donc les relations entre les événements pour permettre la détection d'opportunités commerciales et d'incidents. A partir des opportunités détectées grâce à la corrélation d'événements en temps réel, une organisation peut par exemple effectuer les types d'action suivants :

- Offrir une remise sur l'expédition pour certains ou tous les clients au moment des soldes.
- Au cours des 30 jours suivants, calculer les frais de port en fonction du transporteur, du montant de la commande et de la quantité commandée.
- Envoyer un chèque-cadeau de 25 euros aux clients qui achètent des biens d'une valeur de plus de 500 euros entre le 1er juillet 2005 et le 31 décembre 2005.
- Notifier à un administrateur qu'une commande n'a pas fini d'être traitée dans un délai de 36 heures.
- Notifier à un administrateur qu'il y a eu plus de quatre tentatives de connexion sur un même ordinateur en l'espace de 30 secondes.

La technologie ACT comprend les éléments principaux suivants :

Langage de règle Active Correlation Technology
: Langage XML destiné à l'écriture de règles de corrélation d'événement. Ces règles peuvent ensuite être déployées dans les environnements d'exécution Active Correlation Technology.

Moteur Active Correlation Technology
: Composant Active Correlation Technology qui traite les événements selon la sortie du compilateur Active Correlation Technology.

Générateur de règles Active Correlation Technology
: Interface graphique permettant d'écrire des règles de corrélation dans le langage de règle Active Correlation Technology.

Un environnement d'exécution Active Correlation Technology est une application dans laquelle le moteur Active Correlation Technology est intégré.

Chapitre 2. Présentation du langage de règle

Cette présentation décrit les concepts clé du langage de règle Active Correlation Technology.

Un modèle de règle représente une situation de corrélation d'événements (telle qu'une condition de seuil ou la détection d'un événement en double). Le langage de règle Active Correlation Technology comprend sept modèles de règle qui, selon les études, représentent la plupart des situations de corrélation d'événements que les clients d'IBM doivent gérer. Six d'entre eux définissent des règles avec état, et le septième une règle sans état.

Les règles avec état corrèlent plusieurs événements produits au cours d'une période spécifique et génèrent une réponse à ces événements. Les règles sans état traitent un seul événement qui répond à une certaine condition et génèrent une réponse à cet événement.

règle avec état
: Règle qui conserve des informations d'état, sur les caractéristiques d'une instance de règle, afin d'agir sur un ensemble d'événements au cours d'une période donnée. Les règles définies par l'un des modèles suivants sont des règles avec état : collecte, calcul, duplication, séquence, seuil et temporisateur.

règle sans état
: Règle qui ne conserve pas d'informations d'état, et qui par conséquent ne peut agir que sur un événement à la fois. Une règle définie par le modèle de filtrage est une règle sans état.

Références associées

Chapitre 4, «Récapitulatif relatif à l'organisation du jeu de règles», à la page 47
Ce document répertorie tous les éléments de langage d'un jeu de règles, d'un bloc de règles et de chaque type de règle. Il fait office de référence pouvant être facilement consultée en cas d'interrogation sur le codage d'un jeu de règles.

Chapitre 5, «Guide des éléments de langage», à la page 59
Ce guide décrit en détail les éléments de langage du schéma XML pour le langage de règle Active Correlation Technology. Les éléments de langage sont répertoriés dans l'ordre alphabétique et les attributs disponibles pour chaque élément sont décrits dans la rubrique relative à l'élément concerné.

Chapitre 6, «Glossaire», à la page 123
Ce glossaire contient les termes et définitions de concepts importants dans Active Correlation Technology.

Morphologie d'une règle

Les parties les plus élémentaires d'une règle sont la sélection d'événement, la clé de groupement, la plage temporelle pour les règles avec état, la réponse à la règle, l'intervalle d'activation et les actions de cycle de vie. Une règle contient également des expressions et des variables. Une expression est un code qui comporte une logique personnalisée pouvant être ajoutée à une règle.

Sélection d'événement

Les critères de sélection d'événement déterminent les événements autorisés à être traités par la règle. L'élément <eventSelector> définit ces critères pour une règle. La sélection d'événement s'applique à toutes les règles, excepté celles qui sont définies par le modèle de temporisateur. La règle temporisée ne traitant pas d'événement, elle ne contient aucun critère de sélection d'événement.

Clé de groupement

Généralement, chaque règle active possède une instance de règle, ou copie, exécutée dans le moteur Active Correlation Technology. Cependant, une même règle est parfois nécessaire pour différents groupes d'événements souvent liés à différents groupes de ressources. La clé de groupement permet de conduire une règle à créer une instance de règle (ou une copie) pour chaque groupe d'événements possédant des caractéristiques communes.

Elle constitue une forme supplémentaire de sélection d'événement. Si une règle comportant une clé de groupement reçoit un événement qui possède la caractéristique définie par cette clé, cet événement est dirigé vers l'instance de règle correspondante. Par exemple, vous pouvez définir une règle qui collecte tous les événements de sécurité de type `Audit Failure` (Echec de l'audit) et spécifier que la clé de groupement doit être l'attribut de nom d'hôte d'un événement. Cette règle peut alors être utilisée plusieurs fois, chacune de ses copies étant exécutée pour chaque valeur unique de l'attribut de nom d'hôte. Vous pouvez également surveiller tous les systèmes qui reçoivent l'événement `Audit Failure` pour savoir si plus de 10 événements de ce type se produisent au cours d'un intervalle de 2 minutes pour chaque nom d'hôte.

L'élément <groupingKey> définit la clé de groupement d'une règle, et est valide dans les règles définies par les modèles de collecte, de calcul, de duplication, de séquence et de seuil.

Plage temporelle pour les règles avec état

Les règles avec état corrèlent plusieurs événements produits au cours d'une certaine période. La plage temporelle, définie par l'élément <timeWindow>, constitue ainsi une partie élémentaire d'une règle avec état. Elle spécifie la période au cours de laquelle la règle avec état effectue des traitements pour correspondre à son modèle.

Réponse à la règle

Les actions de réponse à la règle définissent les actions à appliquer lorsque la règle a achevé ses opérations de traitement. Chaque élément de langage suivant définit un type différent d'action de réponse à la règle :

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

Les types d'actions de réponse à la règle disponibles pour une règle dépendent du modèle de règle.

Intervalle d'activation

L'intervalle d'activation définit le moment où une règle est active ou inactive. L'élément <activationInterval> définit l'intervalle d'activation d'une règle.

Une règle peut être activée ou désactivée à un moment précis dans le temps ou sous l'effet d'un événement spécifique.

Si vous spécifiez qu'une règle doit être activée ou désactivée à un moment précis dans le temps *et* sous l'effet d'un événement spécifique, cette règle sera alors activée ou désactivée par ce qui survient en premier, à savoir le moment précis dans le temps ou la réception de l'événement. Toutefois, la règle peut dans ce cas être activée ou désactivée par une multitude d'événements tout au long de son cycle de vie. Par exemple, elle peut être activée par un événement, désactivée, activée à un moment défini dans le temps, désactivée à nouveau, puis activée par un autre événement.

L'élément <activationByGroupingKey> est contenu dans l'élément <activationInterval>. Il contient des éléments qui spécifient les événements pouvant activer et désactiver une instance de règle définie par <groupingKey>.

Actions de cycle de vie

Les actions de cycle de vie définissent les actions à appliquer lors de ces quatre étapes principales du cycle de vie d'une règle : chargement, activation, désactivation et déchargement.

<lifeCycleActions> contient les éléments suivants, qui définissent ces actions :

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

Cycle de vie d'une règle

Chaque étape du cycle de vie d'une règle peut comporter plusieurs causes et effets. En écrivant et en incluant des expressions dans les actions de cycle de vie (définies par l'élément <lifeCycleActions>), un rédacteur de règle peut définir les actions à appliquer à chaque étape.

Etapes du cycle de vie d'une règle

Les quatre étapes principales du cycle de vie d'une règle sont les suivantes :

Chargement
: Chargement de la règle dans le moteur en cours d'exécution d'Active Correlation Technology, ayant pour effet de déclencher les actions contenues dans l'élément <onLoad>.

Activation
: Activation de la règle, ayant pour effet de déclencher les actions contenues dans l'élément <onActivation>.

Désactivation
: Désactivation de la règle, ayant pour effet de déclencher les actions contenues dans l'élément <onDeactivation>.

Déchargement
Déchargement de la règle depuis le moteur Active Correlation Technology en cours d'exécution, ayant pour effet de déclencher les actions contenues dans l'élément <onUnload>.

Dans le cycle de vie d'une règle, les étapes d'activation et de désactivation peuvent se produire plusieurs fois, tandis que les étapes de chargement et de déchargement n'ont lieu qu'une seule fois.

Il n'est généralement pas nécessaire de définir des actions de cycle de vie. La liste suivante énumère des exemples de situations dans lesquelles vous pouvez être amené à définir une action de cycle de vie particulière :

- Suite au chargement d'une règle donnée, vous pouvez être amené à créer une connexion vers un système externe (tel qu'un gestionnaire de base de données) auquel l'accès doit être effectué depuis cette règle. Après déchargement de la règle, vous souhaiterez supprimer la connexion et exécuter les processus de nettoyage nécessaires.
- Suite à l'activation d'une règle donnée, vous pouvez être amené à vérifier que certaines ressources sont disponibles pour cette règle.
- Lorsqu'une règle de seuil est désactivée sans que le seuil déterminé soit atteint et la période spécifiée achevée, vous pouvez être amené à acheminer vers une personne un message pour l'en informer.

L'activation et la désactivation d'une règle pouvant se produire plusieurs fois au cours du cycle de vie, toutes les actions que vous codez pour ces étapes peuvent s'effectuer fréquemment.

Causes et effets de chaque étape du cycle de vie

Le tableau tableau 1 répertorie les causes et effets de chaque étape du cycle de vie.

Tableau 1. Causes et effets de chaque étape du cycle de vie

Etape du cycle de vie	**Causes**	**Effets**
Chargement	L'une des circonstances suivantes : • Une règle ou un bloc de règles est ajouté ou remplacé, ce qui provoque le chargement de la ou des nouvelles règles. • Le jeu de règles est remplacé dans le moteur Active Correlation Technology, ce qui provoque le chargement des règles du nouveau jeu de règles.	Les actions contenues dans l'élément <onLoad> sont exécutées.
Activation	La règle est activée. Une règle peut être activée de l'une des manières suivantes : • Conformément aux définitions de l'élément <activationInterval> • A l'aide de la méthode activate(), disponible depuis la variable act_lib • Par des appels d'application de la méthode activate() dans le moteur Active Correlation Technology	Si la règle est inactive, les actions contenues dans l'élément <onActivation> sont exécutées.

Tableau 1. Causes et effets de chaque étape du cycle de vie (suite)

Etape du cycle de vie	Causes	Effets
Désactivation	La règle est désactivée. Une règle peut être désactivée de l'une des manières suivantes : • Conformément aux définitions de l'élément <activationInterval>, mais l'élément <deactivateOnEvent> contenu dans <activationByGroupingKey> ne provoque pas la désactivation de la règle • A l'aide de la méthode deactivate(), disponible depuis la variable act_lib • Par des appels d'application de la méthode deactivate() dans le moteur Active Correlation Technology	Si la règle est active, les actions contenues dans l'élément <onDeactivation> sont exécutées.
Déchargement	L'une des circonstances suivantes : • Le moteur Active Correlation Technology s'arrête, ce qui provoque le déchargement des règles. • Une règle ou un bloc de règles est supprimé ou remplacé, ce qui provoque le déchargement de la ou des anciennes règles. • Le jeu de règles est supprimé ou remplacé dans le moteur Active Correlation Technology, ce qui provoque le déchargement des règles de l'ancien jeu de règles.	Si la règle est active, les actions contenues dans l'élément <onDeactivation> sont exécutées, suivies des actions contenues dans l'élément <onUnload>. Sinon, seules les actions contenues dans l'élément <onUnload> sont exécutées.

Organisation des règles

Le langage de règle Active Correlation Technology organise les règles en blocs de règles qui font partie d'un jeu de règles.

Jeu de règles

Le jeu de règles contient des règles, organisées dans des blocs de règles, qui doivent être exécutées par un moteur Active Correlation Technology. Il s'agit d'une unité d'exécution de règles. Chaque moteur Active Correlation Technology opère sur un seul jeu de règles à la fois.

Les règles contenues dans un jeu de règles sont déclenchées par des événements envoyés au moteur Active Correlation Technology. Ces événements sont acheminés de manière séquentielle vers les règles correspondantes en fonction des critères de sélection d'événement de chaque règle ; une seule règle est exécutée à la fois. Le même événement peut s'appliquer à, et donc déclencher, de multiples règles. Ces règles ne sont pas nécessairement liées entre elles, mais elles sont susceptibles de l'être.

L'ordre des blocs de règles et des règles au sein d'un jeu de règles détermine le flux des événements dans ce jeu.

Les variables et les importations peuvent être définies au niveau du jeu de règles pour être utilisées dans des expressions (code contenant une logique personnalisée)

sur toute la portée du jeu de règles. Une importation est un moyen spécifique aux langages de programmation d'accéder aux codes externes. Un rédacteur de règle peut définir une importation pour importer des modules externes (tels que des classes Java) en vue de les utiliser dans des expressions contenues dans des règles.

Bloc de règles

Le bloc de règles est l'unité organisationnelle des règles de groupement par fonction dans des domaines au sein du jeu de règles. Un domaine est la catégorie à laquelle s'applique un groupe de règles selon sa fonction. Par exemple, un domaine peut représenter une zone géographique particulière, une discipline de gestion informatique (telles que la détection de sécurité ou la corrélation d'événements de réseau) ou une organisation métier (une entreprise particulière ou un service dans une entreprise).

Les blocs de règles peuvent contenir des règles et d'autres blocs de règles. Etant donné que les blocs de règles peuvent être imbriqués, il est possible de construire une hiérarchie de règles. Par exemple, un jeu de règles peut contenir un bloc de règles pour la corrélation d'événements réseaux et ce dernier peut contenir deux autres blocs de règles : un pour la corrélation de la couche 2 et un pour la corrélation du protocole IP.

Un jeu de règles offre donc des fonctions de corrélation d'événements pour une variété de domaines, et un bloc de règles prévoit l'organisation de ces différents domaines pouvant avoir besoin d'accéder à un jeu de règles semblable.

Les variables et les importations peuvent être définies au niveau du bloc de règles pour être utilisées dans des expressions sur toute la portée du bloc de règles. La portée d'un bloc de règles inclut toutes les règles et autres blocs de règles contenus dans le bloc de règles.

Règle

La règle constitue l'unité de corrélation utilisée pour reconnaître des relations parmi des événements et pour exécuter les réponses à la règle appropriées. Une règle est une implémentation d'un des sept modèles de règle suivants et est organisée, selon sa fonction, dans un bloc de règles faisant partie d'un jeu de règles :

- modèle de collecte
- modèle de calcul
- modèle de duplication
- modèle de filtrage
- modèle de séquence
- modèle de seuil
- modèle de temporisateur

Chaque règle peut fournir des fonctions de corrélation d'événements selon son modèle et les règles peuvent être enchaînées par transmission des événements. Par ce chaînage de règles, les fonctions de corrélation d'événements des différents modèles peuvent être regroupées ou imbriquées.

Les variables peuvent être définies au niveau de la règle pour être utilisées dans des expressions sur toute la portée de la règle.

Récapitulatif

Pour résumer, le jeu de règles est l'unité d'exécution, le bloc de règles est l'unité organisationnelle et la règle est l'unité de corrélation. Un jeu de règles contient un ou plusieurs blocs de règles dont chacun peut contenir d'autres blocs de règles. Chacun des blocs de règles contient des règles pour un domaine spécifique. Les blocs de règles peuvent être imbriqués pour construire une hiérarchie de règles. L'ordre des blocs de règles et des règles au sein d'un jeu de règles détermine le flux des événements dans ce jeu.

Les variables et les importations peuvent être définies au niveau du jeu de règles ou du bloc de règles pour être utilisées dans des expressions au sein des règles. La portée de la variable ou de l'importation est respectivement le jeu de règles ou le bloc de règles. Les variables peuvent aussi être définies au niveau de la règle, mais leur portée est alors limitée à la règle seulement.

Modèles de règle

Un modèle de règle représente une situation de corrélation d'événements (telle qu'une condition de seuil ou la détection d'un événement en double). Le langage de règle Active Correlation Technology définit les modèles de règle suivants : collecte, calcul, duplication, filtrage, séquence, seuil et temporisateur.

Le modèle d'une règle a trouvé une correspondance lorsque la situation définie par la règle se produit. Le cas échéant, la règle conclut ses opérations de traitement en appliquant les actions de réponse à la règle appropriées. Lorsqu'une règle est active, elle peut correspondre plusieurs fois au modèle de règle.

Les règles définies par le modèle de filtrage sont uniquement des règles sans état dans le langage de règle. Toutes les autres règles sont avec état.

Modèle de collecte

Une règle de collecte est définie par le modèle de collecte. Elle collecte un groupe d'événements sélectionnés au cours d'un intervalle de temps. Il s'agit d'une règle avec état.

Présentation

Le modèle de collecte permet de regrouper des événements semblables entre eux au cours d'une période donnée. Cette période est indiquée par une plage temporelle obligatoire définie par l'élément <timeWindow> dans le langage de règle.

Conditions d'exécution de la réponse à la règle

Dans le cas du modèle de collecte, la réponse à la règle est exécutée au terme de la plage temporelle, comme défini par l'élément <onTimeWindowComplete>.

Exemple d'utilisation de ce modèle de règle

Une règle définie par un modèle de collecte peut être utilisée de la manière suivante :

Elle collecte les événements qui répondent aux critères d'un sélecteur d'événement donné au cours de la période indiquée. Au terme de cette

période, elle récapitule les événements collectés en un seul événement contenant le nombre total d'événements et des informations caractéristiques sur les événements récapitulés.

Référence associée

«Récapitulatif relatif à la règle de collecte», à la page 48
Ce récapitulatif répertorie tous les éléments de langage de la règle de collecte.

Modèle de calcul

Une règle de calcul est définie par le modèle de calcul. Elle s'applique à un calcul effectué sur des événements collectés (au moyen d'une expression), à chaque réception d'événement au cours d'un intervalle donné. Il s'agit d'une règle avec état.

Présentation

Le modèle de calcul exécute une fonction de calcul, définie par l'élément <computeFunction> dans le langage de règle, sur chaque événement accepté au cours d'une période. Cette période est indiquée par une plage temporelle obligatoire définie par l'élément <timeWindow>.

Conditions d'exécution de la réponse à la règle

Dans le cas du modèle de calcul, la réponse à la règle est exécutée au terme de la plage temporelle, comme défini par l'élément <onTimeWindowComplete>. La valeur du calcul est disponible au cours de l'action <onTimeWindowComplete>.

Exemple d'utilisation de ce modèle de règle

Supposons qu'une application traite des événements de commandes client. Une règle définie par un modèle de calcul peut être utilisée de la manière suivante :

Chaque fois qu'un événement est reçu, sa valeur est ajoutée à la valeur totale de toutes les commandes ayant eu lieu au cours de la période spécifiée ; la valeur totale mise à jour des commandes est ensuite publiée dans une interface utilisateur.

Référence associée

«Récapitulatif relatif à la règle de calcul», à la page 50
Ce récapitulatif répertorie tous les éléments de langage de la règle de calcul.

Modèle de duplication

Une règle de duplication est définie par le modèle de duplication. Elle compte le deuxième événement et les suivants acceptés dans l'intervalle de temps défini, tout en omettant leur traitement par le jeu de règles. Il s'agit d'une règle avec état.

Présentation

Le modèle de duplication est généralement utilisé pour isoler des événements semblables (en double) au cours d'une période. Un événement en double est semblable, en un sens, à un événement précédent, mais il ne s'agit pas forcément d'une copie exacte de cet événement. Les événements sont des doublons dès qu'ils répondent aux critères de sélection d'événement pour la règle. La période est indiquée par une plage temporelle obligatoire, telle que définie par l'élément <timeWindow> dans le langage de règle.

Conditions d'exécution de la réponse à la règle

Avec le modèle de duplication, la réponse à la règle s'exécute dans les conditions suivantes :

- Lorsque le premier événement est détecté, tel que défini par l'élément <onDetection>.
- Lorsque chaque événement en double est traité, tel que défini par l'élément <onNextEvent>.
- Au terme de la plage temporelle, tel que défini par l'élément <onTimeWindowComplete>.

Le premier événement déclenche l'action <onDetection> même si aucun événement en double n'a été reçu. La raison d'un tel comportement s'explique par le fait que vous souhaiterez peut-être réacheminer le premier événement et ignorer le traitement du jeu de règles des événements en double. Dans ce cas, vous pouvez ajouter une action de réponse à la règle qui réachemine le premier événement lorsque l'action <onDetection> est déclenchée pour la règle.

Le traitement par défaut des événements en double (le deuxième événement et les suivants) consiste à compter un événement en double, tout en omettant son traitement par le jeu de règles. Si vous souhaitez effectuer une action supplémentaire sur un événement en double, vous pouvez définir explicitement une action <onNextEvent>. Par exemple, dans certains cas, l'événement en double représente un événement susceptible d'être déjà journalisé dans une base de données ou un autre référentiel. Dès lors, vous souhaiterez peut-être coder une action <onNextEvent> pour supprimer l'événement en double de ces autres emplacements.

Une action <onTimeWindowComplete> peut être utilisée pour créer un enregistrement récapitulatif de tous les événements incluant le nombre de doublons qui ont été traités.

Exemple d'utilisation de ce modèle de règle

Supposons que le message «Refus de service» continue à être émis par le même type de ressource (un moniteur de sécurité). Ce message indique une éventuelle violation de sécurité. Une règle qui effectuerait les actions suivantes constituerait un bon exemple d'utilisation de règle définie par un modèle de duplication :

> Une fois que le message «Refus de service» s'est affiché depuis le moniteur de sécurité, tous les doublons de cet événement se produisant au cours d'une période de 30 secondes sont comptés, mais ne sont pas envoyés à une console opérateur. De même, à la fin de la période de 30 secondes, la règle génère un événement indiquant le nombre de messages «Refus de service» qui ont été émis au cours de la période.

Référence associée

«Récapitulatif relatif à la règle de duplication», à la page 51
Ce récapitulatif répertorie tous les éléments de langage de la règle de duplication.

Modèle de filtrage

Une règle de filtrage est définie par le modèle de filtrage. Elle applique une certaine action lorsqu'elle accepte un événement. Puisqu'elle agit sur un seul événement, c'est une règle sans état.

Présentation

Le modèle de filtrage permet d'agir sur des événements particuliers qui répondent aux critères de sélection d'événement. Contrairement aux autres modèles de règle, il ne conserve aucune information d'état associée (telle que l'historique des événements passés).

Conditions d'exécution de la réponse à la règle

Dans le cas du modèle de filtrage, la réponse à la règle est exécutée suite à la réception d'un événement qui répond aux critères de sélection, comme défini par l'élément <onDetection>.

Exemple d'utilisation de ce modèle de règle

Une règle définie par un modèle de filtrage peut être utilisée de la manière suivante :

> si un événement Statut du serveur (`ServerStatus`) indique que la charge du serveur (`serverLoad`) est supérieure à 95 %, la règle exécute une action qui envoie un message à un administrateur.

Référence associée

«Récapitulatif relatif à la règle de filtrage», à la page 52
Ce récapitulatif répertorie tous les éléments de langage de la règle de filtrage.

Modèle de séquence

Une règle de séquence est définie par le modèle de séquence. Elle détecte l'arrivée d'une séquence donnée d'événements au cours d'un intervalle de temps. La séquence peut être ordonnée ou aléatoire. Il s'agit d'une règle avec état.

Présentation

Le modèle de séquence recherche une séquence d'événements sur une période et détermine si la séquence est complète ou non. Une séquence incomplète inclut un ou plusieurs des événements dans l'ordre défini, mais pas tous.

La période est indiquée par une plage temporelle obligatoire, telle que définie par l'élément <timeWindow> dans le langage de règle. Chaque événement de la séquence est défini par un élément <eventSelector> distinct dans la règle. La séquence d'événements peut être détectée dans l'un des ordres suivants :

- Dans l'ordre dans lequel les éléments <eventSelector> sont codés pour la règle. Dans ce cas, quand la règle détecte l'événement défini par le premier élément <eventSelector>, la détection de la séquence est lancée. La règle attend ensuite l'événement défini par le deuxième élément <eventSelector>.
- Dans un ordre aléatoire. Dans ce cas, quand la règle détecte l'un des événements définis par les éléments <eventSelector>, la détection de la séquence est lancée. La règle attend ensuite un autre événement défini par les éléments <eventSelector>.

Le modèle de séquence est différent des autres modèles de règle sur plusieurs points ; voici les principaux :

- Il possède plusieurs éléments <eventSelector> permettant de définir les événements acceptés par la règle. Il requiert un minimum de deux éléments <eventSelector>.

- Lorsqu'un événement satisfait les critères définis par l'un des éléments <eventSelector>, cet élément <eventSelector> est par la suite exclu du traitement d'événement dans cette instance de règle.
- L'attribut alias dans l'élément <eventSelector> est valide uniquement dans une règle de séquence ; il nomme de façon unique un événement sélectionné par un sélecteur d'événement donné dans cette règle. Dans une expression au sein d'un prédicat de filtrage ou d'une action, vous pouvez utiliser la variable act_eventList pour accéder à un événement dans une règle de séquence par son nom d'alias.

Conditions d'exécution de la réponse à la règle

Avec le modèle de séquence, la réponse à la règle s'exécute dans les situations suivantes :

- Lorsque la séquence d'événements dans son ensemble est détectée dans la plage temporelle, tel que défini par l'élément <onDetection>.
- Lorsqu'un ou plusieurs événements arrivent au cours de la plage temporelle, mais pas la séquence dans son ensemble, tel que défini par l'élément <onTimeOut>.

  Le modèle de séquence peut permettre de déterminer si une séquence donnée est incomplète. Par exemple, si un événement «system down» (arrêt du système) se produit sans être suivi d'un événement «system up» (reprise du système), le programme d'écriture de règle peut coder une action <onTimeOut> prenant en charge ce type d'événement manquant.

Exemple d'utilisation de ce modèle de règle

Scénario illustrant la détection d'une séquence complète :
Supposons que, dans un environnement informatique, un administrateur souhaite savoir si la valeur de la taille de segment de mémoire DB2 affecte WebSphere Application Server et, si c'est le cas, qu'il souhaite corriger ce problème. Ainsi, si les événements suivants se produisent dans l'ordre qui suit au cours d'une période définie, l'administrateur souhaitera augmenter la valeur de la taille de segment de mémoire DB2 et redémarrer le gestionnaire de base de données :

1. Une exception d'allocation de ressources WebSphere Application Server. Supposons qu'il s'agisse d'un événement de type `WASResourceAllocationException`.
2. Le message d'erreurDB2 qui indique : «Segment de mémoire insuffisant pour le traitement de l'instruction». Supposons qu'il s'agisse d'un événement de type `DB2NotEnoughHeap`.

Pour ce scénario, deux éléments <eventSelector> sont définis dans une règle de séquence et les événements doivent arriver dans l'ordre dans lequel les éléments <eventSelector> sont codés (et non dans un ordre aléatoire). Le premier élément <eventSelector> recherche l'événement `WASResourceAllocationException`, et le deuxième élément <eventSelector> recherche l'événement `DB2NotEnoughHeap`. Posons que les événements suivants sont présentés au système au cours de la plage temporelle définie :

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

Le comportement de la règle est le suivant :

1. Le premier événement, `WASResourceAllocationException`, est accepté. Les critères du premier élément <eventSelector> ayant été satisfaits, le premier élément <eventSelector> est par la suite exclu du traitement d'événement dans cette règle.
2. Le deuxième événement, `DB2BackupStarted`, est ignoré.
3. Le troisième événement, `WASResourceAllocationException`, est ignoré.
4. Le quatrième événement, `WASResourceAllocationException`, est ignoré.
5. Le cinquième événement, `DB2NotEnoughHeap`, est accepté et il termine la séquence. L'action de réponse à la règle <onDetection> s'exécute. Cette action est définie pour augmenter la valeur de la taille de segment de mémoire DB2 et redémarrer le gestionnaire de base de données. La règle revient à son état initial.

   Le premier élément <eventSelector> est à présent inclus dans le futur traitement d'événement par cette règle.

Scénario illustrant la détection d'une séquence incomplète :
Supposons qu'une entreprise souhaite que toutes les commandes client soient prêtes à être livrées dans un délai d'une heure à compter de la réception de la commande et qu'il veuille savoir si c'est le cas ou non.

Pour ce scénario, deux éléments <eventSelector> sont définis dans une règle de séquence et les événements doivent arriver dans l'ordre de codage des éléments <eventSelector> (et non dans un ordre aléatoire). Le premier élément <eventSelector> recherche l'événement `Netsales` avec `operationType=Order`, et le deuxième élément <eventSelector> recherche l'événement `Netsales` avec `operationType=Delivery`. Posons que les événements suivants sont présentés au système au cours de la plage temporelle définie d'une heure :

1. Un événement `Netsales` avec `operationType=Order`
2. Un événement `Netsales` avec `operationType=Order`

Le comportement de la règle est le suivant :

1. Le premier événement est accepté. Les critères du premier élément <eventSelector> ayant été satisfaits, le premier élément <eventSelector> est par la suite exclu du traitement d'événement dans cette règle.
2. Le deuxième événement est ignoré.
3. Comme un événement `Netsales` avec `operationType=Delivery` n'est pas reçu dans la plage temporelle définie, l'action de réponse à la règle <onTimeOut> est exécutée. Cette action est définie pour avertir un responsable de l'entreprise que la commande d'un client n'est pas prête à être livrée dans un délai d'une heure suivant sa réception. La règle revient à son état initial.

   Le premier élément <eventSelector> est à présent inclus dans le futur traitement d'événement par cette règle.

Concepts associés

«Accès aux informations relatives aux événements», à la page 23
Les exemples qui suivent montrent comment procéder pour accéder aux informations relatives aux événements par le biais des variables livrées avec Active Correlation Technology.

Référence associée

«Récapitulatif relatif à la règle de séquence», à la page 53
Ce récapitulatif répertorie tous les éléments de langage de la règle de séquence.

Modèle de seuil

Une règle de seuil est définie par le modèle de seuil. Elle collecte un groupe d'événements sélectionnés au cours d'un intervalle de temps et détermine, une fois que chaque événement a été reçu, si une condition de seuil est remplie. Il s'agit d'une règle avec état.

Présentation

Le modèle de seuil collecte les événements au cours d'un intervalle de temps, jusqu'à ce qu'une valeur de seuil soit atteinte. La période est indiquée par une plage temporelle obligatoire, telle que définie par l'élément <timeWindow> dans le langage de règle.

Le modèle de seuil fournit les trois options suivantes pour un type de seuil :

seuil de comptage d'événement
Grâce à ce type de seuil, vous pouvez définir le nombre d'événements devant répondre aux critères de sélection au cours d'une période donnée. La valeur de seuil définie est comparée au nombre d'événements qui ont été acceptés. Lorsque le comptage d'événements est égal à la limite définie au cours de la plage temporelle, le seuil est atteint.

Ce type de seuil peut être utile pour une vérification très simple du comptage d'événements. Il peut, par exemple, répondre à la question suivante : «Cinq événements d'échec de connexion se sont-ils produits dans la même minute ?»

Ce seuil est défini par l'élément <eventCountThreshold>. L'élément <eventCountThreshold> spécifie également l'un des deux modes d'intervalle de temps possibles suivants pour la plage temporelle :

intervalle fixe
Un intervalle fixe commence à la réception du premier événement qui répond aux critères de sélection, et s'achève dans l'un des cas suivants :

- La règle atteint son seuil au cours de la durée spécifiée.
- La durée spécifiée s'est écoulée.

intervalle glissant
Un intervalle glissant commence à la réception du premier événement qui correspond aux critères de sélection. Cependant, si la règle n'a pas atteint son seuil après écoulement de la durée spécifiée, la plage temporelle réajuste l'heure de début pour la faire glisser jusqu'à l'heure de réception d'un nouveau «premier» événement, qui est généralement l'événement accepté suivant. L'intervalle glissant poursuit ce réajustement jusqu'à que l'une des situations suivantes se produisent :

- La règle atteint son seuil au cours de la durée spécifiée.
- L'événement qui démarre la plage temporelle n'est suivi d'aucun autre événement au cours de la durée spécifiée.

L'événement qui démarre la plage temporelle (et devient le nouveau «premier» événement) est l'événement qui répond à ce critère : son heure de réception, ajoutée à la durée de l'intervalle de temps définie pour la règle, est supérieure à l'heure en cours. Nous obtenons l'équation suivante :

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

S'il n'existe aucun événement de ce type, l'intervalle glissant ne peut plus réajuster l'heure de début et s'achève.

seuil calculé

Grâce à ce type de seuil, vous pouvez rédiger du code (ou utiliser du code écrit par quelqu'un d'autre) qui réalise un calcul sur chaque événement accepté et renvoie une valeur de seuil calculée qui est conservée dans une variable préalablement définie. Cette valeur de seuil calculée est ensuite comparée à une valeur de seuil définie afin d'établir si le seuil a été atteint.

Un calcul complexe peut donc être mis en oeuvre afin de créer (ou mettre à jour) une valeur de seuil calculée, éventuellement à l'aide de données sauvegardées à partir d'événements précédents, et le rédacteur de la règle peut définir la valeur de seuil définie indépendamment de la logique qui calcule la valeur de seuil calculée.

Ce type de seuil peut être utile pour le regroupement et la comparaison d'une valeur par rapport à une valeur de seuil définie. Il peut par exemple permettre de calculer les ventes totales (exprimées en euros) réalisées auprès d'un client donné sur une période donnée, pour ensuite comparer cette somme à une valeur de seuil définie.

Ce seuil est défini par l'élément <computedThreshold>.

seuil booléen

Grâce à ce type de seuil, vous pouvez rédiger du code (ou utiliser du code écrit par quelqu'un d'autre) qui renvoie une valeur `true` ou `false` pour chaque événement accepté. Si cette valeur est `true`, le seuil est atteint. Si cette valeur est `false`, la règle de seuil continue ses opérations de traitement jusqu'à l'expiration de la plage temporelle ou jusqu'à ce qu'elle accepte un autre événement.

Ce type de seuil peut permettre de vérifier une plage de valeurs. Par exemple, si l'utilisation de l'unité centrale doit se situer entre 30 % et 80 % à tout moment, ce seuil peut vérifier en permanence que l'utilisation demeure dans cette plage.

Ce seuil est défini par l'élément <booleanThreshold>.

Conditions d'exécution de la réponse à la règle

Avec le modèle de seuil, la réponse à la règle s'exécute aux moments suivants :

- Lorsque le seuil a été atteint, tel que défini par l'élément <onDetection>.
- Lorsqu'un ou plusieurs événements sont acceptés mais que le seuil n'est pas atteint au cours de la plage temporelle, telle que définie par l'élément <onTimeOut>.

Exemple d'utilisation de ce modèle de règle

Une règle qui effectuerait les opérations suivantes constitue un bon exemple d'utilisation du modèle de seuil avec le seuil de comptage d'événement :

Si plus de quatre (4) événements `Server unreachable` (serveur introuvable) sont émis par le même sous-réseau au cours d'un intervalle de temps glissant de 30 secondes, la règle exécute une vérification de l'état d'un routeur.

Référence associée

«Récapitulatif relatif à la règle de seuil», à la page 55
Ce récapitulatif répertorie tous les éléments de langage de la règle de seuil.

Modèle de temporisateur

Une règle temporisée est définie par le modèle de temporisateur. Elle déclenche des actions à intervalles réguliers. Il s'agit d'une règle avec état. Bien que les règles temporisées ne traitent pas les événements, elles peuvent être activées ou désactivées par un événement.

Présentation

Le modèle de temporisateur est semblable à un temporisateur qui démarre au début d'une période et s'arrête à l'issue de cette période. La période est indiquée par une plage temporelle obligatoire, telle que définie par l'élément <timeWindow> dans le langage de règle.

Sauf indication contraire, le modèle de temporisateur se répète jusqu'à désactivation de la règle temporisée. C'est pourquoi, lorsque la règle temporisée démarre, elle se met en attente pendant la période définie avant de déclencher toute action et elle répète ce comportement jusqu'à sa désactivation ou bien jusqu'à l'arrêt du moteur Active Correlation Technology.

La règle temporisée est unique en ce qu'elle ne contient pas de critères de sélection d'événement. La règle temporisée commence à procéder au traitement conformément à l'intervalle d'activation pour la règle tel que défini par l'élément <activationInterval>. Si l'élément <activationInterval> par défaut est utilisé et que le modèle de temporisateur est défini en mode répétition, la règle temporisée démarre lorsqu'elle est chargée par le moteur Active Correlation Technology et s'arrête en même temps que lui. Pour activer une règle temporisée avec un événement, vous devez définir l'événement dans l'élément <activateOnEvent> au sein de l'élément <activationInterval> de la règle.

Conditions d'exécution de la réponse à la règle

Avec le modèle de temporisateur, la réponse à la règle s'exécute lorsque la plage temporelle est terminée, tel que défini par l'élément <onTimeWindowComplete>.

Exemple d'utilisation de ce modèle de règle

Le modèle de temporisateur peut être utile pour l'implémentation de règles de nettoyage. Exemple d'utilisation du modèle de temporisateur : une règle qui agit comme suite :

> Toutes les 30 minutes, la règle exécute une action qui efface des événements d'information anodins ouverts depuis plus de 48 heures.

Référence associée

«Récapitulatif relatif à la règle temporisée», à la page 56
Ce récapitulatif répertorie tous les éléments de langage de la règle temporisée.

Points communs et particularités de différents modèles de règle

Cette matrice offre une présentation générale des points communs et des particularités de différents modèles de règle.

Le tableau tableau 2, à la page 20 répertorie les principaux éléments de langage des règles et comporte une croix dans la colonne du type de règle pour lequel l'élément est valide. Les principaux éléments de langage sont les éléments enfant

directs des différents types de règle. Cette liste n'inclut pas les éléments contenus dans ces éléments enfant directs, qui peuvent également varier en fonction du type de règle. De plus, la validité de certains attributs d'élément peut elle aussi varier en fonction du type de règle.

Tableau 2. Matrice indiquant les points communs et les particularités de différents modèles de règle

Elément	collecte	calcul	duplication	filtrage	séquence	seuil	temporisateur
<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

Expressions

Une expression est un code qui contient une logique personnalisée pouvant être ajoutée à une règle. Elle peut également accéder à un code externe au moteur Active Correlation Technology. Dans le langage de règle, les expressions sont valides uniquement dans des contextes ou éléments de langage de règle spécifiques.

Les rédacteurs de règle peuvent coder les expressions dans différents buts, suivant le contexte et le résultat à obtenir. Les expressions sont souvent utilisées pour initialiser des variables, définir des critères de sélection d'événement et spécifier des actions de réponse à la règle ou de cycle de vie.

Eléments de langage contenant des expressions

Chaque élément de langage contenant une expression possède un attribut expressionLanguage, qui identifie le langage de programmation dans lequel est rédigée l'expression. Le langage de programmation Java est le seul langage d'expression pris en charge.

Les expressions peuvent être contenues dans les éléments de langage de règle suivants.

- <varInitializer> pour une variable de jeu de règles, de bloc de règles ou de règle
- <filteringPredicate> dans <eventSelector>
- <computedValue> dans <groupingKey>
- <computeFunction> dans une règle de calcul
- <booleanThreshold> dans une règle de seuil
- <computedThreshold> dans une règle de seuil
- Actions de réponse à une règle :

  - <action> dans <onDetection>. Cette action est valide uniquement pour les règles de duplication, de filtrage, de séquence et de seuil.
  - <action> dans <onNextEvent>. Cette action est valide uniquement pour la règle de duplication.
  - <action> dans <onTimeOut>. Cette action est valide uniquement pour les règles de séquence et de seuil.
  - <action> dans <onTimeWindowComplete>. Cette action est valide uniquement pour les règles de collecte et de calcul, ainsi que pour la règle temporisée.
- Actions de cycle de vie d'une règle :
  - <action> dans <onLoad>
  - <action> dans <onActivation>
  - <action> dans <onDeactivation>
  - <action> dans <onUnload>

Aide apportée par la technologie ACT pour le codage des expressions

Pour aider le rédacteur de la règle à coder les expressions, la technologie ACT offre une fonction permettant d'effectuer les opérations suivantes :

- Importer des objets et modules externes (tels que des classes Java) en vue de les utiliser dans des expressions.
- Initialiser des variables de jeu de règles, bloc de règles ou règle, et y accéder par la suite.
- Au moyen de la variable act_event, accéder à l'événement actuellement traité par une règle.
- Au moyen de la variable act_eventCount, accéder au nombre d'événements acceptés par une règle.
- Au moyen de la variable act_eventList, accéder à la liste des événements acceptés par une règle. Cette fonction comprend l'accès à différents attributs d'un événement, ainsi que l'accès à chaque événement par son nom d'alias dans une règle de séquence.
- Au moyen de la variable act_lib, accéder à des méthodes qui permettent d'obtenir et de définir des variables, et de contrôler un flux d'événements à l'aide d'un jeu de règles.
- Au moyen de la variable act_location, accéder à l'emplacement d'une expression dans la hiérarchie de la règle.
- Au moyen de la variable act_nodeName, accéder au nom qualifié complet d'un noeud.
- Au moyen de la variable act_threshold, accéder à la valeur de seuil définie pour une règle de seuil.

Importation et accès à des modules et des objets externes

Cet exemple indique comment vous pouvez rendre un code externe (classes Java, par exemple) et des objets externes accessibles aux expressions. Un objet externe est un objet créé par une application pour communiquer avec des expressions.

Pour pouvoir accéder à un code externe depuis une expression, vous devez rendre ce code accessible aux expressions.

Une importation est un moyen spécifique aux langages de programmation de rendre un code externe accessible à des expressions. L'élément <import> contient un type d'expression particulier qui spécifie les modules externes (classes Java, par exemple) à importer en vue de les utiliser dans d'autres expressions au sein de règles. Une importation peut être définie au niveau du jeu de règles ou d'un bloc de règles.

L'élément <import> suivant contient une expression, rédigée en langage de programmation Java, qui importe la classe StaticHelper et la classe Queue, référençables à partir d'autres expressions :

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

Bien que l'utilisation d'un nom de classe complet ne soit pas obligatoire dans l'instruction d'importation, vous devriez en spécifier un pour éviter d'allonger la durée de la compilation. Par exemple, la classe Java devrait être définie en tant que `com.ibm.act.sample.StaticHelper`, plutôt que `com.ibm.act.sample.*` ou `com.ibm.act.*`.

Accès à une méthode statique

L'exemple suivant indique comment une expression contenue dans une action de réponse à la règle fait référence à la classe StaticHelper après son importation :

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

Accès à une méthode d'instanciation pour un objet

L'exemple suivant indique comment une expression contenue dans une action de réponse à la règle fait référence à la classe Queue après son importation. Dans cet exemple, un objet externe dont le nom est `OutputQueueOne` et le type est `Queue`, est obtenu et utilisé pour placer un événement dans une file d'attente spécifique.

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

Initialisation et accès à des variables

Cet exemple indique comment vous pouvez initialiser des variables de jeu de règles, bloc de règles ou règle, et y accéder par la suite.

Une variable peut être définie au niveau du jeu de règles, d'un bloc de règles ou d'une règle. Avant d'être accessible, elle doit être initialisée dans une expression d'initialisation. L'expression suivante initialise deux variables, l'une nommée hostsList et l'autre 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>
```

L'accès à toutes les variables s'effectue au moyen des expressions. L'exemple suivant montre comment accéder aux variables hostsList et hostsString, initialisées dans l'exemple précédent, au moyen d'une expression contenue dans une action de réponse à la règle. Dans cet exemple, hostsList est modifiée et hostsString reçoit une nouvelle valeur.

```
<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>
```

Accès aux informations relatives aux événements

Les exemples qui suivent montrent comment procéder pour accéder aux informations relatives aux événements par le biais des variables livrées avec Active Correlation Technology.

Exemple d'accès à l'événement en cours :
Le code suivant montre comment utiliser la variable act_event pour obtenir l'attribut 'hostname' associé à un événement :

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

Exemple d'accès aux événements au moyen de la liste d'événements par index :
Le code suivant montre comment utiliser la variable act_eventList pour obtenir le premier événement dans la liste d'événements:

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

Exemple d'accès aux événements au moyen de la liste d'événements par alias :
Contrairement aux autres types de règle, la règle de séquence autorise plusieurs sélecteurs d'événement ; elle requiert d'ailleurs un minimum de deux sélecteurs. L'attribut alias dans l'élément <eventSelector> est valide uniquement dans une règle de séquence ; il nomme de façon unique un événement sélectionné par un sélecteur d'événement donné dans cette règle. Dans une expression au sein d'un prédicat de filtrage ou d'une action, vous pouvez utiliser la variable act_eventList pour accéder à un événement dans une règle de séquence par son nom d'alias.

Le code suivant montre deux sélecteurs d'événement pour une règle de séquence. Les noms d'alias sont TECevent et 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>
```

Le code suivant montre comment utiliser la variable act_eventList pour obtenir l'événement accepté par le premier sélecteur, TECevent :

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

Valeurs recommandées pour le codage des expressions

Ces informations donnent un certain nombre de valeurs recommandées, de conseils et d'astuces pour un codage efficace et efficient des expressions.

- Pour en faciliter la compréhension, la plupart des exemples d'expression fournis ici incluent le code Java directement dans les constructions XML. Néanmoins, lorsque vous créez des règles, la pratique recommandée consiste à utiliser des modules externes pour contenir le code Java et à appeler ces modules externes dans le cadre des expressions.

  Vous pouvez aussi utiliser ou modifier des fragments existants, ou créer de nouveaux fragments dans le générateur de règles pour fournir le code destiné aux appels de modules externes. Les fragments sont des extraits de code source utilisables dans des expressions. Les fragments apparaissent dans la vue Fragment du générateur de règles.

  Avec cette approche, les tâches de conception, développement, édition, test et débogage du code Java peuvent être exécutées dans l'environnement de développement intégré (IDE) de votre choix, et contrôlées comme faisant partie de votre processus de développement standard.
- Pour éviter qu'un code d'expression soit analysé en tant que langage XML, imbriquez-le dans une section CDATA, dans laquelle `<![CDATA[` précède immédiatement le code et `]]>` le suit immédiatement, comme dans l'exemple suivant :

  ```
  <onTimeOut>
  <action expressionLanguage="java">
  <![CDATA[
    IEvent firstEvent = act_eventList.get(0);
    System.out.println("Expired Item: " + firstEvent.getAttribute("sourceComponentId.location"));
  ]]>
  </onTimeOut>
  La gestion de règle et les autres méthodes du moteur Active Correlation Technology
  ne peuvent être appelées à partir d'une action de réponse à la règle ou de cycle de vie.
  ```

  Les analyseurs syntaxiques XML ignorent tout ce qui est à l'intérieur d'une section CDATA.
- Si vous utilisez la méthode act_lib.getExternalContext() dans une expression, ne stockez pas l'objet retourné par la méthode dans une variable de jeu ou de bloc de règle. En effet, une application peut modifier la référence à l'instance d'objet, la variable de jeu ou de bloc de règles associée n'étant ainsi pas mise à jour.
- Si vous utilisez une instruction de retour (`return;`) dans l'expression d'un élément <action>, et que des éléments <action> supplémentaires sont codés pour l'action de réponse à la règle ou de cycle de vie correspondante, l'exécution du code se termine là où l'instruction de retour est codée, puis reprend à partir de l'expression de l'élément <action> suivant.
- La gestion de règle et les autres méthodes du moteur Active Correlation Technology ne peuvent être appelées à partir d'une action de réponse à la règle ou de cycle de vie.

Rubriques connexes

«Importation et accès à des modules et des objets externes», à la page 21
Cet exemple indique comment vous pouvez rendre un code externe (classes Java, par exemple) et des objets externes accessibles aux expressions. Un objet externe est un objet créé par une application pour communiquer avec des expressions.

«Inclusion de fragments dans des expressions contenues dans des règles», à la page 43
Vous pouvez inclure des fragments provenant de la vue Fragment d'Eclipse Workbench dans les expressions contenues dans des règles.

Variables

Dans le langage de règle, certaines variables sont utilisées pour stocker des informations relatives aux événements sur différentes occurrences d'événement ou règles. Ces informations sont ensuite accessibles au moyen d'expressions dans les règles. Certains types de variables sont définis par le programme d'écriture de règle, tandis que d'autres sont fournis par la technologie ACT. Certains types sont accessibles directement dans une expression, tandis que d'autres le sont uniquement par le biais de méthodes fournies par la technologie ACT.

Variables définies dans l'élément <variable> et accessibles par le biais de méthodes

Vous pouvez définir une variable dans l'élément <variable> pour une règle, un bloc de règles ou un jeu de règles. Vous pouvez accéder à cette variable dans une expression en appliquant l'une des méthodes suivantes :

- la méthode getVariable() ou l'une des méthodes get*javatype*Variable()
- la méthode setVariable() ou l'une des méthodes set*javatype*Variable()

Par exemple, si vous définissez la variable *rule_writer_variable* dans l'élément <variable> pour une règle, vous pouvez accéder à cette variable au moyen du code suivant :

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

Variables fournies par la technologie ACT et accessibles directement dans une expression

Les variables suivantes sont fournies par la technologie ACT. Vous pouvez utiliser ces variables en ligne dans une expression.

- act_event
- act_eventList
- act_lib

Par exemple, avec le code suivant, vous pouvez accéder à la variable act_event pour obtenir l'attribut 'hostname' d'un événement :

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

Variables fournies par la technologie ACT et accessibles par le biais de méthodes

Les variables suivantes sont fournies par la technologie ACT. Vous pouvez accéder à ces variables dans une expression en appliquant la méthode getVariable() ou l'une des méthodes get*javatype*Variable().

- act_eventCount
- act_location
- act_nodeName
- act_threshold

Par exemple, avec le code suivant, vous pouvez accéder à la variable act_eventCount :

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

Le tableau tableau 3 indique les constantes fournies par l'interface IACTLibrary pour ces variables. Dans votre code, pour vous assurer que les fautes d'orthographe et les coquilles sont identifiées au moment de la compilation et non de l'exécution, utilisez toujours des constantes qui représentent ces variables plutôt que les variables elles-mêmes. Par exemple, utilisez `act_lib.getIntVariable(IACTLibrary.EVENTCOUNT);` et non `act_lib.getIntVariable("act_eventCount");`.

Tableau 3. Variables avec des constantes associées

Variable	Constante associée
act_eventCount	EVENTCOUNT
act_location	LOCATION
act_nodeName	NODENAME
act_threshold	THRESHOLD

Référence associée

«Elément variable», à la page 118
L'élément <variable> définit une variable et contient des informations sous une forme qui peut être référencée par expressions. Une variable peut être définie au niveau du jeu de règles, d'un bloc de règles ou d'une règle.

Types de données pour les variables Active Correlation Technology

Les variables qui sont fournies par la technologie ACT sont associées à différents types de données.

Le tableau tableau 4 indique les types de données associés à ces variables.

Tableau 4. Types de données pour les variables Active Correlation Technology

Variable	Type de données
act_ event	Type défini par l'interface com.ibm.correlation.IEvent
act_ eventCount	int
act_ eventList	Type défini par l'interface com.ibm.correlation.IEventList
act_ lib	Type défini par l'interface com.ibm.correlation.IACTLibrary
act_ location	java.lang.String
act_ nodeName	java.lang.String
act_ threshold	Comme cette variable correspond à la valeur de l'attribut de seuil de l'élément <computedThreshold> ou <eventCountThreshold> d'une règle de seuil, le type de données doit être le même que celui qui est associé à la valeur d'attribut de seuil.

Contextes d'expression dans lesquels les variables sont valides

Les variables qui sont fournies par la technologie ACT ne sont valides que dans des contextes d'expression spécifiques.

Le tableau tableau 5 répertorie les contextes d'expression dans lesquels ces variables sont valides. Dans le tableau ci-après, chaque variable valide dans un contexte d'expression donné est indiquée par un X inscrit dans la colonne la représentant. D'autres règles relatives à l'usage de chacune de ces variables sont décrites dans les rubriques qui leur sont respectivement consacrées.

Tableau 5. Contextes d'expression dans lesquels les variables sont valides

Contexte d'expression	act_ event	act_ eventCount	act_ eventList	act_ lib	act_ location	act_ nodeName	act_ threshold
<action> dans <onActivation>	X			X	X	X	X
<action> dans <onDeactivation>	X	X	X	X	X	X	X
<action> dans <onDetection>	X	X	X	X	X	X	X
<action> dans <onLoad>				X	X	X	X
<action> dans <onNextEvent>	X	X	X	X	X	X	
<action> dans <onTimeOut>		X	X	X	X	X	X
<action> dans <onTimeWindowComplete>		X	X	X	X	X	
<action> dans <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> pour une règle				X	X	X	X

Variable act_event

La variable act_event donne accès à des méthodes qui s'appliquent à l'événement en cours.

Détails

Les règles temporisées ne traitant pas d'événements, la variable act_event contenue dans ces règles s'applique uniquement aux événements qui les activent ou les désactivent.

La variable act_event ne s'applique au sein des actions <onActivation> et <onDeactivation> que si un événement a activé ou désactivé la règle. Dans la cas contraire, la variable est nulle.

Exemple de programmation

Le code suivant permet d'accéder à la variable act_event pour obtenir l'attribut 'hostname' d'un événement :

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

Méthodes accessibles

Les méthodes auxquelles la variable act_event donne accès sont définies dans l'interface IEvent, comme indiqué dans le tableau tableau 6, à la page 28.

Tableau 6. Interface IEvent, méthodes correspondantes et emplacement des descriptions de méthode Javadoc

Interface	Méthodes	Emplacement des descriptions de méthode Javadoc
IEvent	• get • getAttribute • getBooleanAttribute • getByteAttribute • getShortAttribute • getIntAttribute • getLongAttribute • getFloatAttribute • getDoubleAttribute • getStringAttribute • set • getTimeStamp • setTimeStamp • getType • getOriginal	com.ibm.correlation.IEvent

Variable act_eventCount

La variable act_eventCount est un entier égal au nombre d'événements acceptés par la règle.

Détails

Pour une règle de duplication, la valeur de la variable act_eventCount correspond au nombre total d'événements acceptés, ce qui comprend à la fois l'événement original et toute duplication de cet événement. Pour tous les autres types de règle, cette valeur correspond à la taille de la liste d'événements, accessible par la variable act_eventList à l'aide de la méthode act_eventList.size().

Les variables act_eventCount et act_eventList ne sont pas valides dans une règle temporisée, car ce type de règle ne traite pas les événements.

Si une règle comporte une clé de groupement, les variables act_eventCount, act_eventList et act_threshold ne sont pas valides dans les contextes d'expression suivants :

- Actions de cycle de vie
- <filteringPredicate> dans <activateOnEvent> ou <deactivateOnEvent> dans <activationInterval>
- <computedValue>

En effet, dans ce cas, les variables de règle s'appliquent uniquement à une instance de règle et il n'existe pas d'instance de règle au moment où ces expressions sont exécutées.

Exemple de programmation

Le code suivant permet d'accéder à la variable act_lib pour obtenir le nombre d'événements acceptés par une règle:

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

Variable act_eventList

La variable act_eventList donne accès à des méthodes qui s'appliquent à la liste des événements acceptés par une règle.

Détails

Les règles de filtrage et de duplication possèdent toujours une liste comptant au maximum un événement, car les premières sont sans état et les secondes ne conservent que le premier événement analysé.

Les variables act_eventCount et act_eventList ne sont pas valides dans une règle temporisée, car ce type de règle ne traite pas les événements.

Si une règle comporte une clé de groupement, les variables act_eventCount, act_eventList et act_threshold ne sont pas valides dans les contextes d'expression suivants :

- Actions de cycle de vie
- <filteringPredicate> dans <activateOnEvent> ou <deactivateOnEvent> dans <activationInterval>
- <computedValue>

En effet, dans ce cas, les variables de règle s'appliquent uniquement à une instance de règle et il n'existe pas d'instance de règle au moment où ces expressions sont exécutées.

Exemple de programmation

Le code suivant permet d'accéder à la variable act_eventList pour obtenir le deuxième événement de la liste d'événements :

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

Méthodes accessibles

Les méthodes auxquelles la variable act_eventList donne accès sont définies dans l'interface IEventList, comme indiqué dans le tableau tableau 7.

Tableau 7. Interface IEventList, méthodes correspondantes et emplacement des descriptions de méthode Javadoc

Interface	Méthodes	Emplacement des descriptions de méthode Javadoc
IEventList	• get • size • isEmpty • listIterator	com.ibm.correlation.IEventList

Variable act_lib

La variable act_lib donne accès à des méthodes de bibliothèque dans Active Correlation Technology.

Détails

Les méthodes auxquelles la variable act_lib peut accéder dépendent de l'élément de langage de règle qui contient l'expression dans laquelle elle est utilisée. Voir le tableau tableau 8.

Tableau 8. Méthodes auxquelles la variable act_lib peut accéder, selon le contexte de l'expression qui la contient

Contexte d'expression	Méthodes IACTLibrary	Méthodes IExitableActionLibrary	Méthodes IActionLibrary
<action> dans <onActivation>	X		
<action> dans <onDeactivation>	X		
<action> dans <onDetection>	X	X	X
<action> dans <onLoad>	X		
<action> dans <onNextEvent>	X	X	X
<action> dans <onTimeOut>	X		X
<action> dans <onTimeWindowComplete>	X		X
<action> dans <onUnload>	X		
<booleanThreshold>	X		
<computedThreshold>	X		
<computedValue>	X		
<computeFunction>	X		
<filteringPredicate>	X		
<varInitializer>	X		

Exemple de programmation

Le code suivant permet d'accéder à la variable act_lib pour appeler la méthode permettant de quitter le jeu de règles :

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

Méthodes accessibles

Les méthodes auxquelles la variable act_lib donne accès sont définies dans les interfaces IACTLibrary, IExitableActionLibrary et IActionLibrary, comme indiqué dans le tableau tableau 9, à la page 31.

Tableau 9. Interfaces, méthodes correspondantes et emplacement des descriptions de méthode Javadoc

Interface	Méthodes	Emplacement des descriptions de méthode Javadoc
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 Les méthodes définies dans l'interface IACTLibrary sont également disponibles dans l'interface IActionLibrary.	com.ibm.correlation.IActionLibrary
IExitableActionLibrary	• exitRuleSet • exitRuleBlock Les méthodes définies dans les interfaces IACTLibrary et IActionLibrary sont également disponibles dans l'interface IExitableActionLibrary.	com.ibm.correlation.IExitableActionLibrary

Variable act_location

La variable act_location est une chaîne qui identifie l'emplacement d'une expression dans la hiérarchie de la règle.

Détails

L'emplacement est un nom qualifié complet qui indique la position de l'expression dans la hiérarchie de la règle. Il se présente sous la forme *identifier.identifier...*, où chaque occurrence de *identifier* désigne l'un des éléments suivants :

- La valeur de l'attribut de nom d'un élément XML situé dans la hiérarchie respective.
- Pour les éléments qui figurent plusieurs fois dans un bloc de règles ou une règle et ne possèdent pas d'attribut de nom : l'élément XML qui contient l'expression, suivi d'un numéro d'index entre parenthèses. Ce numéro d'index indique la position de l'expression dans l'élément qui la contient. Le décompte des numéros d'index commence à partir de 0, et non de 1. Par conséquent, si un élément est contenu dans le troisième élément <action>, par exemple, le numéro d'index sera `action[2]`.

Ces identificateurs sont classés par ordre décroissant, depuis le bloc de règles le plus vaste jusqu'à l'élément le plus petit contenant l'expression.

Exemple de programmation

Le code suivant permet d'accéder à la variable act_lib pour obtenir l'emplacement de l'expression :

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

Exemples d'emplacement renvoyés par la variable

Les valeurs suivantes sont des exemples d'emplacement renvoyés par la variable act_location.

`ruleBlockA.ruleA.eventSelector[3].filteringPredicate`
L'expression est contenue dans l'élément suivant :
- Le bloc de règles dont l'attribut de nom a pour valeur `ruleBlockA`
- La règle dont l'attribut de nom a pour valeur `ruleA`
- Le quatrième élément <eventSelector>
- L'élément <filteringPredicate>

`ruleBlockA.ruleA.onDetection.action[5]`
L'expression est contenue dans l'élément suivant :
- Le bloc de règles dont l'attribut de nom a pour valeur `ruleBlockA`
- La règle dont l'attribut de nom a pour valeur `ruleA`
- L'élément <onDetection>
- Le sixième élément <action>

`ruleBlockA.ruleA.variableA.varInitializer`
L'expression est contenue dans l'élément suivant :
- Le bloc de règles dont l'attribut de nom a pour valeur `ruleBlockA`
- La règle dont l'attribut de nom a pour valeur `ruleA`
- La variable dont l'attribut de nom a pour valeur `variableA`
- L'élément <varInitializer>

Variable act_nodeName

La variable act_nodeName est une chaîne qui identifie le nom qualifié complet d'un noeud.

Détails

Dans Active Correlation Technology, un noeud est un objet situé dans la hiérarchie de la règle qui peut être ajouté, supprimé ou remplacé de manière indépendante et individuelle au sein d'un jeu de règles. En particulier, les objets suivants sont des

noeuds : règles, blocs de règles, variables de bloc de règles et variables de jeu de règles. Un objet ne pouvant être manipulé de manière indépendante et individuelle au-dessous du niveau de la règle, la variable de règle n'est pas un noeud.

Pour une règle dont l'attribut de nom a pour valeur `rule1`, et située dans un bloc de règles dont l'attribut de nom a pour valeur `ruleBlockA`, le nom de noeud qualifié complet est `ruleBlockA.rule1`. Les règles étant ordonnées de manière hiérarchique au sein d'un jeu de règles, un point (.) est utilisé dans un nom de noeud qualifié complet pour indiquer une descente vers un niveau de noeud moins élevé.

Exemple de programmation

Le code suivant permet d'accéder à la variable act_nodeName pour obtenir le nom qualifié complet d'un noeud :

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

Variable act_threshold

La variable act_threshold est la valeur de l'attribut de seuil, à savoir la valeur de seuil définie, de l'élément <computedThreshold> ou <eventCountThreshold> d'une règle de seuil.

Détails

La variable act_threshold est uniquement valide dans une règle de seuil.

Si une règle comporte une clé de groupement, les variables act_eventCount, act_eventList et act_threshold ne sont pas valides dans les contextes d'expression suivants :

- Actions de cycle de vie
- <filteringPredicate> dans <activateOnEvent> ou <deactivateOnEvent> dans <activationInterval>
- <computedValue>

En effet, dans ce cas, les variables de règle s'appliquent uniquement à une instance de règle et il n'existe pas d'instance de règle au moment où ces expressions sont exécutées.

Exemple de programmation

Le code suivant permet d'accéder à la variable act_lib pour obtenir une valeur de seuil définie :

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

Flux d'événements dans un jeu de règles

Dans un jeu de règles, le flux d'événements suit l'ordre dans lequel les blocs de règles et les règles sont codés. Lorsque le moteur Active Correlation Technology reçoit un événement, il détermine le type d'événement et identifie les règles qui l'utilisent pour l'activation de la règle, le traitement de l'événement ou la désactivation de la règle.

Utilisation des événements par les règles

Chaque règle qui utilise un événement commence par déterminer si ce dernier répond à tous les critères spécifiés pour l'activation de la règle, le traitement de l'événement ou la désactivation de la règle. Si c'est le cas, elle applique les actions suivantes :

Pour l'activation de la règle
Les actions contenues dans l'élément <onActivation> de la règle sont exécutées si elles sont codées.

Pour le traitement de l'événement
La règle traite l'événement. Lorsque le modèle de règle correspond, les actions de réponse à la règle sont exécutées si elles sont codées. Dans certaines situations :

- L'action peut interrompre le traitement de l'événement pour la suite du bloc de règles ou du jeu de règles.
- L'action peut envoyer un événement nouveau ou existant vers une autre règle ou un autre bloc de règles pour qu'ils procèdent à son traitement.

Pour la désactivation de la règle
Les actions contenues dans l'élément <onDeactivation> de la règle sont exécutées si elles sont codées.

Méthodes susceptibles d'influencer le flux d'événements

Active Correlation Technology offre des méthodes qui peuvent être appelées pour influencer le flux d'événements dans le jeu de règles. Ces méthodes sont disponibles par le biais de la variable act_lib.

exitRuleSet
Cette méthode spécifie que l'événement en cours n'est traité par aucune règle supplémentaire dans le jeu de règles.

exitRuleBlock
Cette méthode spécifie que l'événement en cours n'est traité par aucune règle supplémentaire dans le bloc de règles en cours ou dans tout bloc de règles contenu dans ce dernier. Cependant, il est traité par des règles supplémentaires en-dehors de la portée du bloc en cours.

forward
Cette méthode spécifie qu'un événement doit être envoyé vers d'autres règles et blocs de règles, bien que la règle en cours n'ait pas fini de le traiter. Ensuite, chaque règle et chaque bloc de règles traite entièrement l'événement avant de le renvoyer vers la règle qui a appelé la méthode forward.

forwardOnCompletion
Cette méthode spécifie qu'un événement doit être envoyé vers d'autres règles et blocs de règles au terme de son traitement par la règle en cours.

Chapitre 3. Présentation de la rédaction de règles

Avant de rédiger des règles pour corréler des événements, vous devez comprendre et planifier la corrélation d'événements et concevoir les règles. Vous pouvez utiliser le générateur de règles Active Correlation Technology pour rédiger les règles et compiler le jeu de règles.

Le générateur de règles Active Correlation Technology est une interface graphique destinée à la rédaction de règles. Le générateur comprend une aide en ligne. Dans le générateur de règles, le fichier de jeu de règles obtenu est un fichier XML du type actl.

La technologie ACT n'offre pas de système de gestion des modifications ou de contrôle des versions.

Planification d'une corrélation d'événements

Pour planifier une corrélation d'événements, il est nécessaire de comprendre ou de s'informer sur cette notion et sur les différents moyens de mise en oeuvre dans votre application.

Assurez-vous de bien comprendre les concepts suivants :

- Les informations contenues dans les rubriques Chapitre 1, «Introduction», à la page 3 et Chapitre 2, «Présentation du langage de règle», à la page 5
- Les événements traités par votre application

Chaque application peut traiter un ensemble d'événements différent, comme le montrent les exemples suivants :

Entreprise d'assurance
Dans une entreprise d'assurance, il est possible de générer et corréler des événements qui suivent le flux de travaux au cours du processus de déclaration de sinistre, afin de déterminer si les processus de l'entreprise s'achèvent dans les délais convenus.

Vente Dans un type d'entreprise différent, il est possible de récapituler, rapporter et comparer périodiquement les résultats des ventes à un objectif fixé, afin de montrer dans quelle mesure les objectifs de vente sont atteints pour une période donnée.

Environnement informatique
Dans un environnement informatique, un système vital peut générer un événement toutes les minutes pour indiquer que l'exécution d'un serveur de base de données se déroule normalement. Vous pouvez écrire des règles de corrélation pour surveiller la réception de ces signaux de présence et appliquer certaines actions de réponse à la règle si un signal attendu n'est pas reçu.

Vous devez également comprendre le format des événements que traite votre application. Active Correlation Technology offre des classes et des méthodes Java qui permettent d'accéder aux données des événements traités par le moteur Active Correlation Technology. Cependant, il est important de comprendre les objets d'événement sous-jacents dans leurs grandes lignes si vous souhaitez utiliser ces classes et méthodes pour accéder aux événements traités ou les modifier.

Pour planifier une corrélation d'événements, procédez comme suit :

1. Déterminez les événements de votre application que vous souhaitez corréler.
2. Déterminez les modèles de règle à utiliser pour corréler ces événements.

   Un modèle de règle représente une situation de corrélation d'événement spécifique et peut permettre de corréler des événements qui contribuent à cette situation d'une manière ou d'une autre. Réfléchissez à la relation entre les événements que traite votre application et les modèles de règle définis par le langage de règle Active Correlation Technology. Vous déterminerez ainsi plus facilement les modèles de règle nécessaires.

   Utilisez toujours le modèle le mieux adapté à votre situation de corrélation d'événements. Par exemple, si vous avez besoin d'une règle pour détecter une séquence d'événements donnée, ne rédigez pas un code dans le but d'inclure le comportement du modèle de séquence dans les actions de réponse à une règle de filtrage. Utilisez plutôt le modèle de séquence pour créer une règle de séquence.
3. Identifiez les constructions de chaque modèle de règle à utiliser.

   Les informations suivantes récapitulent les principales constructions du langage de règle, mais les détails de chacune d'elles sont propres au modèle de règle. La présentation de ces informations suit à peu près celle qui est utilisée dans l'interface graphique du générateur de règles :

   Caractéristiques
   Définition des caractéristiques de la règle, y compris son nom, sa description et son modèle. Pour plus de détails, reportez-vous aux rubriques suivantes :
   - «Elément collectionRule», à la page 75
   - «Elément computationRule», à la page 77
   - «Elément duplicateRule», à la page 85
   - «Elément filterRule», à la page 93
   - «Elément sequenceRule», à la page 108
   - «Elément thresholdRule», à la page 113
   - «Elément timerRule», à la page 116

   Variables
   Définition des variables de règle, y compris le nom, le type, la description et l'expression d'initialisation de chaque variable. Pour plus de détails, voir «Elément variable», à la page 118.

   Sélection d'événement
   Définition du critère qui détermine les événements autorisés à être traités par la règle. Pour plus de détails, voir «Elément eventSelector», à la page 90.

   Clé de groupement
   Définition de la clé de groupement, qui permet de conduire une règle à créer une instance de règle (ou une copie) pour chaque groupe d'événements possédant des caractéristiques communes. Pour plus de détails, voir «Elément groupingKey», à la page 94.

   Spécifications des modèles
   Spécification de la période au cours de laquelle la règle avec état effectue des traitements pour correspondre à son modèle, et définition des particularités de certains modèles de règle avec état. Pour plus de détails, voir «Elément timeWindow», à la page 117.

Pour la règle de calcul, cette spécification comprend la définition du calcul à appliquer aux événements collectés. Pour plus de détails, voir «Elément computeFunction», à la page 82.

Pour la règle de seuil, elle comprend la définition du type de seuil et d'autres informations spécifiques à ce type. Pour plus de détails, reportez-vous aux rubriques suivantes :

- «Elément booleanThreshold», à la page 74
- «Elément computedThreshold», à la page 79
- «Elément eventCountThreshold», à la page 87

Réponses à la règle
Définition des actions à appliquer lorsque la règle a achevé ses opérations de traitement.

Pour plus de détails, reportez-vous aux rubriques suivantes :

- Pour les règles de duplication, de filtrage, de séquence ou de seuil : «Elément onDetection», à la page 101
- Pour les règles de duplication : «Elément onNextEvent», à la page 102
- Pour les règles de séquence ou de seuil : «Elément onTimeOut», à la page 103
- Pour les règles de collecte, de calcul, de duplication ou pour la règle temporisée : «Elément onTimeWindowComplete», à la page 104

Intervalle d'activation
Définition du moment où une règle est active ou inactive. Pour plus de détails, voir «Elément activationInterval», à la page 69.

Cycle de vie
Définition des actions à appliquer, le cas échéant, lors de ces quatre étapes principales dans le cycle de vie d'une règle : chargement, activation, désactivation et déchargement. Il n'est généralement pas nécessaire de définir ces actions. Pour plus de détails, voir «Elément lifeCycleActions», à la page 98.

4. Identifiez les méthodes Java et les fragments associés qui doivent être appelés dans des expressions de règle. Au lieu de rédiger un code Java complet dans les expressions de règle, les rédacteurs de règle devraient utiliser des méthodes Java permettant d'appeler des modules externes. Ces modules externes peuvent être fournis par l'application qui intègre la technologie ACT ou être créés par le rédacteur de la règle, selon le cas. Les fragments qui sont associés à chacune des méthodes Java devraient également être identifiés. Pour plus d'informations, voir «Valeurs recommandées pour le codage des expressions», à la page 24.

Passez à la rubrique «Conception des règles pour la corrélation d'événements».

Conception des règles pour la corrélation d'événements

Après avoir planifié la corrélation d'événements, vous devez concevoir les règles pour corréler les événements.

Pour commencer, effectuez la «Planification d'une corrélation d'événements», à la page 35.

Pour concevoir les règles, procédez comme suit :

1. Concevez l'organisation des règles et des blocs de règles au sein du jeu de règles.
2. Concevez le niveau auquel les différentes variables doivent être définies, par exemple, au niveau du jeu de règles, du bloc de règles ou de la règle.
3. Concevez les éléments de chaque règle en fonction du modèle de règle.

   Cette étape utilise les constructions de chaque modèle de règle que vous avez identifiées au cours de l'étape de planification.
4. Concevez les interactions entre les règles, en particulier pour ce qui est du réacheminement des événements et de l'omission du traitement des événements en double par le jeu de règles.

   Pour plus de détails, voir «Flux d'événements dans un jeu de règles», à la page 33.
5. Concevez, développez et testez toutes les méthodes Java et les fragments associés que vous avez créés pour être appelés au sein d'expressions.

Passez à la rubrique «Mise en route pour l'utilisation du générateur de règles».

Mise en route pour l'utilisation du générateur de règles

Après avoir conçu les règles servant à corréler des événements, vous pouvez utiliser le générateur de règles Active Correlation Technology pour les écrire.

Ce paragraphe énumère les principales étapes d'écriture de règle à l'aide du générateur de règles.

1. Ouvrez Eclipse Workbench.
2. Définissez votre perspective dans Eclipse Workbench.
3. Définissez vos préférences pour la technologie ACT, ou acceptez les préférences par défaut.
4. Créez un projet dans lequel vous stockerez le fichier de jeu de règles ou utilisez un projet existant.
5. Créez un fichier de jeu de règles, puis stockez-le dans le projet de votre choix.
6. Créez au moins un bloc de règles dans le jeu de règles. Vous pouvez créer des blocs de règles supplémentaires, ainsi que des blocs dans des blocs.
7. Créez des règles dans les blocs de règles.
8. Validez le jeu de règles.
9. Compilez le jeu de règles.
10. Mettez à jour le jeu de règles selon vos besoins.

Vous pouvez également inclure des fragments provenant de la vue Fragment d'Eclipse Workbench dans les expressions contenues dans des règles.

Définition de la perspective dans Eclipse Workbench

Avant toute chose, vous devez définir votre perspective dans Eclipse Workbench. Cette rubrique explique comment ouvrir la perspective Générateur de règles.

Pour commencer, ouvrez Eclipse Workbench, si ce n'est pas déjà fait.

Vous pouvez utiliser une autre perspective que celle du générateur de règles, mais les étapes d'exécution des différentes tâches pourront varier légèrement selon la perspective que vous choisissez.

Pour ouvrir la perspective Générateur de règles, procédez comme suit :

1. Dans le plan de travail, cliquez sur **Fenêtre → Ouvrir la perspective → Autre...**.
2. Cliquez sur **Générateur de règles**, puis sur **OK**. La perspective Générateur de règles apparaît à l'écran.

Passez à la rubrique «Définition des préférences».

Définition des préférences

Avant de créer un fichier de jeu de règles, vous devez vous assurer que vos préférences pour la technologie ACT sont définies correctement dans Eclipse Workbench. Cette rubrique explique comment définir ces préférences.

Pour commencer, ouvrez Eclipse Workbench, si ce n'est pas déjà fait.

Pour définir vos préférences pour la technoogie ACT, procédez comme suit :

1. Dans le plan de travail, cliquez sur **Fenêtre → Préférences...**.
2. Cliquez sur **Active Correlation Technology**, puis, dans la page Active Correlation Technology, indiquez si vous voulez ajouter «act» comme préfixe pour les règles et les blocs de règles nouvellement créés dans le générateur de règles. Par défaut, le préfixe «act» *n'est pas* ajouté.
3. Développez **Active Correlation Technology**. Les éléments suivants peuvent également apparaître en fonction de votre application. Cliquez sur ces éléments pour définir les préférences associées.

Elément	Préférence associée
Fournisseur de définition Common Base Event	Vous pouvez spécifier les fichiers XML qui fournissent la structure pour un plusieurs types d'événements (y compris les noms et les types des attributs qu'un événement donné peut contenir) en conformité avec la spécification Common Base Event (événement de base commun). Ces types d'événement et ces attributs sont ensuite disponibles lorsque vous créez des règles.
Compilateur	Vous pouvez spécifier les éléments primaires suivants : • Si les jeux de règles compilés doivent être sauvegardés • Le type de fichier pour les jeux de règles compilés • Le chemin de classes pour le code à utiliser au moment de la compilation Par défaut, les jeux de règles compilés sont sauvegardés et apparaissent dans la vue Navigateur avec le type de fichier `acts`, ce qui indique que la sortie du compilateur est un jeu de règles sérialisé.

Passez à la rubrique «Création d'un projet en vue du stockage d'un fichier de jeu de règles», à la page 40.

Création d'un projet en vue du stockage d'un fichier de jeu de règles

Dans Eclipse Workbench, lorsque vous créez un fichier de jeu de règles, vous devez définir le projet dans lequel vous allez stocker le fichier. Vous devez donc créer un projet avant de créer un fichier de jeu de règles, ou utiliser un projet existant. Cette rubrique explique comment créer un projet dans Eclipse Workbench.

Pour commencer, ouvrez Eclipse Workbench, si ce n'est pas déjà fait. Ouvrez également la perspective Générateur de règles.

Pour créer un projet simple dans le plan de travail, procédez comme suit :

1. Cliquez sur **Fichier** → **Nouveau** → **Projet...**.
2. Dans l'assistant Nouveau projet, cliquez sur **Projet** → **simple**, puis sur **Suivant**.
3. Dans la zone **Nom du projet**, saisissez un nom unique pour le projet. Vous pouvez également utiliser l'emplacement par défaut pour le projet ou choisir un emplacement différent.
4. Cliquez sur **Fin**. Le nouveau projet apparaît ensuite dans la vue Navigateur de la perspective Générateur de règles.

Passez à la rubrique «Création d'un jeu de règles».

Création d'un jeu de règles

Cette rubrique explique comment créer un jeu de règles.

Pour commencer, ouvrez Eclipse Workbench, si ce n'est pas déjà fait. Ouvrez également la perspective Générateur de règles.

Pour créer un fichier de jeu de règles, procédez comme suit :

1. Cliquez sur **Fichier** → **Nouveau** → **Fichier de jeu de règles**.
2. Dans l'assistant de fichier de jeu de règles, cliquez sur le nom du dossier associé au projet dans lequel vous voulez stocker le fichier de jeu de règles. Il s'agit probablement du projet que vous avez créé dans «Création d'un projet en vue du stockage d'un fichier de jeu de règles». Le nom de dossier est alors affiché dans la première zone.
3. Dans la zone **Nom du fichier**, saisissez un nom pour le fichier de jeu de règles. Le fichier doit être de type actl.
4. Cliquez sur **Suivant**.
5. Saisissez un nom unique pour le jeu de règles et une description facultative. Si vous ne voulez pas utiliser la valeur par défaut pour la zone **Codage XML**, spécifiez également le style de codage du fichier de jeu de règles, qui est un fichier XML.
6. Cliquez sur **Fin**. Le fichier de jeu de règles apparaît alors dans son dossier de projet dans la vue Navigateur. Comme le fichier est validé lors de sa sauvegarde, l'indication «Validé» apparaît à côté du fichier. Le fichier de jeu de règles apparaît également dans la vue Structure.

Passez à la rubrique «Création d'un bloc de règles».

Création d'un bloc de règles

Cette rubrique explique comment créer un bloc de règles au sein d'un jeu de règles ou d'un autre bloc de règles.

Pour commencer, ouvrez Eclipse Workbench, si ce n'est pas déjà fait. Ouvrez également la perspective Générateur de règles.

Si vous créez un jeu de règles pour la première fois, vous devez créer au moins un bloc de règles au sein d'un jeu de règles avant de créer des règles. Après avoir créé ce premier bloc de règles, vous pouvez créer des blocs supplémentaires, y compris dans des blocs.

Pour créer un bloc de règles, procédez comme suit :

1. Dans la vue Navigateur, cliquez deux fois sur le nom du fichier de jeu de règles que vous voulez mettre à jour. Le fichier s'ouvre alors dans la vue Structure. Lorsque vous cliquez sur le jeu de règles dans la vue Structure, les informations qui lui sont actuellement associées apparaissent dans la zone de l'éditeur.
2. Dans la vue Structure, cliquez avec le bouton droit de la souris sur le jeu de règles.
3. Cliquez sur **Nouvel enfant → Bloc de règles**. Un bloc de règles est alors affiché au sein du jeu de règles dans la vue Structure et les informations sur le bloc de règles sont affichées dans la zone de l'éditeur.

Vous pouvez maintenant créer des blocs de règles supplémentaires en procédant comme suit :

- Pour créer un bloc de règles en tant qu'homologue d'un bloc de règles existant, cliquez avec le bouton droit de la souris sur le bloc de règles existant, puis cliquez sur **Nouvel élément apparenté → Bloc de règles**.
- Pour créer un bloc de règles au sein d'un bloc de règles existant, cliquez avec le bouton droit de la souris sur le bloc de règles existant, puis cliquez sur **Nouvel enfant → Bloc de règles**.

Vous pouvez également mettre à jour les informations sur le jeu de règles et sur chaque bloc de règles en utilisant l'éditeur. Passez à la rubrique «Création d'une règle».

Création d'une règle

Cette rubrique explique comment créer une règle.

Pour commencer, ouvrez Eclipse Workbench, si ce n'est pas déjà fait. Ouvrez également la perspective Générateur de règles.

Une règle doit être créée au sein d'un bloc de règles. Pour créer une règle, procédez comme suit :

1. Dans la vue Structure, cliquez avec le bouton droit de la souris sur le bloc de règles à mettre à jour.
2. Cliquez sur **Nouvel enfant** et sur le type de règle que vous voulez créer. La règle est alors affichée au sein du bloc de règles dans la vue Structure et les informations concernant la règle sont affichées dans la zone de l'éditeur.

Vous pouvez ajouter des règles supplémentaires aux blocs de règles en procédant de la même manière. Vous pouvez également mettre à jour les informations concernant chaque règle en utilisant l'éditeur. Passez à la rubrique «Validation d'un jeu de règles», à la page 42.

Validation d'un jeu de règles

Cette rubrique explique comment valider un jeu de règles avant de le compiler.

Pour commencer, ouvrez Eclipse Workbench, si ce n'est pas déjà fait. Ouvrez également la perspective Générateur de règles.

Pour valider un jeu de règles, procédez comme suit :

1. Dans la vue Structure, cliquez avec le bouton droit de la souris sur une règle, un bloc de règles ou le jeu de règles.
2. Cliquez sur **Valider le jeu de règles**. Lorsque la validation est terminée, une fenêtre de message apparaît pour indiquer l'occurrence éventuelle d'incidents. Si la validation est effectuée avec succès, l'indication «Validé» apparaît à droite du nom de fichier associé au jeu de règles dans la vue Navigateur.

A l'issue de la validation, passez à la rubrique «Compilation d'un jeu de règles».

Compilation d'un jeu de règles

Cette rubrique explique comment compiler un jeu de règles.

Pour commencer, ouvrez Eclipse Workbench, si ce n'est pas déjà fait. Ouvrez également la perspective Générateur de règles.

Dans la vue Navigateur ou Structure, cliquez avec le bouton droit de la souris sur le jeu de règles que vous voulez compiler, puis cliquez sur **Compiler le jeu de règles**. Toutes les erreurs de compilation sont affichées dans la vue Erreurs.

Par défaut, s'il n'existe aucune erreur de navigation, le jeu de règles compilé apparaît dans la vue Navigateur avec le type de fichier pas défaut `acts`, ce qui indique qu'il s'agit d'un jeu de règles sérialisé. En outre, dans la vue Navigateur, le mot «Compilé» apparaît à droite du nom de fichier associé au jeu de règles.

Mise à jour d'un jeu de règles

Cette rubrique explique comment mettre à jour un jeu de règles.

Ouvrez Eclipse Workbench, si ce n'est pas déjà fait. Ouvrez également la perspective Générateur de règles.

Dans la vue Structure, cliquez sur l'élément (règle, bloc de règles ou jeu de règles) à mettre à jour. Les informations actuelles relatives à l'élément sont alors affichées dans la zone de l'éditeur, ce qui vous permet de les mettre à jour si nécessaire.

Pour créer une règle ou un bloc de règles en tant qu'homologue d'une règle ou d'un bloc de règles existant, procédez comme suit :

1. Cliquez avec le bouton droit de la souris sur le bloc de règles ou la règle existant.
2. Cliquez sur **Nouvel élément apparenté**, puis sur l'élément (bloc de règles ou type de règle) à ajouter.

Pour créer une règle ou un bloc de règles dans un bloc de règles existant, procédez comme suit :

1. Cliquez avec le bouton droit de la souris sur le bloc de règles existant.

2. Cliquez sur **Nouvel enfant**, puis sur l'élément (bloc de règles ou type de règle) à ajouter.

Pour accéder à d'autres fonctions de mise à jour dans la vue Structure, procédez comme suit :

1. Cliquez avec le bouton droit de la souris sur l'élément (bloc de règles ou règle) à mettre à jour.
2. Cliquez sur l'élément de menu correspondant à la fonction, par exemple **Supprimer**, **Copier** ou **Coller**.

Inclusion de fragments dans des expressions contenues dans des règles

Vous pouvez inclure des fragments provenant de la vue Fragment d'Eclipse Workbench dans les expressions contenues dans des règles.

Pour commencer, ouvrez Eclipse Workbench, si ce n'est pas déjà fait. Ouvrez également la perspective Générateur de règles.

La vue Fragment apparaît dans la perspective Générateur de règle. Les fragments sont organisés en catégories selon leur fonction.

Pour inclure un fragment de la vue Fragment, procédez comme suit :

1. Cliquez sur une catégorie de fragments pour voir les noms des fragments qu'elle contient.
2. Cliquez sur le fragment que vous voulez inclure dans une expression.
3. Faites glisser le fragment dans la zone **Expression** correspondante. Le code est placé sur la position du curseur dans la zone **Expression**. Si le code nécessite des informations en entrée du rédacteur de règle, comme un message textuel ou des valeurs de variable spécifiques, vous êtes invité à fournir ces informations avant que le code soit inclus dans l'expression.

Passez à la rubrique «Validation d'un jeu de règles», à la page 42.

Partie 2. Document de référence pour le rédacteur de règle

Chapitre 4. Récapitulatif relatif à l'organisation du jeu de règles

Ce document répertorie tous les éléments de langage d'un jeu de règles, d'un bloc de règles et de chaque type de règle. Il fait office de référence pouvant être facilement consultée en cas d'interrogation sur le codage d'un jeu de règles.

Le tableau tableau 10 explique la signification de la notation qui suit chaque élément de langage. *n* représente un nombre illimité.

Tableau 10. Explication de la notation qui définit le nombre d'occurrences d'un élément de langage

Notation	Signification
(0, 1)	0 indique que l'élément de langage est facultatif. 1 indique qu'en cas de codage, une seule occurrence est autorisée.
(0, *n*)	0 indique que l'élément de langage est facultatif. *n* indique qu'en cas de codage, plusieurs occurrences sont autorisées.
(1, 1)	Le premier 1 indique que l'élément de langage est obligatoire. Le second 1 indique qu'une seule occurrence est autorisée.
(1, *n*)	1 indique que l'élément de langage est obligatoire. *n* indique que plusieurs occurrences sont autorisées.
(2, *n*)	2 indique que deux occurrences de l'élément de langage sont obligatoires. *n* indique que des occurrences supplémentaires sont autorisées.

Ces éléments doivent être codés dans l'ordre indiqué. Il n'est pas nécessaire de coder les éléments facultatifs, mais tous les éléments codés doivent être placés dans le bon ordre.

Récapitulatif relatif au jeu de règles

Ce récapitulatif répertorie les éléments de langage d'un jeu de règles.

Eléments d'un jeu de règles

<ruleSet> contient les éléments suivants :

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

Référence associée

«Récapitulatif relatif au bloc de règles»
Ce récapitulatif répertorie les éléments de langage d'un bloc de règles.

Récapitulatif relatif au bloc de règles

Ce récapitulatif répertorie les éléments de langage d'un bloc de règles.

Eléments du bloc de règles

<ruleBlock> contient les éléments suivants.

S'ils sont codés, les éléments <comment>, <import> et <variable> doivent l'être dans l'ordre indiqué. Les éléments restants peuvent être codés dans n'importe quel ordre.

- <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)

Référence associée

«Récapitulatif relatif à la règle de collecte»
Ce récapitulatif répertorie tous les éléments de langage de la règle de collecte.

«Récapitulatif relatif à la règle de calcul», à la page 50
Ce récapitulatif répertorie tous les éléments de langage de la règle de calcul.

«Récapitulatif relatif à la règle de duplication», à la page 51
Ce récapitulatif répertorie tous les éléments de langage de la règle de duplication.

«Récapitulatif relatif à la règle de filtrage», à la page 52
Ce récapitulatif répertorie tous les éléments de langage de la règle de filtrage.

«Récapitulatif relatif à la règle de séquence», à la page 53
Ce récapitulatif répertorie tous les éléments de langage de la règle de séquence.

«Récapitulatif relatif à la règle de seuil», à la page 55
Ce récapitulatif répertorie tous les éléments de langage de la règle de seuil.

«Récapitulatif relatif à la règle temporisée», à la page 56
Ce récapitulatif répertorie tous les éléments de langage de la règle temporisée.

Récapitulatif relatif à la règle de collecte

Ce récapitulatif répertorie tous les éléments de langage de la règle de collecte.

Eléments de la règle

<collectionRule> contient les éléments suivants :

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

  - <activationTime> (0, 1)
    - <start> (0, 1)
      - L'un des trois élément suivants (1, 1) :
        - <dateTime>
        - <whenLoaded>
        - <inactiveWhenLoaded>
    - <stop> (0, 1)
      - L'un des trois élément suivants (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)
  - Les trois éléments suivants, dans n'importe quel ordre (1, n):
    - <attributeAlias>
      - <eventAttribute> (2, n)
    - <attributeName>
    - <computedValue>
- <timeWindow> (1, 1)
  - L'un des deux élément suivants (1, 1) :

- <timeInterval>
- <runUntilDeactivated>

• <onTimeWindowComplete> (0, 1)
  – <action> (0, n)

Récapitulatif relatif à la règle de calcul

Ce récapitulatif répertorie tous les éléments de langage de la règle de calcul.

Eléments de la règle

<computationRule> contient les éléments suivants :

• <comment> (0, 1)
• <variable> (0, n)
  – <comment> (0, 1)
  – <varInitializer> (1, 1)
• <activationInterval> (0, 1)
  – <activationTime> (0, 1)
    - <start> (0, 1)
      • L'un des trois élément suivants (1, 1) :
        – <dateTime>
        – <whenLoaded>
        – <inactiveWhenLoaded>
    - <stop> (0, 1)
      • L'un des trois élément suivants (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)
  - Les trois éléments suivants, dans n'importe quel ordre (1, n):
    - <attributeAlias>
      - <eventAttribute> (2, n)
    - <attributeName>
    - <computedValue>
- <computeFunction> (1, 1)
- <timeWindow> (1, 1)
  - L'un des deux élément suivants (1, 1) :
    - <timeInterval>
    - <runUntilDeactivated>
- <onTimeWindowComplete> (0, 1)
  - <action> (0, n)

Récapitulatif relatif à la règle de duplication

Ce récapitulatif répertorie tous les éléments de langage de la règle de duplication.

Eléments de la règle

<duplicateRule> contient les éléments suivants :

- <comment> (0, 1)
- <variable> (0, n)
  - <comment> (0, 1)
  - <varInitializer> (1, 1)
- <activationInterval> (0, 1)
  - <activationTime> (0, 1)
    - <start> (0, 1)
      - L'un des trois élément suivants (1, 1) :
        - <dateTime>
        - <whenLoaded>
        - <inactiveWhenLoaded>
    - <stop> (0, 1)
      - L'un des trois élément suivants (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)
    – Les trois éléments suivants, dans n'importe quel ordre (1, n):
      - <attributeAlias>
        • <eventAttribute> (2, n)
      - <attributeName>
      - <computedValue>
  • <timeWindow> (1, 1)
    – L'un des deux élément suivants (1, 1) :
      - <timeInterval>
      - <runUntilDeactivated>
  • <onDetection> (0, 1)
    – <action> (0, n)
  • <onNextEvent> (0, 1)
    – <action> (0, n)
  • <onTimeWindowComplete> (0, 1)
    – <action> (0, n)

Récapitulatif relatif à la règle de filtrage

Ce récapitulatif répertorie tous les éléments de langage de la règle de filtrage.

Eléments de la règle

<filterRule> contient les éléments suivants :

- <comment> (0, 1)
- <variable> (0, n)
  - <comment> (0, 1)
  - <varInitializer> (1, 1)
- <activationInterval> (0, 1)
  - <activationTime> (0, 1)
    - <start> (0, 1)
      - L'un des trois éléments suivants (1, 1) :
        - <dateTime>
        - <whenLoaded>
        - <inactiveWhenLoaded>
    - <stop> (0, 1)
      - L'un des trois éléments suivants (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)

Récapitulatif relatif à la règle de séquence

Ce récapitulatif répertorie tous les éléments de langage de la règle de séquence.

Eléments de la règle

L'élément <sequenceRule> contient les éléments suivants :

- <comment> (0, 1)
- <variable> (0, n)
  - <comment> (0, 1)
  - <varInitializer> (1, 1)
- <activationInterval> (0, 1)
  - <activationTime> (0, 1)
    - <start> (0, 1)
      - L'un des trois élément suivants (1, 1) :
        - <dateTime>
        - <whenLoaded>
        - <inactiveWhenLoaded>
    - <stop> (0, 1)
      - L'un des trois élément suivants (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)
  - Les trois éléments suivants, dans n'importe quel ordre (1, n):
    - <attributeAlias>
      - <eventAttribute> (2, n)
    - <attributeName>
    - <computedValue>
- <timeWindow> (1, 1)
  - L'un des deux élément suivants (1, 1) :
    - <timeInterval>
    - <runUntilDeactivated>
- <onDetection> (0, 1)
  - <action> (0, n)
- <onTimeOut> (0, 1)
  - <action> (0, n)

Récapitulatif relatif à la règle de seuil

Ce récapitulatif répertorie tous les éléments de langage de la règle de seuil.

Eléments de la règle

<thresholdRule> contient les éléments suivants :

- <comment> (0, 1)
- <variable> (0, n)
  - <comment> (0, 1)
  - <varInitializer> (1, 1)
- <activationInterval> (0, 1)
  - <activationTime> (0, 1)
    - <start> (0, 1)
      - L'un des trois élément suivants (1, 1) :
        - <dateTime>
        - <whenLoaded>
        - <inactiveWhenLoaded>
    - <stop> (0, 1)
      - L'un des trois élément suivants (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)
  - Les trois éléments suivants, dans n'importe quel ordre (1, n):
    - <attributeAlias>
      - <eventAttribute> (2, n)
    - <attributeName>
    - <computedValue>
- L'un des trois élément suivants (1, 1) :
  - <booleanThreshold>
  - <computedThreshold>
  - <eventCountThreshold>
- <timeWindow> (1, 1)
  - L'un des deux élément suivants (1, 1) :
    - <timeInterval>
    - <runUntilDeactivated>
- <onDetection> (0, 1)
  - <action> (0, n)
- <onTimeOut> (0, 1)
  - <action> (0, n)

Récapitulatif relatif à la règle temporisée

Ce récapitulatif répertorie tous les éléments de langage de la règle temporisée.

Eléments de la règle

<timerRule> contient les éléments suivants :

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

  - <comment> (0, 1)
  - <varInitializer> (1, 1)
- <activationInterval> (0, 1)
  - <activationTime> (0, 1)
    - <start> (0, 1)
      - L'un des trois éléments suivants (1, 1) :
        - <dateTime>
        - <whenLoaded>
        - <inactiveWhenLoaded>
    - <stop> (0, 1)
      - L'un des trois éléments suivants (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)
  - L'un des deux élément suivants (1, 1) :
    - <timeInterval>
    - <runUntilDeactivated>
- <onTimeWindowComplete> (0, 1)
  - <action> (0, n)

Chapitre 5. Guide des éléments de langage

Ce guide décrit en détail les éléments de langage du schéma XML pour le langage de règle Active Correlation Technology. Les éléments de langage sont répertoriés dans l'ordre alphabétique et les attributs disponibles pour chaque élément sont décrits dans la rubrique relative à l'élément concerné.

Dans XML, et dans d'autres langages de balisage tels que SGML et HTML, un élément est une unité élémentaire qui comprend une balise d'ouverture, une balise de fermeture, des attributs associés et leurs valeurs, et tout texte contenu entre les balises d'ouverture et de fermeture. Un attribut est une paire valeur-nom qui est codée dans un élément pour définir une certaine caractéristique de cet élément. Un attribut possède un type de données qui identifie le type de l'information apportée dans sa valeur (par exemple, numérique, textuelle ou booléenne).

Dans le langage XML, un espace de nom est un identificateur URI qui fournit un nom unique à associer aux éléments et aux définitions de type d'un schéma. L'identificateur URI indique quel schéma XML contient la définition d'un élément. Un espace de nom est défini par une chaîne de préfixe suivie de deux points. Le schéma de langage de règle Active Correlation Technology est défini dans trois fichiers différents et utilise les trois espaces de nom suivants :

xsd: Cet espace de nom indique que l'élément de langage est défini dans le schéma XML standard décrit à l'adresse http://www.w3.org.

br: Cet espace de nom indique que l'élément de langage est défini dans le schéma de base du jeu de règles Active Correlation Technology situé dans le fichier ACTparser.jar, dans le sous-répertoire com/ibm/correlation/ruleparser/xml/RuleSetBase.xsd. Par exemple, `br:ruleSet` fait référence à l'élément ruleSet défini dans le fichier RuleSetBase.xsd.

act: Cet espace de nom indique que l'élément de langage est défini dans le schéma de langage Active Correlation Technology situé dans le fichier ACTparser.jar, dans le sous-répertoire com/ibm/correlation/ruleparser/xml/ACTL.xsd. Par exemple, `act:ruleSet` fait référence à l'élément ruleSet défini dans le fichier ACTL.xsd.

Dans le schéma de langage de règle, les éléments de langage sont définis soit comme des éléments, soit comme des types complexes, par exemple :

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

Dans le schéma, les attributs minOccurs et maxOccurs définissent respectivement le nombre mininum et maximum d'occurrences pour un élément de langage. tableau 11 explique la signification des différentes valeurs de attributs minOccurs et maxOccurs.

Tableau 11. Attributs du schéma qui définissent le nombre d'occurrences d'un élément de langage

Attribut	Valeur de l'attribut	Signification
minOccurs	0	L'élément de langage est facultatif.

Tableau 11. Attributs du schéma qui définissent le nombre d'occurrences d'un élément de langage (suite)

Attribut	Valeur de l'attribut	Signification
minOccurs	1	L'élément de langage doit apparaître au moins une fois. 1 est la valeur par défaut de l'attribut minOccurs.
minOccurs	2	L'élément de langage doit apparaître au moins deux fois.
maxOccurs	1	L'élément de langage ne peut apparaître plus d'une fois. 1 est la valeur par défaut de l'attribut maxOccurs.
maxOccurs	unbounded	L'élément de langage peut apparaître indéfiniment.

Elément action

L'élément <action> contient une expression qui définit une action de réponse à la règle ou de cycle de vie.

Détails

Voir «Variables», à la page 25 pour obtenir plus d'informations sur les variables qui peuvent être utilisées dans les expressions. L'emploi de certaines variables dépend du contexte d'expression.

Attributs

<action> possède les attributs suivants :

Tableau 12. Attributs de l'élément <action>

Nom	Description	Type de données	Obligatoire ?
expressionLanguage	Identifie le langage de programmation dans lequel l'expression est rédigée. Le langage de programmation Java étant le seul langage d'expression pris en charge, la seule valeur correcte pour cet attribut est `java`.	xsd:NMTOKEN	Oui
name	Identifie l'action. Cet identificateur peut servir à déterminer des incidents, en particulier s'il s'agit d'un nom unique parmi tous les éléments <action> définis pour une action spécifique de réponse à la règle ou de cycle de vie.	xsd:NMTOKEN	Non

Contenu dans

<action> est contenu dans les éléments suivants :

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

- <onTimeWindowComplete>
- <onUnload>

Contient

<action> ne contient aucun élément.

Concepts associés

«Expressions», à la page 20
Une expression est un code qui contient une logique personnalisée pouvant être ajoutée à une règle. Elle peut également accéder à un code externe au moteur Active Correlation Technology. Dans le langage de règle, les expressions sont valides uniquement dans des contextes ou éléments de langage de règle spécifiques.

Elément activateOnEvent

L'élément <activateOnEvent> définit les événements qui peuvent activer la règle ou, pour les règles comportant un élément <groupingKey>, une instance de règle.

Les trois modes de sélection possibles sont les suivants :

- Utilisation d'un ou de plusieurs éléments <eventType> avec un élément <filteringPredicate>
- Utilisation d'un ou de plusieurs éléments <eventType> sans élément <filteringPredicate>
- Utilisation d'un élément <filteringPredicate> sans élément <eventType>

Si la règle est inactive et qu'aucun élément <eventType> ou <filteringPredicate> n'est codé, tous les événements qui se produisent sont sélectionnés.

Ne pas coder d'élément <eventType> peut avoir un impact négatif sur les performances du système.

Supposons que vous souhaitiez sélectionner tous les événements de type `Audit Failure`. Vous pouvez utiliser un prédicat de filtrage pour restreindre davantage les critères de sélection et n'inclure que les événements dont l'un des attributs comporte une valeur donnée. Par exemple, vous coderiez un élément <eventType> pour sélectionner tous les événements de type `Audit Failure` et un élément <filteringPredicate> pour sélectionner uniquement les événements dont l'attribut de nom d'hôte comporte la valeur `MyCriticalSystem`.

Attributs

<activateOnEvent> ne possède aucun attribut.

Contenu dans

<activateOnEvent> est contenu dans les éléments suivants :

- <activationInterval>
- <activationByGroupingKey>

Contient

<activateOnEvent> contient les éléments suivants.

Ces éléments doivent être codés dans l'ordre indiqué. Il n'est pas nécessaire de coder les éléments facultatifs, mais tous les éléments codés doivent être placés dans le bon ordre.

Tableau 13. Eléments contenus dans l'élément <activateOnEvent>

Elément	Obligatoire ou facultatif ?
<eventType>	Facultatif. Le nombre d'occurrences autorisé est de 0 ou plus.
<filteringPredicate>	Facultatif. Le nombre d'occurrences autorisé est de 0 ou 1.
<stopAfter>	Cet élément est valide uniquement lorsque l'élément <activateOnEvent> est contenu dans l'élément <activationByGroupingKey>. Facultatif. Le nombre d'occurrences autorisé est de 0 ou 1.

Elément activationByGroupingKey

Il contient des éléments qui spécifient les événements pouvant activer et désactiver une instance de règle définie par <groupingKey>. L'élément <groupingKey> n'étant pas valide pour les règles temporisées et de filtrage, l'élément <activationByGroupingKey> ne s'applique pas à ces règles.

Détails

La fonction offerte par l'élément <activationByGroupingKey> s'applique aux règles dans lesquelles vous définissez une clé de groupement. Elle vous permet de contrôler l'activation et la désactivation des instances de règle en fonction de cette clé de groupement. Lorsque vous codez l'élément <activationByGroupingKey>, chaque instance de règle peut être activée ou désactivée individuellement, en fonction des conditions <activateOnEvent> et <deactivateOnEvent> contenues dans <activationByGroupingKey>.

L'exemple suivant illustre l'utilisation de l'élément <activationByGroupingKey> dans une règle de calcul.

- La règle de calcul suivante accepte les événements du type `StockSharesTraded`. Ces événements indiquent le nombre d'actions échangées pour de nombreuses entreprises différentes.
- La clé de groupement est l'attribut stockSymbol d'un événement. La valeur de l'attribut stockSymbol est le nom d'une entreprise spécifique.
- La valeur de l'attribut sharesTraded d'un événement correspond au nombre d'actions échangées pour l'entreprise concernée (cette dernière est indiquée par la valeur de l'attribut stockSymbol).
- Pour une entreprise spécifique, la règle de calcul crée un rapport qui indique le nombre d'actions échangées pour cette entreprise au cours d'un intervalle de dix minutes. Cependant, pour que la règle de calcul puisse créer ce rapport, elle doit recevoir un événement de type `StartReporting` dont l'attribut stockSymbol a pour valeur le nom de l'entreprise concernée.

```
<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>
```

Attributs

<activationByGroupingKey> ne possède aucun attribut.

Contenu dans

<activationByGroupingKey> est contenu dans l'élément suivant :

- <activationInterval>

Contient

<activationByGroupingKey> contient les éléments suivants.

Ces éléments doivent être codés dans l'ordre indiqué. Il n'est pas nécessaire de coder les éléments facultatifs, mais tous les éléments codés doivent être placés dans le bon ordre.

Tableau 14. Eléments contenus dans l'élément <activationByGroupingKey>

Elément	Obligatoire ou facultatif ?
<activateOnEvent>	Facultatif. Le nombre d'occurrences autorisé est de 0 ou 1.
<deactivateOnEvent>	Facultatif. Le nombre d'occurrences autorisé est de 0 ou 1.

Relations entre les éléments <activationInterval> et <activationByGroupingKey>

Les éléments <activateOnEvent> et <deactivateOnEvent> sont contenus dans ces deux éléments :

- <activationInterval>
- <activationByGroupingKey>, qui est lui-même contenu dans <activationInterval>

Le comportement d'une règle peut varier d'une part selon son activité en cours, et d'autre part selon les interactions entre les définitions de <activateOnEvent> et de <deactivateOnEvent> dans les éléments <activationInterval> et <activationByGroupingKey>. L'exemple suivant illustre les différentes interactions possibles entre ces définitions.

Dans cet exemple, une règle de duplication est définie afin de supprimer des événements sur des systèmes qui sont en mode maintenance, puis de créer, à la fin de la maintenance, un état récapitulatif du nombre d'événements supprimés.

Par défaut, une règle contenant une clé de groupement définie autorise le traitement de toutes les valeurs de clé de groupement. Par conséquent, lorsque des événements répondent aux critères définis pour la règle, toutes les instances de règle sont actives et prêtes à accepter ces événements selon toutes les valeurs de clé de groupement. L'intervalle d'activation de la règle est le même que celui qui aurait prévalu s'il n'y avait pas eu de clé de groupement. En effet, par essence, tous les événements qui répondent aux critères de sélection définis pour la règle sont traités.

Dans l'exemple suivant, la clé de groupement est `hostname` et les définitions contenues dans l'élément <activationInterval> spécifient les actions suivantes :

1. Activer toutes les instances de règle lors de la réception d'un événement de type `StartMaintenanceModeAllHosts`.
2. Désactiver toutes les instances de règle au bout de deux heures ou lors de la réception d'un événement de type `StopMaintenanceModeAllHosts`.

```
<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>
  <igroupingKey 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>
```

Dans certaines situations, il se peut que vous souhaitiez décider des instances de règle particulières à activer et du moment de leur activation. Pour ces situations, vous devez coder l'élément <activationByGroupingKey>.

L'exemple suivant enrichit l'exemple précédent et illustre la manière dont vous pouvez utiliser la valeur de clé de groupement pour sélectionner les instances de règle à traiter. Les définitions contenues dans l'élément <activationByGroupingKey> spécifient les actions suivantes :

1. Autoriser le traitement des instances de règle pour des noms d'hôte spécifiques lors de la réception d'événements de type `StartMaintenanceMode` pour ces noms d'hôte.
2. Désactiver ces instances de règle après deux heures d'activation ou lors de la réception d'un événement de type `StopMaintenanceMode` pour le nom d'hôte concerné.

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

Les affirmations suivantes résument ce qui se produit lorsque vous codez l'élément <activationByGroupingKey> :

- Lorsque l'élément <activateOnEvent> est codé dans l'élément <activationByGroupingKey>, sont seuls autorisés à être traités les événements ayant la même valeur de clé de groupement que celle de l'événement qui répond à la condition <activationByGroupingKey> <activateOnEvent>.
- Lorsque l'élément <deactivateOnEvent> est codé dans l'élément <activationByGroupingKey>, les événements dont la valeur de clé de groupement est la même que celle de l'événement qui répond à la condition <activationByGroupingKey> <deactivateOnEvent> *ne sont pas* autorisés à être traités.

Effet des différentes définitions d'activation et de désactivation sur l'état d'une règle

Les tableaux tableau 15, à la page 66 et tableau 16, à la page 68 montrent l'effet des différentes définitions d'activation et de désactivation sur l'état d'une règle. Ces tableaux utilisent les conventions suivantes :

- *A* est un événement d'activation.
- Dans la notation «*A[x]*, » *x* représente la valeur de clé de groupement. Par exemple, *A[1]* est un événement d'activation dont la valeur de clé de groupement est 1.
- *D* est un événement de désactivation.
- Dans la notation «*D[x]*, » *x* représente la valeur de clé de groupement. Par exemple, *D[1]* est un événement de désactivation dont la valeur de clé de groupement est 1.

Tableau 15. Changements d'état d'une règle en fonction des différentes définitions d'activation

Etat initial de la règle	**L'état de la règle est potentiellement affecté par**	**Etat final de la règle**
Inactif	L'heure définie dans <activationInterval> <activationTime> <start>	1. La règle est activée. 2. Les actions <onActivation> sont exécutées. 3. Toutes les valeurs de clé de groupement sont autorisées.
	La méthode activate()	
	L'événement *A*, défini dans <activationInterval> <activateOnEvent>	
	L'événement *A[1]*, défini dans <activationByGroupingKey> <activateOnEvent> (*sans* <stopAfter>)	1. La règle est activée. 2. Les actions <onActivation> sont exécutées. 3. Seule la valeur de clé de groupement 1 est autorisée. Lorsque cette instance de règle correspond au modèle de règle, la valeur de clé de groupement 1 n'est plus autorisée.
	L'événement *A[2]*, défini dans <activateOnEvent> (*avec* <stopAfter>)	1. La règle est activée. 2. Les actions <onActivation> sont exécutées. 3. Seule la valeur de clé de groupement 2 est autorisée, et uniquement pour la durée spécifiée après l'élément <stopAfter>. Au cours de cette durée, cette instance de règle peut correspondre plusieurs fois au modèle de règle.
• Actif • Autorisant toutes les valeurs de clé de groupement	L'heure définie dans <activationInterval> <activationTime> <start>	L'état de la règle n'a connu aucun changement. Il est identique à l'état initial.
	La méthode activate()	
	L'événement *A*, défini dans <activationInterval> <activateOnEvent>	
	L'événement *A[1]*, défini dans <activationByGroupingKey> <activateOnEvent> (*sans* <stopAfter>)	
	L'événement *A[2]*, défini dans <activateOnEvent> (*avec* <stopAfter>)	

Tableau 15. Changements d'état d'une règle en fonction des différentes définitions d'activation (suite)

Etat initial de la règle	**L'état de la règle est potentiellement affecté par**	**Etat final de la règle**
• Actif • Autorisant toutes les valeurs de clé de groupement qui ont déclenché les instances de règle en fonction des définitions <activationByGroupingKey> <activateOnEvent>	L'heure définie dans <activationInterval> <activationTime> <start>	L'état de la règle n'a connu aucun changement. Il est identique à l'état initial.
	La méthode activate()	
	L'événement *A*, défini dans <activationInterval> <activateOnEvent>	Toutes les valeurs de clé de groupement sont autorisées.
	L'événement *A[1]*, défini dans <activationByGroupingKey> <activateOnEvent> (*sans* <stopAfter>)	• La valeur de clé de groupement 1 est autorisée, en plus des valeurs précédemment autorisées. • Lorsque cette instance de règle correspond au modèle de règle, la valeur de clé de groupement 1 n'est plus autorisée.
	L'événement *A[2]*, défini dans <activateOnEvent> (*avec* <stopAfter>)	• La valeur de clé de groupement 2 est désormais autorisée, en plus des valeurs précédemment autorisées. • Elle l'est uniquement pour la durée spécifiée après l'élément <stopAfter>. • Au cours de cette durée, cette instance de règle peut correspondre plusieurs fois au modèle de règle.
• Actif • Autorisant toutes les clés de groupement, exceptées celles qui ne sont pas autorisées par les définitions <activationByGroupingKey> <deactivateOnEvent>	L'heure définie dans <activationInterval> <activationTime> <start>	L'état de la règle n'a connu aucun changement. Il est identique à l'état initial.
	La méthode activate()	
	L'événement *A*, défini dans <activationInterval> <activateOnEvent>	Toutes les valeurs de clé de groupement sont autorisées.
	L'événement *A[1]*, défini dans <activationByGroupingKey> <activateOnEvent> (*sans* <stopAfter>)	La valeur de clé de groupement 1 est autorisée, en plus des valeurs précédemment autorisées.
	L'événement *A[2]*, défini dans <activateOnEvent> (*avec* <stopAfter>)	La valeur de clé de groupement 2 est désormais autorisée, en plus des valeurs précédemment autorisées.

Tableau 16. Changements d'état d'une règle en fonction des différentes définitions de désactivation

Etat initial de la règle	**L'état de la règle est potentiellement affecté par**	**Etat final de la règle**
Inactif	L'heure définie dans <activationInterval> <activationTime> <stop>	L'état de la règle n'a connu aucun changement. Il est identique à l'état initial.
	La méthode deactivate()	
	L'événement *D*, défini dans <activationInterval> <deactivateOnEvent>	
	L'événement *D[1]*, défini dans <activationByGroupingKey> <deactivateOnEvent>	
	La durée définie dans <activationByGroupingKey> <activateOnEvent> <stopAfter> se termine pour l'événement *A[2]*	
• Actif • Autorisant toutes les valeurs de clé de groupement	L'heure définie dans <activationInterval> <activationTime> <stop>	1. Toutes les instances de règle sont désactivées. 2. Les actions <onDeactivation> sont exécutées. 3. La règle est désactivée.
	La méthode deactivate()	
	L'événement *D*, défini dans <activationInterval> <deactivateOnEvent>	
	L'événement *D[1]*, défini dans <activationByGroupingKey> <deactivateOnEvent>	• La valeur de clé de groupement 1 n'est plus autorisée. • Si l'instance de règle comportant une valeur de clé de groupement 1 est active, elle est désactivée.
	La durée définie dans <activationByGroupingKey> <activateOnEvent> <stopAfter> se termine pour l'événement *A[2]*	L'instance de règle comportant une valeur de clé de groupement 2 est désactivée.
• Actif • Autorisant toutes les valeurs de clé de groupement qui ont déclenché les instances de règle en fonction des définitions <activationByGroupingKey> <activateOnEvent>	L'heure définie dans <activationInterval> <activationTime> <stop>	1. Toutes les instances de règle sont désactivées. 2. Les actions <onDeactivation> sont exécutées. 3. La règle est désactivée.
	La méthode deactivate()	
	L'événement *D*, défini dans <activationInterval> <deactivateOnEvent>	
	L'événement *D[1]*, défini dans <activationByGroupingKey> <deactivateOnEvent>	• La valeur de clé de groupement 1 n'est plus autorisée. • Si l'instance de règle comportant une valeur de clé de groupement 1 est active, elle est désactivée.
	La durée définie dans <activationByGroupingKey> <activateOnEvent> <stopAfter> se termine pour l'événement *A[2]*	• La valeur de clé de groupement 2 n'est plus autorisée. • L'instance de règle comportant une valeur de clé de groupement 2 est désactivée.

Tableau 16. Changements d'état d'une règle en fonction des différentes définitions de désactivation (suite)

Etat initial de la règle	L'état de la règle est potentiellement affecté par	Etat final de la règle
• Actif • Autorisant toutes les clés de groupement, exceptées celles qui ne sont pas autorisées par les définitions <activationByGroupingKey> <deactivateOnEvent>	L'heure définie dans <activationInterval> <activationTime> <stop>	1. Toutes les instances de règle sont désactivées. 2. Les actions <onDeactivation> sont exécutées. 3. La règle est désactivée.
	La méthode deactivate()	
	L'événement *D*, défini dans <activationInterval> <deactivateOnEvent>	
	L'événement *D[1]*, défini dans <activationByGroupingKey> <deactivateOnEvent>	• La valeur de clé de groupement 1 n'est plus autorisée. • Si l'instance de règle comportant une valeur de clé de groupement 1 est active, elle est désactivée.
	La durée définie dans <activationByGroupingKey> <activateOnEvent> <stopAfter> se termine pour l'événement *A[2]*	L'instance de règle comportant une valeur de clé de groupement 2 est désactivée.

Elément activationInterval

<activationInterval> contient des éléments qui définissent le moment où une règle est active ou inactive.

Détails

Une règle peut être activée ou désactivée à un moment précis dans le temps ou sous l'effet d'un événement spécifique.

Si vous spécifiez qu'une règle doit être activée ou désactivée à un moment précis dans le temps *et* sous l'effet d'un événement spécifique, cette règle sera alors activée ou désactivée par ce qui survient en premier, à savoir le moment précis dans le temps ou la réception de l'événement. Toutefois, la règle peut dans ce cas être activée ou désactivée par une multitude d'événements tout au long de son cycle de vie. Par exemple, elle peut être activée par un événement, désactivée, activée à un moment défini dans le temps, désactivée à nouveau, puis activée par un autre événement.

Dans un environnement d'affaires, vous pouvez par exemple être amené à activer une règle suite à la réception d'un événement indiquant l'ouverture des opérations boursières. Dans un environnement informatique, vous pouvez être amené à démarrer une plage de maintenance à 06h00 le 29 octobre 2005 et à la terminer à l'une des heures suivantes, en fonction de l'élément qui se produit en premier :

- A 11h30 le 30 octobre 2005
- Suite à la réception d'un événement qui indique que la maintenance est achevée

Attributs

<activationInterval> ne possède aucun attribut.

Contenu dans

<activationInterval> est contenu dans les éléments suivants :

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

Contient

<activationInterval> contient les éléments suivants.

Ces éléments doivent être codés dans l'ordre indiqué. Il n'est pas nécessaire de coder les éléments facultatifs, mais tous les éléments codés doivent être placés dans le bon ordre.

Tableau 17. Eléments contenus dans <activationInterval>

Elément	Obligatoire ou facultatif ?
<activationTime>	Facultatif. Le nombre d'occurrences autorisé est de 0 ou 1.
<activateOnEvent>	Facultatif. Le nombre d'occurrences autorisé est de 0 ou 1.
<deactivateOnEvent>	Facultatif. Le nombre d'occurrences autorisé est de 0 ou 1.
<activationByGroupingKey>	Facultatif. Le nombre d'occurrences autorisé est de 0 ou 1.

Relations entre les éléments contenus

Les éléments <start> et <stop> contenus dans <activationTime> constituent une méthode statique d'activation et de désactivation d'une règle. Ils permettent d'activer ou de désactiver une règle à un moment précis dans le temps. A l'inverse, les éléments <activateOnEvent> et <deactivateOnEvent> constituent une méthode dynamique d'activation et de désactivation d'une règle. Ils permettent de l'activer ou de la désactiver lors de l'occurrence d'un événement donné. Par exemple, une règle est activée par tout événement qui répond aux critères définis pour l'élément <activateOnEvent>, si elle n'est pas déjà active. Elle est désactivée par tout événement répondant aux critères définis pour l'élément <deactivateOnEvent>, si elle n'est pas déjà inactive. Ainsi, certains événements peuvent altérer la définition statique du moment où une règle doit être activée ou désactivée.

Le tableau 18, à la page 71 décrit le moment et le mode d'activation et de désactivation d'une règle, en fonction de certaines combinaisons pouvant comporter le codage des éléments suivants :

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

Dans le tableau 18, à la page 71, *X* désigne le nom d'un événement qui active la règle et *Y*, le nom d'un événement qui la désactive.

Si l'élément <start> n'est pas codé du tout, l'heure de début par défaut correspond à celle qui est définie par l'élément <whenLoaded>.

Si l'élément <stop> n'est pas codé du tout, l'heure de fin par défaut correspond à celle qui est définie par l'élément <never>.

Tableau 18. Activité d'une règle selon différentes combinaisons de codage des éléments contenus dans <activationInterval>

<activationTime>				
<start>	**<stop>**	**<activateOnEvent>**	**<deactivateOnEvent>**	**Activité de la règle**
<whenLoaded>	<never>			La règle est active lors de son chargement, puis reste active au cours de l'exécution du moteur Active Correlation Technology.
<whenLoaded>	<never>		*Y*	La règle est active lors de son chargement. L'événement *Y* la désactive.
<whenLoaded>	<never>	*X*	*Y*	La règle est active lors de son chargement. L'événement *Y* la désactive et l'événement *X* la réactive. La désactivation et la réactivation peuvent se produire plusieurs fois.
<whenLoaded>	<after>			La règle est active lors de son chargement, puis désactivée après un intervalle de temps spécifié.
<whenLoaded>	<dateTime>			La règle est active lors de son chargement, puis désactivée à une date et une heure spécifiées.
<inactiveWhenLoaded>	<never>	*X*		La règle est inactive lors de son chargement. Elle est activée par l'événement *X*, puis reste active au cours de l'exécution du moteur Active Correlation Technology.
<inactiveWhenLoaded>	<never>	*X*	*Y*	La règle est inactive lors de son chargement. L'événement *X* active la règle et l'événement *Y* la désactive. L'activation et la désactivation peuvent se produire plusieurs fois.
<dateTime>	<dateTime>			Une date et une heure sont spécifiées pour l'activation et la désactivation de la règle.
<dateTime>	<dateTime>	*X*	*Y*	Une date et une heure sont spécifiées pour l'activation et la désactivation de la règle. L'événement *X* active la règle et l'événement *Y* la désactive. Les événements *X* et *Y* peuvent activer et désactiver la règle plusieurs fois.
<dateTime>	<never>			La règle est activée à une date et une heure spécifiées, puis reste active au cours de l'exécution du moteur Active Correlation Technology.
<dateTime>	<never>		*Y*	La règle est activée à une date et une heure spécifiées. L'événement *Y* la désactive.
<dateTime>	<never>	*X*	*Y*	La règle est activée à une date et une heure spécifiées. L'événement *Y* la désactive et l'événement *X* la réactive. La désactivation et la réactivation peuvent se produire plusieurs fois.
<dateTime>	<after>			La règle est activée à une date et une heure spécifiées, et désactivée après un intervalle de temps spécifié.
<dateTime>	<after>	*X*	*Y*	La règle est activée à une date et une heure spécifiées, et désactivée après un intervalle de temps spécifié. L'événement *X* active la règle et l'événement *Y* la désactive. L'activation et la désactivation peuvent se produire plusieurs fois.

Elément activationTime

L'élément <activationTime> définit des moments précis dans le temps où une règle est activée ou désactivée.

Attributs

<activationTime> ne possède aucun attribut.

Contenu dans

<activationTime> est contenu dans l'élément suivant :

- <activationInterval>

Contient

<activationTime> contient les éléments suivants.

Ces éléments doivent être codés dans l'ordre indiqué. Il n'est pas nécessaire de coder les éléments facultatifs, mais tous les éléments codés doivent être placés dans le bon ordre.

Tableau 19. Eléments contenus dans <activationTime>

Elément	Obligatoire ou facultatif ?
<start>	Facultatif. Le nombre d'occurrences autorisé est de 0 ou 1.
<stop>	Facultatif. Le nombre d'occurrences autorisé est de 0 ou 1.

Elément after

L'élément <after> spécifie la durée pendant laquelle la règle doit rester active après son activation. Au terme de cette durée, la règle doit être désactivée.

Attributs

<after> possède les attributs suivants :

Tableau 20. Attributs de l'élément <after>

Nom	Description	Type de données	Obligatoire ?
duration	Spécifie la quantité de temps pour la durée. Le type de données de cet attribut dépend de la valeur de l'attribut unit.	• Si la valeur de l'attribut unit est `ISO-8601`, ce type de données est xsd:duration. • Si la valeur de l'attribut unit est `milliseconds`, ce type de données est xsd:positiveInteger.	Oui
unit	Spécifie l'unité de temps à utiliser. Les valeurs correctes pour cet attribut sont : • `ISO-8601` • `milliseconds`	xsd:string	Oui

Utilisation de la norme ISO 8601 pour la durée

Lorsque le code `ISO-8601` est utilisé comme valeur de l'attribut unit (unité), la valeur de l'attribut duration (durée) utilise la norme ISO 8601 pour spécifier la durée sous forme de chaîne unique. La spécification du type de données selon le schéma XML standard utilise la norme ISO 8601 pour fournir un type de données appelé `duration`. Ce type de données est décrit plus en détail à l'adresse http://www.w3.org/TR/xmlschema-2/#duration.

La chaîne suivante correspond au format du type de données `duration` utilisé dans le schéma XML standard :

`PnYnMnDTnHnMnS`

- `P` est le caractère qui apparaît toujours en début de chaîne.
- *n*`Y` représente le nombre d'années. Une année équivaut à 365 jours. Ainsi, coder `1Y` revient à coder `365D`.
- *n*`M` représente le nombre de mois. Un mois équivaut à 30 jours. Ainsi, coder `1M` revient à coder `30D`.
- *n*`D` représente le nombre de jours.
- `T` est un séparateur placé entre les unités de date (années, mois et jour) et les unités d'heure (heures, minutes et secondes). Les unités d'heure suivent toujours le caractère `T`.
- *n*`H` représente le nombre d'heures.
- *n*`M` représente le nombre de mois.
- *n*`S` représente le nombre de secondes.

Exemples de ce format :

- `P5DT12H` indique 5 jours et demi.
- `PT59M59S` indique 59 minutes et 59 secondes.
- `P1M` indique un mois.

Contenu dans

<after> est contenu dans l'élément suivant :

- <stop>

Contient

<after> ne contient aucun élément.

Elément attributeAlias

L'élément <attributeAlias> donne un nom d'alias qui permet d'associer entre eux les attributs d'événement possédant la même signification dans différents événements, tout en portant des noms différents. Par exemple, pour l'attribut d'événement indiquant le nom du système à l'origine de l'événement, il se peut que trois événements différents utilisent ces trois noms différents : ″host″, ″hostname″ et ″source″. <attributeAlias> contient les éléments <eventAttribute> qui décrivent les attributs d'événement à associer en tant qu'attribut unique pour la clé de groupement.

Détails

L'élément <attributeAlias> et son attribut aliasName ne sont valides que dans le contexte d'une clé de groupement. Cet élément et son attribut ne peuvent être référencés dans aucune expression, y compris une expression comportant l'élément <computedValue>.

Attributs

<attributeAlias> possède les attributs suivants :

Tableau 21. Attributs de l'élément <attributeAlias>

Nom	Description	Type de données	Obligatoire ?
aliasName	Définit le nom des attributs d'événement décrits dans les éléments <eventAttribute> et devant être associés en tant qu'attribut unique pour la clé de groupement. Ce nom doit être unique dans la règle.	xsd:NMTOKEN	Oui

Contenu dans

<attributeAlias> est contenu dans l'élément suivant :

- <groupingKey>

Contient

<attributeAlias> contient l'élément suivant :

Tableau 22. Eléments contenus dans <attributeAlias>

Elément	Obligatoire ou facultatif ?
<eventAttribute>	2 occurrences de cet élément sont obligatoires. Les occurrences supplémentaires sont autorisées.

Elément attributeName

L'élément <attributeName> contient le nom d'un attribut d'événement spécifique qui fait partie de la clé de groupement. Ce nom doit correspondre à celui qui est utilisé dans l'appel de la méthode getAttribute pour la variable act_event.

Attributs

<attributeName> ne possède aucun attribut.

Contenu dans

<attributeName> est contenu dans l'élément suivant :

- <groupingKey>

Contient

<attributeName> ne contient aucun élément.

Elément booleanThreshold

L'élément <booleanThreshold> est valide uniquement pour la règle de seuil. Il contient une expression qui est appelée à chaque réception d'événement. Cette expression calcule ou compare la valeur de seuil en fonction de l'événement en cours et de tout autre événement ayant été accepté par la règle. Elle renvoie une valeur booléenne `true` ou `false` pour indiquer si le seuil a été atteint ou non.

Détails

Voir «Variables», à la page 25 pour obtenir plus d'informations sur les variables qui peuvent être utilisées dans les expressions. L'emploi de certaines variables dépend du contexte d'expression.

Attributs

<booleanThreshold> possède les attributs suivants :

Tableau 23. Attributs de l'élément <booleanThreshold>

Nom	Description	Type de données	Obligatoire ?
expressionLanguage	Identifie le langage de programmation dans lequel l'expression est rédigée. Le langage de programmation Java étant le seul langage d'expression pris en charge, la seule valeur correcte pour cet attribut est `java`.	xsd:NMTOKEN	Oui

Contenu dans

<booleanThreshold> est contenu dans l'élément suivant :

- <thresholdRule>

Contient

<booleanThreshold> ne contient aucun élément.

Concepts associés

«Expressions», à la page 20
Une expression est un code qui contient une logique personnalisée pouvant être ajoutée à une règle. Elle peut également accéder à un code externe au moteur Active Correlation Technology. Dans le langage de règle, les expressions sont valides uniquement dans des contextes ou éléments de langage de règle spécifiques.

Elément collectionRule

L'élément <collectionRule> définit une règle selon le modèle de collecte.

Attributs

<collectionRule> possède les attributs suivants :

Tableau 24. Attributs de l'élément <collectionRule>

Nom	Description	Type de données	Obligatoire ?
name	Identifie la règle. Cet identificateur doit être unique dans le bloc de règles qui contient la règle. Il ne doit pas contenir de point.	xsd:NMTOKEN	Oui

Tableau 24. Attributs de l'élément <collectionRule> (suite)

Nom	Description	Type de données	Obligatoire ?
processOnlyForwardedEvents	Spécifie si la règle reçoit tous les événements ou seulement ceux qui sont acheminés depuis d'autres règles. La valeur par défaut est `false`, ce qui indique que la règle reçoit tous les événements, y compris ceux qui sont acheminés depuis d'autres règles.	xsd:boolean	Non

Contenu dans

<collectionRule> est contenu dans l'élément suivant :

- <ruleBlock>

Contient

<collectionRule> contient les éléments suivants.

Ces éléments doivent être codés dans l'ordre indiqué. Il n'est pas nécessaire de coder les éléments facultatifs, mais tous les éléments codés doivent être placés dans le bon ordre.

Tableau 25. Eléments contenus dans <collectionRule>

Elément	Obligatoire ou facultatif ?
<comment>	Facultatif. Le nombre d'occurrences autorisé est de 0 ou 1.
<variable>	Facultatif. Le nombre d'occurrences autorisé est de 0 ou plus.
<activationInterval>	Facultatif. Le nombre d'occurrences autorisé est de 0 ou 1.
<lifeCycleActions>	Facultatif. Le nombre d'occurrences autorisé est de 0 ou 1.
<eventSelector>	Facultatif. Le nombre d'occurrences autorisé est de 0 ou 1.
<groupingKey>	Facultatif. Le nombre d'occurrences autorisé est de 0 ou 1.
<timeWindow>	Obligatoire. Une seule occurrence autorisée.
<onTimeWindowComplete>	Facultatif. Le nombre d'occurrences autorisé est de 0 ou 1.

Concepts associés

«Modèle de collecte», à la page 11
Une règle de collecte est définie par le modèle de collecte. Elle collecte un groupe d'événements sélectionnés au cours d'un intervalle de temps. Il s'agit d'une règle avec état.

Elément comment

L'élément <comment> peut comporter une description de la fonction et de l'objectif du jeu de règles, du bloc de règles, de la règle ou de la variable qui le contient.

Attributs

<comment> ne possède aucun attribut.

Contenu dans

<comment> est contenu dans les éléments suivants :

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

Contient

<comment> ne contient aucun élément.

Elément computationRule

L'élément <computationRule> définit une règle selon le modèle de calcul.

Attributs

<computationRule> possède les attributs suivants :

Tableau 26. Attributs de l'élément <computationRule>

Nom	Description	Type de données	Obligatoire ?
name	Identifie la règle. Cet identificateur doit être unique dans le bloc de règles qui contient la règle. Il ne doit pas contenir de point.	xsd:NMTOKEN	Oui

Tableau 26. Attributs de l'élément <computationRule> (suite)

Nom	Description	Type de données	Obligatoire ?
processOnlyForwardedEvents	Spécifie si la règle reçoit tous les événements ou seulement ceux qui sont acheminés depuis d'autres règles. La valeur par défaut est `false`, ce qui indique que la règle reçoit tous les événements, y compris ceux qui sont acheminés depuis d'autres règles.	xsd:boolean	Non

Contenu dans

<computationRule> est contenu dans l'élément suivant :

- <ruleBlock>

Contient

<computationRule> contient les éléments suivants.

Ces éléments doivent être codés dans l'ordre indiqué. Il n'est pas nécessaire de coder les éléments facultatifs, mais tous les éléments codés doivent être placés dans le bon ordre.

Tableau 27. Eléments contenus dans <computationRule>

Elément	Obligatoire ou facultatif ?
<comment>	Facultatif. Le nombre d'occurrences autorisé est de 0 ou 1.
<variable>	Facultatif. Le nombre d'occurrences autorisé est de 0 ou plus.
<activationInterval>	Facultatif. Le nombre d'occurrences autorisé est de 0 ou 1.
<lifeCycleActions>	Facultatif. Le nombre d'occurrences autorisé est de 0 ou 1.
<eventSelector>	Facultatif. Le nombre d'occurrences autorisé est de 0 ou 1.
<groupingKey>	Facultatif. Le nombre d'occurrences autorisé est de 0 ou 1.
<computeFunction>	Obligatoire. Une seule occurrence autorisée.
<timeWindow>	Obligatoire. Une seule occurrence autorisée.
<onTimeWindowComplete>	Facultatif. Le nombre d'occurrences autorisé est de 0 ou 1.

Concepts associés

«Modèle de calcul», à la page 12
Une règle de calcul est définie par le modèle de calcul. Elle s'applique à un calcul effectué sur des événements collectés (au moyen d'une expression), à chaque réception d'événement au cours d'un intervalle donné. Il s'agit d'une règle avec état.

Elément computedThreshold

L'élément <computedThreshold> est valide uniquement pour la règle de seuil. Il contient une expression qui, appelée à chaque réception d'événement, calcule ou compare la valeur de seuil en fonction de l'événement en cours et de tout autre événement ayant répondu aux critères de sélection de la règle. Cette expression renvoie la valeur de seuil calculée à stocker dans une variable définie pour la règle, qui utilise ensuite cette valeur pour la comparer à la valeur de seuil définie.

Détails

Voir «Variables», à la page 25 pour obtenir plus d'informations sur les variables qui peuvent être utilisées dans les expressions. L'emploi de certaines variables dépend du contexte d'expression.

Attributs

<computedThreshold> possède les attributs suivants :

Tableau 28. Attributs de l'élément <computedThreshold>

Nom	Description	Type de données	Obligatoire ?
expressionLanguage	Identifie le langage de programmation dans lequel l'expression est rédigée. Le langage de programmation Java étant le seul langage d'expression pris en charge, la seule valeur correcte pour cet attribut est `java`.	xsd:NMTOKEN	Oui
threshold	Définit la valeur de seuil à atteindre. Cette valeur de seuil définie doit être une valeur numérique représentée sous forme de chaîne et pouvant être convertie en type de données valide pour la variable de règle.	xsd:string	Oui

Tableau 28. Attributs de l'élément <computedThreshold> (suite)

Nom	Description	Type de données	Obligatoire ?
assignTo	Identifie le nom de la variable qui détient la valeur de seuil calculée renvoyée à partir de cette expression. Cette variable doit être préalablement définie pour la règle (au niveau du jeu de règles, du bloc de règles ou de la règle) à l'aide de l'élément <variable>. Elle doit comporter l'un des types de données numériques suivants : • java.lang.Double • java.lang.Float • java.lang.Integer • java.lang.Long • java.lang.String Si la variable est définie au niveau du jeu de règles ou du bloc de règles, elle n'est pas réinitialisée après que le modèle de règle a trouvé une correspondance.	xsd:NMTOKEN	Oui
thresholdComparison	Définit l'opérateur de comparaison entre la valeur de seuil calculée et la valeur de seuil définie. Les valeurs correctes pour cet opérateur sont : • `lessThan` • `lessThanOrEqualTo` • `greaterThan` • `greaterThanOrEqualTo`	xsd:string	Oui

Contenu dans

<computedThreshold> est contenu dans l'élément suivant :

- <thresholdRule>

Contient

<computedThreshold> ne contient aucun élément.

Concepts associés

«Expressions», à la page 20
Une expression est un code qui contient une logique personnalisée pouvant être ajoutée à une règle. Elle peut également accéder à un code externe au moteur Active Correlation Technology. Dans le langage de règle, les expressions sont valides uniquement dans des contextes ou éléments de langage de règle spécifiques.

Elément computedValue

L'élément <computedValue> contient une expression qui s'exécute lorsque la règle reçoit un événement lui indiquant de créer une chaîne en fonction de la valeur d'un ou plusieurs des attributs de cet événement. Cette chaîne peut ensuite être utilisée dans une clé de groupement.

Détails

Le rédacteur de règle peut être amené à utiliser ces éléments dans la clé de groupement :

- Une sous-chaîne de la valeur d'un attribut d'événement. Par exemple, si la valeur d'un attribut d'événement contient une adresse IP imbriquée, l'expression contenue dans l'élément <computedValue> pourrait extraire cette adresse comme valeur unique à utiliser dans la clé de groupement.
- Des sous-chaînes de valeurs provenant de différents attributs d'événement. Par exemple, l'expression contenue dans l'élément <computedValue> pourrait extraire ces sous-chaînes pour créer une valeur unique à utiliser dans la clé de groupement.

Si l'expression contenue dans l'élément <computedValue> renvoie une valeur NULL, la règle traite cette valeur comme une valeur d'attribut manquante.

Voir «Variables», à la page 25 pour obtenir plus d'informations sur les variables qui peuvent être utilisées dans les expressions. L'emploi de certaines variables dépend du contexte d'expression.

Attributs

<computedValue> possède les attributs suivants :

Tableau 29. Attributs de l'élément <computedValue>

Nom	Description	Type de données	Obligatoire ?
expressionLanguage	Identifie le langage de programmation dans lequel l'expression est rédigée. Le langage de programmation Java étant le seul langage d'expression pris en charge, la seule valeur correcte pour cet attribut est `java`.	xsd:NMTOKEN	Oui

Contenu dans

<computedValue> est contenu dans l'élément suivant :

- <groupingKey>

Contient

<computedValue> ne contient aucun élément.

Concepts associés

«Expressions», à la page 20
Une expression est un code qui contient une logique personnalisée pouvant être ajoutée à une règle. Elle peut également accéder à un code externe au moteur

Active Correlation Technology. Dans le langage de règle, les expressions sont valides uniquement dans des contextes ou éléments de langage de règle spécifiques.

Elément computeFunction

L'élément <computeFunction> est valide uniquement pour la règle de calcul. Il contient une expression qui, appelée à chaque réception d'événement, renvoie une valeur à stocker dans une variable définie pour la règle. La valeur renvoyée à partir de cette expression doit correspondre au type de données de la variable nommée dans l'attribut assignTo de l'élément <computeFunction>.

Détails

Voir «Variables», à la page 25 pour obtenir plus d'informations sur les variables qui peuvent être utilisées dans les expressions. L'emploi de certaines variables dépend du contexte d'expression.

Attributs

<computeFunction> possède les attributs suivants :

Tableau 30. Attributs de l'élément <computeFunction>

Nom	Description	Type de données	Obligatoire ?
expressionLanguage	Identifie le langage de programmation dans lequel l'expression est rédigée. Le langage de programmation Java étant le seul langage d'expression pris en charge, la seule valeur correcte pour cet attribut est `java`.	xsd:NMTOKEN	Oui
assignTo	Identifie le nom de la variable qui détient la valeur renvoyée à partir de cette expression. Cette variable doit être préalablement définie pour la règle (au niveau du jeu de règles, du bloc de règles ou de la règle) à l'aide de l'élément <variable>. Si la variable est définie au niveau du jeu de règles ou du bloc de règles, elle n'est pas réinitialisée après que le modèle de règle a trouvé une correspondance.	xsd:NMTOKEN	Oui

Contenu dans

<computeFunction> est contenu dans l'élément suivant :

- <computationRule>

Contient

<computeFunction> ne contient aucun élément.

Concepts associés

«Expressions», à la page 20
Une expression est un code qui contient une logique personnalisée pouvant être ajoutée à une règle. Elle peut également accéder à un code externe au moteur

Active Correlation Technology. Dans le langage de règle, les expressions sont valides uniquement dans des contextes ou éléments de langage de règle spécifiques.

Elément dateTime

L'élément <dateTime> définit la date et l'heure auxquelles une règle est activée ou désactivée. Cependant, la règle est activée ou désactivée seulement si elle a été chargée dans un moteur Active Correlation Technology en cours d'exécution avant l'heure qui a été définie.

Détails

Si la règle n'a pas été chargée dans un moteur Active Correlation Technology en cours d'exécution avant l'heure qui a été définie pour l'activation, elle ne sera jamais activée. Si la règle n'a pas été chargée dans un moteur Active Correlation Technology en cours d'exécution avant l'heure qui a été définie pour la désactivation, elle recevra l'état défini par l'élément <start> et ne sera jamais désactivée par l'élément <stop>.

L'élément <dateTime> doit contenir une chaîne qui respecte le format du type de données `dateTime` dans le schéma XML standard. Par exemple, `dateTime` comprend des séquences de caractères délimitées qui prennent la forme suivante :

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

- *yyyy* est un nombre à quatre chiffres ou plus représentant l'année. S'il y a plus de quatre chiffres, les zéros non significatifs sont interdits et `0000` est interdit.
- Les '-' restants sont des séparateurs entre les composants de la partie date.
- Le premier *mm* est un nombre à deux chiffres représentant le mois et commençant par `01`.
- *dd* est un nombre à deux chiffres représentant le jour du mois et commençant par `01`.
- `T` est un séparateur indiquant que l'heure du jour vient après.
- *hh* est un nombre à deux chiffres représentant l'heure du jour selon un système d'affichage de type 24 heures, commençant par`00` et finissant par `23`.
- `:` est un séparateur entre les composants de la partie heure du jour.
- Le second *mm* est un nombre à deux chiffres représentant la minute, commençant par `00` et finissant par `59`.
- *ss* est un nombre à deux chiffres représentant toutes les secondes. Il commence par`00` et finit par `59`.
- `'.'` *s*+, s'il apparaît, représente les secondes fractionnelles.
- *zzzzzz*, s'il apparaît, représente le fuseau horaire. Le fuseau horaire comprend des séquences de caractères délimitées sous la forme (('+' | '-') *hh* ':' *mm*) | 'Z', où :
  - '+', s'il apparaît, représente une durée non négative, et '-' ne doit pas apparaître.
  - '-', s'il apparaît, représente une durée non positive, et '+' ne doit pas apparaître.
  - *hh* est un nombre à deux chiffres représentant les heures. Il commence par `00` et finit par `14`.

– *mm* est un nombre à deux chiffres représentant les minutes. Il commence par `00` et finit par `59`. Cependant, si la valeur des heures est `14`, la valeur des minutes doit être `00`
– Z est une forme abrégée pour l'UTC (soit `+00:00` ou `-00:00`) ; il ne doit pas être accompagné d'autres éléments de fuseau horaire.

Voici deux exemples du contenu de l'élément <dateTime> :

- `2005-06-01T13:05:06.07` correspond au 1er juin 2005 à 13h05 6 secondes et 7 centièmes de seconde, heure locale.
- `2005-06-01T13:05:06.07Z` correspond au 1er juin 2005 à 13h05 6 secondes et 7 centièmes de seconde, heure UTC, ce qui équivaut au 1er juin à 9h05 6 secondes et 7 centièmes de seconde HAE (ou `2005-06-01T09:05:06.07-04:00`)

Attributs

<dateTime> ne possède aucun attribut.

Contenu dans

<dateTime> est contenu dans les éléments suivants :

- <start>
- <stop>

Contient

<dateTime> ne contient aucun élément.

Elément deactivateOnEvent

L'élément <deactivateOnEvent> définit les événements qui peuvent désactiver la règle ou, pour les règles définies avec un élément <groupingKey>, une instance de règle.

Les trois modes de sélection possibles sont les suivants :

- Utilisation d'un ou de plusieurs éléments <eventType> avec un élément <filteringPredicate>
- Utilisation d'un ou de plusieurs éléments <eventType> sans élément <filteringPredicate>
- Utilisation d'un élément <filteringPredicate> sans élément <eventType>

Si la règle est active et qu'aucun élément <eventType> ou <filteringPredicate> n'est codé, tous les événements produits sont sélectionnés.

Ne pas coder d'élément <eventType> peut avoir un impact négatif sur les performances du système.

Supposons que vous souhaitiez sélectionner tous les événements de type `Audit Failure`. Vous pouvez utiliser un prédicat de filtrage pour restreindre davantage les critères de sélection et n'inclure que les événements dont l'un des attributs comporte une valeur donnée. Par exemple, vous coderiez un élément <eventType> pour sélectionner tous les événements de type `Audit Failure` et un élément <filteringPredicate> pour sélectionner uniquement les événements dont l'attribut de nom d'hôte comporte la valeur `MyCriticalSystem`.

Attributs

<deactivateOnEvent> ne possède aucun attribut.

Contenu dans

<deactivateOnEvent> est contenu dans les éléments suivants :

- <activationInterval>
- <activationByGroupingKey>

Contient

<deactivateOnEvent> contient les éléments suivants.

Ces éléments doivent être codés dans l'ordre indiqué. Il n'est pas nécessaire de coder les éléments facultatifs, mais tous les éléments codés doivent être placés dans le bon ordre.

Tableau 31. Eléments contenus dans <deactivateOnEvent>

Elément	Obligatoire ou facultatif ?
<eventType>	Facultatif. Le nombre d'occurrences autorisé est de 0 ou plus.
<filteringPredicate>	Facultatif. Le nombre d'occurrences autorisé est de 0 ou 1.

Elément duplicateRule

L'élément <duplicateRule> définit une règle selon le modèle de duplication.

Attributs

<duplicateRule> possède les attributs suivants :

Tableau 32. Attributs de l'élément <duplicateRule>

Nom	Description	Type de données	Obligatoire ?
name	Identifie la règle. Cet identificateur doit être unique dans le bloc de règles qui contient la règle. Il ne doit pas contenir de point.	xsd:NMTOKEN	Oui
processOnlyForwardedEvents	Spécifie si la règle reçoit tous les événements ou seulement ceux qui sont acheminés depuis d'autres règles. La valeur par défaut est `false`, ce qui indique que la règle reçoit tous les événements, y compris ceux qui sont acheminés depuis d'autres règles.	xsd:boolean	Non

Contenu dans

<duplicateRule> est contenu dans l'élément suivant :

- <ruleBlock>

Contient

<duplicateRule> contient les éléments suivants.

Ces éléments doivent être codés dans l'ordre indiqué. Il n'est pas nécessaire de coder les éléments facultatifs, mais tous les éléments codés doivent être placés dans le bon ordre.

Tableau 33. Eléments contenus dans <duplicateRule>

Elément	Obligatoire ou facultatif ?
<comment>	Facultatif. Le nombre d'occurrences autorisé est de 0 ou 1.
<variable>	Facultatif. Le nombre d'occurrences autorisé est de 0 ou plus.
<activationInterval>	Facultatif. Le nombre d'occurrences autorisé est de 0 ou 1.
<lifeCycleActions>	Facultatif. Le nombre d'occurrences autorisé est de 0 ou 1.
<eventSelector>	Facultatif. Le nombre d'occurrences autorisé est de 0 ou 1.
<groupingKey>	Facultatif. Le nombre d'occurrences autorisé est de 0 ou 1.
<timeWindow>	Obligatoire. 1 seule occurrence autorisée.
<onDetection>	Facultatif. Le nombre d'occurrences autorisé est de 0 ou 1.
<onNextEvent>	Facultatif. Le nombre d'occurrences autorisé est de 0 ou 1.
<onTimeWindowComplete>	Facultatif. Le nombre d'occurrences autorisé est de 0 ou 1.

Concepts associés

«Modèle de duplication», à la page 12
Une règle de duplication est définie par le modèle de duplication. Elle compte le deuxième événement et les suivants acceptés dans l'intervalle de temps défini, tout en omettant leur traitement par le jeu de règles. Il s'agit d'une règle avec état.

Elément eventAttribute

L'élément <eventAttribute> permet d'associer un type d'événement et un attribut d'événement au nom d'alias d'attribut défini par <attributeAlias>.

Attributs

<eventAttribute> possède les attributs suivants :

Tableau 34. Attributs de l'élément <eventAttribute>

Nom	Description	Type de données	Obligatoire ?
type	Définit le nom du type d'événement. Il s'agit du nom utilisé pour l'attribut type dans l'élément <eventType>.	xsd:NMTOKEN	Oui
attributeName	Spécifie le nom qualifié complet de l'attribut d'événement qui est associé à d'autres attributs d'événement par le biais du nom d'alias d'attribut. Ce nom doit correspondre à celui qui est utilisé dans la variable act_event pour appeler la méthode getAttribute.	xsd:string	Oui

Contenu dans

<eventAttribute> est contenu dans l'élément suivant :

- <attributeAlias>

Contient

<eventAttribute> ne contient aucun élément.

Elément eventCountThreshold

L'élément <eventCountThreshold> est valide uniquement pour la règle de seuil. Il définit le nombre d'événements qui doivent répondre aux critères de sélection d'événement au cours d'une période donnée. L'élément <eventCountThreshold> spécifie également l'un des deux modes d'intervalle de temps disponibles pour la plage temporelle : fixe ou glissant.

Détails

Intervalle fixe (fixed)

Un intervalle fixe commence à la réception du premier événement qui répond aux critères de sélection, et s'achève dans l'un des cas suivants :

- La règle atteint son seuil au cours de la durée spécifiée.
- La durée spécifiée s'est écoulée.

Intervalle glissant (sliding)

Un intervalle glissant commence à la réception du premier événement qui correspond aux critères de sélection. Cependant, si la règle n'a pas atteint son seuil après écoulement de la durée spécifiée, la plage temporelle réajuste l'heure de début pour la faire glisser jusqu'à l'heure de réception d'un nouveau «premier» événement, qui est généralement l'événement accepté suivant. L'intervalle glissant poursuit ce réajustement jusqu'à que l'une des situations suivantes se produisent :

- La règle atteint son seuil au cours de la durée spécifiée.
- L'événement qui démarre la plage temporelle n'est suivi d'aucun autre événement au cours de la durée spécifiée.

L'événement qui démarre la plage temporelle (et devient le nouveau «premier» événement) est l'événement qui répond à ce critère : son heure de réception, ajoutée à la durée de l'intervalle de temps définie pour la règle, est supérieure à l'heure en cours. Nous obtenons l'équation suivante :

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

S'il n'existe aucun événement de ce type, l'intervalle glissant ne peut plus réajuster l'heure de début et s'achève.

La règle de seuil compte tous les événements acceptés jusqu'à ce que le seuil soit atteint ou que la période indiquée s'achève. Ensuite, elle exécute les actions définies dans l'élément <onDetection> ou <onTimeOut>, selon le cas.

Actions <onDetection>
Ces actions s'exécutent lorsque le comptage d'événements est égal à la valeur définie dans l'attribut threshold de l'élément <eventCountThreshold>, indiquant que le seuil est atteint.

Actions <onTimeOut>
L'heure d'exécution de ces actions dépend du mode d'intervalle de temps, qui peut être fixe ou glissant.

Mode fixed (fixe)
En mode fixe, ces actions s'exécutent à l'expiration de la plage temporelle.

Mode sliding (glissant)
En mode glissant, ces actions s'exécutent si l'événement qui démarre la plage temporelle n'est suivi d'aucun autre événement au cours de la durée spécifiée. En d'autres termes, aucun événement n'est reçu à une heure qui, ajoutée à la durée de l'intervalle de temps définie pour la règle, est supérieure à l'heure en cours.

Le mode d'intervalle de temps pour la plage temporelle est défini par l'attribut timeIntervalMode de l'élément <eventCountThreshold>. Le scénario suivant illustre le comportement des deux modes disponibles, ainsi que leurs différences.

Scénario illustrant les modes fixe et glissant

Supposons que la règle reçoive quatre événements correspondant aux critères de sélection, soit un à chacune de ces heures : 8h00, 8h04, 8h06 et 8h07. Le seuil de comptage d'événements est de 3 et la durée de la plage temporelle est fixée à 5 minutes.

Comportement de la règle en mode `fixed` (fixe)
Dans ce mode d'intervalle de temps, la règle de seuil commence le traitement à 8h00 et exécute les actions <onTimeOut> à 8h05, car elle ne reçoit que deux événements en 5 minutes. Par conséquent, elle n'atteint pas le seuil au cours de la plage temporelle spécifiée. Lorsqu'elle reçoit le troisième événement à 8h06, la règle de seuil commence le traitement de nouveau et exécute les actions <onTimeOut> à 8h11, car elle ne reçoit que deux événements en 5 minutes.

Le mode fixe est statique.

Comportement de la règle en mode `sliding` (glissant)
Dans ce mode d'intervalle de temps, la règle de seuil commence le

traitement à 8h00. A 8h05, heure où la plage temporelle doit s'achever, la règle détecte n'avoir reçu que deux événements. Elle ignore alors l'événement reçu à 8h00 et recalcule la durée, afin qu'elle se termine à 8h09 (le premier événement étant désormais celui de 8h04). Lorsque la règle reçoit l'événement de 8h07, elle exécute les actions <onDetection>, car elle a désormais atteint son seuil (3 événements à 8h04, 8h06 et 8h07) au cours de la dernière plage temporelle (8h04 – 8h09).

Le mode glissant est dynamique, car il réajuste (fait glisser) continuellement l'heure de début, afin que la règle atteigne le seuil défini au cours de la plage temporelle.

Supposons maintenant que la règle reçoive quatre événements correspondant aux critères de sélection, soit un à chacune de ces heures : 8h00, 8h04, 8h06 et 8h10. Le seuil de comptage d'événements est de 3 et la durée de la plage temporelle est fixée à 5 minutes.

Comportement de la règle en mode `sliding` (glissant)

Dans ce cas, la règle de seuil commence le traitement à 8h00. A 8h05, heure où la plage temporelle doit s'achever, la règle détecte n'avoir reçu que deux événements. Elle ignore alors l'événement reçu à 8h00 et recalcule la durée, afin qu'elle se termine à 8h09 (le premier événement étant désormais celui de 8h04).

A 8h09, heure où la plage temporelle doit désormais s'achever, la règle détecte n'avoir reçu que deux événements. Elle ignore alors l'événement reçu à 8h04 et recalcule la durée, afin qu'elle se termine à 8h11 (le premier événement étant désormais celui de 8h06).

A 8h11, heure où la plage temporelle doit désormais s'achever, la règle détecte n'avoir reçu que deux événements. Elle ignore alors l'événement reçu à 8h06 et recalcule la durée, afin qu'elle se termine à 8h15 (le premier événement étant désormais celui de 8h10).

A 8h15, heure où la plage temporelle doit désormais s'achever, la règle détecte n'avoir reçu aucun événement depuis celui de 8h10, le premier de la plage. Elle exécute alors les actions <onTimeOut>.

Attributs

<eventCountThreshold> possède les attributs suivants :

Tableau 35. Attributs de l'élément <eventCountThreshold>

Nom	Description	Type de données	Obligatoire ?
threshold	Définit le nombre d'événements qui doivent répondre aux critères de sélection d'événement au cours d'une période donnée. Il s'agit du seuil de nombres d'événements à atteindre. Cette valeur doit être un entier positif.	xsd:positiveInteger	Oui
timeIntervalMode	Définit si l'intervalle de temps pour la plage temporelle est fixe ou glissant. Les valeurs correctes pour cet attribut sont : • `fixed` (valeur par défaut) • `sliding`	xsd:string	Non

Contenu dans

<eventCountThreshold> est contenu dans l'élément suivant :

- <thresholdRule>

Contient

<eventCountThreshold> ne contient aucun élément.

Elément eventSelector

L'élément <eventSelector> définit les événements qui sont sélectionnés pour être traités par une règle.

Détails

Les trois modes de sélection possibles sont les suivants :

- Utilisation d'un ou de plusieurs éléments <eventType> avec un élément <filteringPredicate>
- Utilisation d'un ou de plusieurs éléments <eventType> sans élément <filteringPredicate>
- Utilisation d'un élément <filteringPredicate> sans élément <eventType>

Dans les cas spécifiques où vous souhaitez qu'une règle traite tous les événements, vous pouvez choisir l'une des options suivantes :

- Ne codez pas d'élément <eventSelector>.
- Codez un élément <eventSelector> qui ne contient aucun élément.

Ne pas coder d'élément <eventType> peut avoir un impact négatif sur les performances du système.

Supposons que vous souhaitiez sélectionner tous les événements de type `Audit Failure`. Vous pouvez utiliser un prédicat de filtrage pour restreindre davantage les critères de sélection et n'inclure que les événements dont l'un des attributs comporte une valeur donnée. Par exemple, vous coderiez un élément <eventType> pour sélectionner tous les événements de type `Audit Failure` et un élément <filteringPredicate> pour sélectionner uniquement les événements dont l'attribut de nom d'hôte comporte la valeur `MyCriticalSystem`.

Attributs

<eventSelector> possède les attributs suivants :

Tableau 36. Attributs de l'élément <eventSelector>

Nom	Description	Type de données	Obligatoire ?
alias	Cet attribut est valide uniquement dans une règle de séquence, la seule à comporter plusieurs éléments <eventSelector>. Il nomme de façon unique un événement sélectionné par un sélecteur d'événement donné dans la règle de séquence. Les prédicats de filtrage et les actions peuvent ensuite utiliser ce nom d'alias pour accéder à cet événement.	xsd:NMTOKEN	Non

Contenu dans

<eventSelector> est contenu dans les éléments suivants :

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

Contient

<eventSelector> contient les éléments suivants.

Ces éléments doivent être codés dans l'ordre indiqué. Il n'est pas nécessaire de coder les éléments facultatifs, mais tous les éléments codés doivent être placés dans le bon ordre.

Tableau 37. Eléments contenus dans <eventSelector>

Elément	Obligatoire ou facultatif ?
<eventType>	Facultatif. Le nombre d'occurrences autorisé est de 0 ou plus.
<filteringPredicate>	Facultatif. Le nombre d'occurrences autorisé est de 0 ou 1.

Elément eventType

L'élément <eventType> définit le type d'événement qui est sélectionné pour être traité par une règle, ou qui active ou désactive la règle.

Attributs

<eventType> possède les attributs suivants :

Tableau 38. Attributs de l'élément <eventType>

Nom	Description	Type de données	Obligatoire ?
type	Définit le type d'événement. Pour les événements conformes à la spécification Common Base Event, ce nom correspond à la valeur de l'attribut extensionName. Pour les événements IBM Tivoli Enterprise Console, il s'agit du nom de classe d'événement défini dans le fichier BAROC. Les événements d'autres formats sont susceptibles d'utiliser un attribut différent pour spécifier le type d'événement.	xsd:NMTOKEN	Oui

Contenu dans

<eventType> est contenu dans les éléments suivants :

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

Contient

<eventType> ne contient aucun élément.

Elément filteringPredicate

L'élément <filteringPredicate> contient une expression qui restreint davantage la sélection des événements qui doivent être traités par la règle, ou qui l'activent ou la désactivent. Ainsi, le filtrage des événements peut être encore plus précis que celui qui est effectué par type d'événement au moyen de l'élément <eventType>.

Détails

L'expression définit une condition et renvoie une valeur booléenne `true` (la condition est remplie) ou `false` (la condition n'est pas remplie).

Voir «Variables», à la page 25 pour obtenir plus d'informations sur les variables qui peuvent être utilisées dans les expressions. L'emploi de certaines variables dépend du contexte d'expression.

Attributs

<filteringPredicate> possède les attributs suivants :

Tableau 39. Attributs de l'élément <filteringPredicate>

Nom	Description	Type de données	Obligatoire ?
expressionLanguage	Identifie le langage de programmation dans lequel l'expression est rédigée. Le langage de programmation Java étant le seul langage d'expression pris en charge, la seule valeur correcte pour cet attribut est `java`.	xsd:NMTOKEN	Oui

Contenu dans

<filteringPredicate> est contenu dans les éléments suivants :

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

Contient

<filteringPredicate> ne contient aucun élément.

Concepts associés

«Expressions», à la page 20
Une expression est un code qui contient une logique personnalisée pouvant être ajoutée à une règle. Elle peut également accéder à un code externe au moteur Active Correlation Technology. Dans le langage de règle, les expressions sont valides uniquement dans des contextes ou éléments de langage de règle spécifiques.

Elément filterRule

L'élément <filterRule> définit une règle selon le modèle de filtrage.

Attributs

<filterRule> possède les attributs suivants :

Tableau 40. Attributs de l'élément <filterRule>

Nom	Description	Type de données	Obligatoire ?
name	Identifie la règle. Cet identificateur doit être unique dans le bloc de règles qui contient la règle. Il ne doit pas contenir de point.	xsd:NMTOKEN	Oui

Tableau 40. Attributs de l'élément <filterRule> (suite)

Nom	Description	Type de données	Obligatoire ?
processOnlyForwardedEvents	Spécifie si la règle reçoit tous les événements ou seulement ceux qui sont acheminés depuis d'autres règles. La valeur par défaut est `false`, ce qui indique que la règle reçoit tous les événements, y compris ceux qui sont acheminés depuis d'autres règles.	xsd:boolean	Non

Contenu dans

<filterRule> est contenu dans l'élément suivant :

- <ruleBlock>

Contient

<filterRule> contient les éléments suivants.

Ces éléments doivent être codés dans l'ordre indiqué. Il n'est pas nécessaire de coder les éléments facultatifs, mais tous les éléments codés doivent être placés dans le bon ordre.

Tableau 41. Eléments contenus dans <filterRule>

Elément	Obligatoire ou facultatif ?
<comment>	Facultatif. Le nombre d'occurrences autorisé est de 0 ou 1.
<variable>	Facultatif. Le nombre d'occurrences autorisé est de 0 ou plus.
<activationInterval>	Facultatif. Le nombre d'occurrences autorisé est de 0 ou 1.
<lifeCycleActions>	Facultatif. Le nombre d'occurrences autorisé est de 0 ou 1.
<eventSelector>	Facultatif. Le nombre d'occurrences autorisé est de 0 ou 1.
<onDetection>	Facultatif. Le nombre d'occurrences autorisé est de 0 ou 1.

Concepts associés

«Modèle de filtrage», à la page 13
Une règle de filtrage est définie par le modèle de filtrage. Elle applique une certaine action lorsqu'elle accepte un événement. Puisqu'elle agit sur un seul événement, c'est une règle sans état.

Elément groupingKey

Généralement, chaque règle active possède une instance de règle, ou copie, exécutée dans le moteur Active Correlation Technology. Cependant, une même règle est parfois nécessaire pour différents groupes d'événement souvent liés à différents groupes de ressources. La clé de groupement comporte un ou plusieurs attributs d'événement, ou parties d'attribut d'événement, qui peuvent être utilisés pour ordonner les événements sélectionnés en différents groupes, afin de permettre un traitement collectif unique pour chaque ensemble. L'élément <groupingKey> définit la clé de groupement d'une règle. Il a pour but de conduire une règle à créer une instance de règle (ou une copie) pour chaque groupe d'événements

possédant des caractéristiques communes (définies par les valeurs des attributs d'événement contenus dans l'élément <groupingKey>).

Détails

Les deux scénarios suivants illustrent l'importance de la clé de groupement.

Scénario 1 :
Deux événements se produisent, un événement DB2down et un événement DB2up. Le programme DB2 est exécuté sur trois ordinateurs, A, B et C. Une règle de séquence est définie pour corréler un événement DB2down à un événement DB2up et alerter l'opérateur lorsque le programme DB2 s'arrête et ne redémarre pas.

Si la règle de séquence est définie sans clé de groupement et qu'elle reçoit un événement DB2down provenant de l'ordinateur A, un événement DB2up provenant d'un ordinateur quelconque viendrait compléter la séquence, mais ne produirait pas l'effet voulu. Au contraire, si la clé de groupement était définie en tant qu'attribut de nom d'hôte, une copie ou une instance unique de la règle serait créée pour chaque valeur unique de cet attribut dans les événements sélectionnés. Le moteur Active Correlation Technology enverrait chaque événement vers l'instance de règle appropriée (celle qui correspondrait à la valeur de l'attribut de nom d'hôte de l'événement). Dès lors, si la règle recevait un événement DB2down provenant de l'ordinateur A, le moteur Active Correlation Technology créerait une instance de règle pour l'ordinateur A. Si elle recevait un événement DB2down provenant de l'ordinateur B, le moteur Active Correlation Technology créerait une seconde instance de règle pour l'ordinateur B. Un événement DB2up reçu depuis l'ordinateur B serait alors traité par le moteur dans la seconde instance de règle. La séquence serait complète et l'opérateur alerté, car les événements DB2down et DB2up reçus depuis l'ordinateur B seraient correctement corrélés.

Scénario 2 :
Un événement pour un message `Incorrect login attempted` (Tentative de connexion incorrecte) se produit pour tous les ordinateurs dans un environnement particulier. Cet événement contient un ID utilisateur. Une règle de seuil est définie de sorte qu'un avertissement soit envoyé à l'opérateur si cet événement se produit plus de 10 fois en 5 minutes.

Une clé de groupement peut être définie en tant qu'ID utilisateur. Une nouvelle instance de règle serait alors créée pour chaque ID utilisateur unique. Les tentatives de connexion de chaque utilisateur seraient suivies dans des instances de règle de seuil uniques, chaque instance procédant à un comptage à part pour l'utilisateur qui lui est associé. L'opérateur recevrait un avertissement si le nombre de connexions incorrectes d'un ID utilisateur était supérieur à 10 au cours d'un intervalle de 5 minutes.

Il existe d'autres variantes :

- La clé de groupement peut être définie comme nom d'hôte, plutôt que comme ID utilisateur. Cette option permettrait de détecter un grand nombre de tentatives de connexion incorrectes sur un seul ordinateur.
- La clé de groupement peut être définie comme une combinaison de nom d'hôte et d'ID utilisateur. Cette option permettrait de détecter une éventuelle tentative de piraterie par un ID utilisateur donné sur un ordinateur donné.

Si le même attribut figure dans tous les types d'événement spécifiés pour une règle, l'utilisation de l'élément <attributeName> est le moyen le plus simple et le

plus courant de définir une clé de groupement.

Attributs

<groupingKey> possède les attributs suivants :

Tableau 42. Attributs de l'élément <groupingKey>

Nom	Description	Type de données	Obligatoire ?
missingAttributeHandling	Définit l'action que la règle doit appliquer dans l'une des conditions suivantes : • Lorsqu'un événement sélectionné possède un attribut qui est contenu dans la clé de groupement, mais dont la valeur est manquante • Lorsque l'expression contenue dans l'élément <computedValue> renvoie une valeur NULL. La règle traite cette valeur comme une valeur d'attribut manquante. Les valeurs correctes pour l'attribut missingAttributeHandling sont : • `ignoreEvent` (valeur par défaut) : la règle ignore l'événement et n'applique aucune action. • `ignoreAttribute` : la règle accepte l'événement, mais ignore l'attribut dont la valeur est manquante. Le moteur Active Correlation Technology inclut alors une valeur de remplacement pour cet attribut.	xsd:string	Non

Contenu dans

<groupingKey> est contenu dans les éléments suivants :

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

Contient

<groupingKey> contient les éléments suivants.

Tableau 43. Eléments contenus dans <groupingKey>

Elément	Obligatoire ou facultatif ?
<attributeAlias>	Un de ces éléments est obligatoire. Le codage de plus d'un de ces éléments est facultatif. Plusieurs occurrences sont autorisées pour les trois éléments. Ces éléments peuvent être codés dans n'importe quel ordre.
<attributeName>	
<computedValue>	

Elément import

L'élément <import> contient une expression qui définit les modules externes (tels que des classes Java) à importer en vue de leur utilisation dans d'autres expressions contenues dans des règles.

Détails

Le code d'expression est une chaîne au sein de l'élément <import>. Le compilateur Active Correlation Technology utilise les instructions d'importation qui sont fournies par les éléments <import> pour compiler le code d'expression des règles qui appellent les méthodes externes.

Attributs

<import> possède les attributs suivants :

Tableau 44. Attributs de l'élément <import>

Nom	Description	Type de données	Obligatoire ?
expressionLanguage	Identifie le langage de programmation dans lequel l'expression est rédigée. Le langage de programmation Java étant le seul langage d'expression pris en charge, la seule valeur correcte pour cet attribut est `java`.	xsd:NMTOKEN	Oui

Contenu dans

<import> est contenu dans les éléments suivants :

- <ruleSet>
- <ruleBlock>

Contient

<import> ne contient aucun élément.

Concepts associés

«Importation et accès à des modules et des objets externes», à la page 21
Cet exemple indique comment vous pouvez rendre un code externe (classes

Java, par exemple) et des objets externes accessibles aux expressions. Un objet externe est un objet créé par une application pour communiquer avec des expressions.

Elément inactiveWhenLoaded

L'élément <inactiveWhenLoaded> spécifie qu'une règle est inactive lorsqu'elle est chargée par le moteur Active Correlation Technology. La règle reste inactive jusqu'à ce qu'elle soit activée par d'autres moyens.

Attributs

<inactiveWhenLoaded> ne possède aucun attribut.

Contenu dans

<inactiveWhenLoaded> est contenu dans l'élément suivant :

- <start>

Contient

<inactiveWhenLoaded> ne contient aucun élément.

Elément lifeCycleActions

L'élément <lifeCycleActions> contient des éléments qui définissent les actions à appliquer lors des quatre étapes principales du cycle de vie d'une règle.

Détails

Les actions définies pour les étapes de chargement et d'activation sont appelées après le chargement ou l'activation effective de la règle, mais avant tout traitement entrepris par cette dernière. Les actions définies pour les étapes de désactivation et de déchargement sont appelées immédiatement avant la désactivation ou le déchargement effectif de la règle.

Attributs

<lifeCycleActions> ne possède aucun attribut.

Contenu dans

<lifeCycleActions> est contenu dans les éléments suivants :

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

Contient

<lifeCycleActions> contient les éléments suivants.

Ces éléments doivent être codés dans l'ordre indiqué. Il n'est pas nécessaire de coder les éléments facultatifs, mais tous les éléments codés doivent être placés dans le bon ordre.

Tableau 45. Eléments contenus dans <lifeCycleActions>

Elément	Obligatoire ou facultatif ?
<onLoad>	Facultatif. Le nombre d'occurrences autorisé est de 0 ou 1.
<onActivation>	Facultatif. Le nombre d'occurrences autorisé est de 0 ou 1.
<onDeactivation>	Facultatif. Le nombre d'occurrences autorisé est de 0 ou 1.
<onUnload>	Facultatif. Le nombre d'occurrences autorisé est de 0 ou 1.

Elément never

L'élément <never> spécifie qu'une règle n'est jamais désactivée à une certaine heure. Elle peut tout de même être désactivée par un événement ou par d'autres moyens.

Attributs

<never> ne possède aucun attribut.

Contenu dans

<never> est contenu dans l'élément suivant :

- <stop>

Contient

<never> ne contient aucun élément.

Elément onActivation

L'élément <onActivation> définit l'action ou l'ensemble d'actions à appliquer lorsque la règle est activée. L'action <onActivation> est appelée après l'activation de la règle, mais avant tout traitement entrepris par cette dernière.

Détails

Si le jeu de règles contient plusieurs règles activées à la même date et à la même heure ou par le même événement, et possédant la même plage temporelle, les actions suivantes pour ces règles ne s'exécutent pas précisément au même moment :

- Actions de réponse à la règle dans les éléments <onTimeOut> et <onTimeWindowComplete>
- Actions de cycle de vie dans les éléments <onActivation> et <onDeactivation>

Ces actions s'exécutent de manière séquentielle dans n'importe quel ordre. Elles ne s'exécutent pas nécessairement dans l'ordre dans lequel elles sont codées dans le jeu de règles. Du fait que chaque action doit être effectuée avant que l'action suivante dans la séquence ne commence, les actions ne s'exécutent pas en même temps.

Attributs

<onActivation> ne possède aucun attribut.

Contenu dans

<onActivation> est contenu dans l'élément suivant :

- <lifeCycleActions>

Contient

<onActivation> contient l'élément suivant :

Tableau 46. Eléments contenus dans <onActivation>

Elément	Obligatoire ou facultatif ?
<action>	Facultatif. Le nombre d'occurrences autorisé est de 0 ou plus.

Elément onDeactivation

L'élément <onDeactivation> définit l'action ou l'ensemble d'actions à appliquer lorsque la règle est désactivée. L'action <onDeactivation> est appelée immédiatement avant la désactivation de la règle.

Détails

Si le jeu de règles contient plusieurs règles activées à la même date et à la même heure ou par le même événement, et possédant la même plage temporelle, les actions suivantes pour ces règles ne s'exécutent pas précisément au même moment :

- Actions de réponse à la règle dans les éléments <onTimeOut> et <onTimeWindowComplete>
- Actions de cycle de vie dans les éléments <onActivation> et <onDeactivation>

Ces actions s'exécutent de manière séquentielle dans n'importe quel ordre. Elles ne s'exécutent pas nécessairement dans l'ordre dans lequel elles sont codées dans le jeu de règles. Du fait que chaque action doit être effectuée avant que l'action suivante dans la séquence ne commence, les actions ne s'exécutent pas en même temps.

Attributs

<onDeactivation> ne possède aucun attribut.

Contenu dans

<onDeactivation> est contenu dans l'élément suivant :

- <lifeCycleActions>

Contient

<onDeactivation> contient l'élément suivant :

Tableau 47. Eléments contenus dans <onDeactivation>

Elément	Obligatoire ou facultatif ?
<action>	Facultatif. Le nombre d'occurrences autorisé est de 0 ou plus.

Elément onDetection

L'élément <onDetection> est valide uniquement pour les règles de duplication, filtrage, séquence et seuil. Cet élément définit l'action ou l'ensemble d'actions à appliquer lorsque le modèle de règle est détecté.

Détails

Le tableau tableau 48explique comment le modèle de règle est détecté pour chaque type de règle où une action <onDetection> est valide.

Tableau 48. Détection d'un modèle de règle en fonction du type de règle

Type de règle	Détection du modèle de règle
duplication	Ce modèle de règle est détecté à la réception du premier événement qui répond aux critères de sélection.
filtrage	Ce modèle de règle est détecté à la réception de tout événement répondant aux critères de sélection.
séquence	Ce modèle de règle est détecté à la réception d'une séquence d'événements répondant aux critères, dans le bon ordre et au cours de la plage temporelle.
seuil	Ce modèle de règle est détecté à la réception d'événements répondant aux critères, au cours de la plage temporelle et lorsque le seuil est atteint.

Attributs

<onDetection> ne possède aucun attribut.

Contenu dans

<onDetection> est contenu dans les éléments suivants :

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

Contient

<onDetection> contient l'élément suivant :

Tableau 49. Eléments contenus dans <onDetection>

Elément	Obligatoire ou facultatif ?
<action>	Facultatif. Le nombre d'occurrences autorisé est de 0 ou plus.

Elément onLoad

L'élément <onLoad> définit l'action ou l'ensemble d'actions à appliquer lorsque la règle est chargée (ou déployée) au cours de l'exécution du moteur Active Correlation Technology. L'action <onLoad> est appelée après le chargement de la règle, mais avant que la règle ne commence tout traitement.

Attributs

<onLoad> ne possède aucun attribut.

Contenu dans

<onLoad> est contenu dans l'élément suivant :

- <lifeCycleActions>

Contient

<onLoad> contient cet élément :

Tableau 50. Eléments contenus dans <onLoad>

Elément	Obligatoire ou facultatif ?
<action>	Facultatif. Le nombre d'occurrences autorisé est de 0 ou plus.

Elément onNextEvent

L'élément <onNextEvent> est valide uniquement pour la règle de duplication. Cet élément définit l'action ou l'ensemble d'actions à appliquer lorsque la règle de duplication reçoit le deuxième événement et tous les événements suivants qui répondent aux critères de sélection au cours de la plage temporelle.

Détails

Pour les règles de duplication, le moteur Active Correlation Technology omet le traitement par le jeu de règles du deuxième événement et de tous les événements suivants qui répondent aux critères de sélection au cours de la plage temporelle définie. Par conséquent, le codage d'une action <onNextEvent> a pour seul objectif de définir un traitement alternatif pour le deuxième événement et les suivants.

Attributs

<onNextEvent> ne possède aucun attribut.

Contenu dans

<onNextEvent> est contenu dans l'élément suivant :

- <duplicateRule>

Contient

<onNextEvent> contient l'élément suivant :

Tableau 51. Eléments contenus dans <onNextEvent>

Elément	Obligatoire ou facultatif ?
<action>	Facultatif. Le nombre d'occurrences autorisé est de 0 ou plus.

Elément onTimeOut

L'élément <onTimeOut> est valide uniquement pour les règles de séquence et de seuil. Cet élément définit l'action ou l'ensemble d'actions à appliquer si la plage temporelle définie pour la règle expire.

Détails

Le tableau tableau 52 explique comment la plage temporelle expire pour chaque type de règle où une action <onTimeOut> est valide.

Tableau 52. Expiration de la plage temporelle en fonction du type de règle

Type de règle	Expiration de la plage temporelle
séquence	La plage temporelle expire si un ou plusieurs événements sont acceptés, mais que toute la séquence des événements n'est pas reçue au cours de la plage temporelle.
seuil	La plage temporelle expire si un ou plusieurs événements sont acceptés, mais que le seuil n'est pas atteint au cours de la plage temporelle.

Si le jeu de règles contient plusieurs règles activées à la même date et à la même heure ou par le même événement, et possédant la même plage temporelle, les actions suivantes pour ces règles ne s'exécutent pas précisément au même moment :

- Actions de réponse à la règle dans les éléments <onTimeOut> et <onTimeWindowComplete>
- Actions de cycle de vie dans les éléments <onActivation> et <onDeactivation>

Ces actions s'exécutent de manière séquentielle dans n'importe quel ordre. Elles ne s'exécutent pas nécessairement dans l'ordre dans lequel elles sont codées dans le jeu de règles. Du fait que chaque action doit être effectuée avant que l'action suivante dans la séquence ne commence, les actions ne s'exécutent pas en même temps.

Attributs

<onTimeOut> ne possède aucun attribut.

Contenu dans

<onTimeOut> est contenu dans les éléments suivants :

- <sequenceRule>
- <thresholdRule>

Contient

<onTimeOut> contient l'élément suivant :

Tableau 53. Eléments contenus dans <onTimeOut>

Elément	Obligatoire ou facultatif ?
<action>	Facultatif. Le nombre d'occurrences autorisé est de 0 ou plus.

Elément onTimeWindowComplete

L'élément <onTimeWindowComplete> est valide uniquement pour les règles de collecte, de calcul, de duplication, ainsi que pour la règle temporisée. Cet élément définit l'action ou l'ensemble des actions à appliquer lorsque la plage temporelle pour la règle est terminée.

Détails

Si le jeu de règles contient plusieurs règles activées à la même date et à la même heure ou par le même événement, et possédant la même plage temporelle, les actions suivantes pour ces règles ne s'exécutent pas précisément au même moment :

- Actions de réponse à la règle dans les éléments <onTimeOut> et <onTimeWindowComplete>
- Actions de cycle de vie dans les éléments <onActivation> et <onDeactivation>

Ces actions s'exécutent de manière séquentielle dans n'importe quel ordre. Elles ne s'exécutent pas nécessairement dans l'ordre dans lequel elles sont codées dans le jeu de règles. Du fait que chaque action doit être effectuée avant que l'action suivante dans la séquence ne commence, les actions ne s'exécutent pas en même temps.

Attributs

<onTimeWindowComplete> ne possède aucun attribut.

Contenu dans

<onTimeWindowComplete> est contenu dans les éléments suivants :

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

Contient

<onTimeWindowComplete> contient l'élément suivant :

Tableau 54. Eléments contenus dans <onTimeWindowComplete>

Elément	Obligatoire ou facultatif ?
<action>	Facultatif. Le nombre d'occurrences autorisé est de 0 ou plus.

Elément onUnload

L'élément <onUnload> définit l'action ou l'ensemble des actions à appliquer lorsque la règle est déchargée ou supprimée du moteur Active Correlation Technology en cours d'exécution. L'action <onUnload> est appelée immédiatement avant le déchargement de la règle.

Attributs

<onUnload> ne possède aucun attribut.

Contenu dans

<onUnload> est contenu dans l'élément suivant :

- <lifeCycleActions>

Contient

<onUnload> contient l'élément suivant :

Tableau 55. Eléments contenus dans <onUnload>

Elément	Obligatoire ou facultatif ?
<action>	Facultatif. Le nombre d'occurrences autorisé est de 0 ou plus.

Elément ruleBlock

L'élément <ruleBlock> permet de grouper des règles liées et d'organiser des règles hiérarchiquement.

Attributs

L'élément <ruleBlock> possède l'attribut suivant :

Tableau 56. Attribut de l'élément <ruleBlock>

Nom	Description	Type de données	Obligatoire ?
name	Identifie le bloc de règles. Cet identificateur doit être unique dans le jeu de règles ou le bloc de règles qui contient ce bloc de règles. Il ne peut pas contenir de point.	xsd:NMTOKEN	Oui

Contenu dans

L'élément <ruleBlock> est contenu dans les éléments suivants :

- <ruleSet>
- <ruleBlock>

Contient

L'élément <ruleBlock> contient les éléments suivants.

S'ils sont codés, les éléments <comment>, <import> et <variable> doivent l'être dans l'ordre indiqué. Les éléments restants peuvent être codés dans n'importe quel

ordre.

Tableau 57. Eléments contenus dans l'élément <ruleBlock>

Elément	Obligatoire ou facultatif ?
<comment>	Facultatif. Le nombre d'occurrences autorisé est de 0 ou 1.
<import>	Facultatif. Le nombre d'occurrences autorisé est de 0 ou plus.
<variable>	Facultatif. Le nombre d'occurrences autorisé est de 0 ou plus.
<ruleBlock>	Facultatif. Le nombre d'occurrences autorisé est de 0 ou plus.
<collectionRule>	Facultatif. Le nombre d'occurrences autorisé est de 0 ou plus.
<computationRule>	Facultatif. Le nombre d'occurrences autorisé est de 0 ou plus.
<duplicateRule>	Facultatif. Le nombre d'occurrences autorisé est de 0 ou plus.
<filterRule>	Facultatif. Le nombre d'occurrences autorisé est de 0 ou plus.
<sequenceRule>	Facultatif. Le nombre d'occurrences autorisé est de 0 ou plus.
<thresholdRule>	Facultatif. Le nombre d'occurrences autorisé est de 0 ou plus.
<timerRule>	Facultatif. Le nombre d'occurrences autorisé est de 0 ou plus.

Elément ruleSet

L'élément <ruleSet> défini par `act:ruleSet` est l'élément principal du langage de règle Active Correlation Technology. Tous les autres éléments sont contenus dans l'élément <ruleSet>.

Détails

Les éléments <ruleSet> définis par le schéma de langage Active Correlation Technology (`act:ruleSet`) et par le schéma de jeu de règles de base Active Correlation Technology (`br:ruleSet`) sont des doublons. Cependant, lorsque vous créez un jeu de règles, vous devez définir l'espace de nom suivant dans l'élément <ruleSet> : `act:ruleSet`.

Attributs

L'élément <ruleSet> possède l'attribut suivant :

Tableau 58. Attribut de l'élément <ruleSet>

Nom	Description	Type de données	Obligatoire ?
name	Identifie le jeu de règles. Cet identificateur doit être unique. Il ne peut pas contenir de point.	xsd:NMTOKEN	Oui

Contenu dans

<ruleSet> étant l'élément principal du langage de règle, il n'est contenu dans aucun élément.

Contient

L'élément <ruleSet> contient les éléments suivants.

Ces éléments doivent être codés dans l'ordre indiqué. Il n'est pas nécessaire de coder les éléments facultatifs, mais tous les éléments codés doivent être placés dans le bon ordre.

Tableau 59. Eléments contenus dans l'élément <ruleSet>

Elément	**Obligatoire ou facultatif ?**
<comment>	Facultatif. Le nombre d'occurrences autorisé est de 0 ou 1.
<import>	Facultatif. Le nombre d'occurrences autorisé est de 0 ou plus.
<variable>	Facultatif. Le nombre d'occurrences autorisé est de 0 ou plus.
<ruleBlock>	Facultatif. Le nombre d'occurrences autorisé est de 0 ou plus.

Elément runUntilDeactivated

L'élément <runUntilDeactivated> spécifie que la plage temporelle reste ouverte jusqu'à la désactivation de la règle. Le début de la plage temporelle de cette règle coïncide donc avec le lancement par la règle du processus de traitement, et sa fin correspond à la désactivation de la règle ou sa suppression du jeu de règles, ou encore à l'arrêt du moteur Active Correlation Technology.

Détails

Le comportement spécifique d'une règle comportant l'élément <runUntilDeactivated> dépend du type de la règle. tableau 60 décrit le comportement de règle pour chaque type de règle dans lequel l'élément <timeWindow> est valide et contient l'élément <runUntilDeactivated>.

Tableau 60. Comportement de règle lorsque l'élément <runUntilDeactivated> est codé

Type de règle	**Comportement de règle lorsque l'élément <runUntilDeactivated> est codé**
collecte	La règle de collecte accepte le premier événement qui satisfait ses critères de sélection d'événement et continue d'accepter et de traiter des événements jusqu'à sa désactivation ; les actions définies dans l'élément <onTimeWindowComplete> sont alors exécutées, immédiatement suivies par les actions définies dans l'élément <onDeactivation>.
calcul	La règle de calcul accepte le premier événement qui satisfait ses critères de sélection d'événement et continue d'accepter et de traiter des événements jusqu'à sa désactivation ; les actions définies dans l'élément <onTimeWindowComplete> sont alors exécutées, immédiatement suivies par les actions définies dans l'élément <onDeactivation>.
duplication	La règle de duplication accepte le premier événement qui satisfait ses critères de sélection d'événement et continue d'accepter et de traiter des événements jusqu'à sa désactivation ; les actions définies dans l'élément <onTimeWindowComplete> sont alors exécutées, immédiatement suivies par les actions définies dans l'élément <onDeactivation>.

Tableau 60. Comportement de règle lorsque l'élément <runUntilDeactivated> est codé (suite)

Type de règle	Comportement de règle lorsque l'élément <runUntilDeactivated> est codé
séquence	La règle de séquence accepte le premier événement qui satisfait ses critères de sélection d'événement et continue d'accepter et de traiter des événements jusqu'à ce que l'une de ces situations se produise : • Le modèle de séquence est détecté. Dans ce cas, les actions définies dans l'élément <onDetection> sont exécutées et la règle revient à son état initial. La règle reprend le traitement d'événement et ce processus peut se répéter de nombreuses fois jusqu'à la désactivation de la règle. • La règle est désactivée au cours du traitement d'événement. Dans ce cas, les actions définies dans l'élément <onTimeOut> sont exécutées, immédiatement suivies par les actions définies dans l'élément <onDeactivation>.
seuil	La règle de seuil accepte le premier événement qui satisfait ses critères de sélection d'événement et continue d'accepter et de traiter des événements jusqu'à ce que l'une de ces situations se produise : • Le modèle de seuil est détecté. Dans ce cas, les actions définies dans l'élément <onDetection> sont exécutées et la règle revient à son état initial. La règle reprend le traitement d'événement et ce processus peut se répéter de nombreuses fois jusqu'à la désactivation de la règle. • La règle est désactivée au cours du traitement d'événement. Dans ce cas, les actions définies dans l'élément <onTimeOut> sont exécutées, immédiatement suivies par les actions définies dans l'élément <onDeactivation>.
temporisateur	Une fois active, la règle temporisée ne fait rien jusqu'à sa désactivation ; les actions définies dans l'élément <onTimeWindowComplete> sont alors exécutées, immédiatement suivies par les actions définies dans l'élément <onDeactivation>. L'attribut de répétition de l'élément <timerRule> est ignoré.

Attributs

L'élément <runUntilDeactivated> ne possède aucun attribut.

Contenu dans

L'élément <runUntilDeactivated> est contenu dans l'élément suivant :

- <timeWindow>

Contient

L'élément <runUntilDeactivated> ne contient aucun élément.

Elément sequenceRule

L'élément <sequenceRule> définit une règle selon le modèle de séquence. La règle de séquence est la seule règle permettant d'avoir plusieurs sélecteurs d'événement. Elle requiert également un minimum de deux sélecteurs d'événement.

Attributs

L'élément <sequenceRule> possède les attributs suivants :

Tableau 61. Attributs de l'élément <sequenceRule>

Nom	Description	Type de données	Obligatoire ?
name	Identifie la règle. Cet identificateur doit être unique dans le bloc de règles qui contient la règle. Il ne doit pas contenir de point.	xsd:NMTOKEN	Oui
processOnlyForwardedEvents	Spécifie si la règle reçoit tous les événements ou seulement ceux qui sont acheminés depuis d'autres règles. La valeur par défaut est `false`, ce qui indique que la règle reçoit tous les événements, y compris ceux qui sont acheminés depuis d'autres règles.	xsd:boolean	Non
arrivalOrder	Indique si les événements doivent arriver dans l'ordre de codage des éléments <eventSelector> de la règle. Valeurs correctes : • `inOrder` (valeur par défaut) • `randomOrder`	xsd:string	Non

Si la valeur de l'attribut arrivalOrder est `randomOrder`, l'ordre dans lequel les éléments <eventSelector> sont codés est important. Les éléments <eventSelector> ayant les critères de sélection d'événement les plus précis doivent être codés avant les éléments <eventSelector> ayant des critères moins précis. Dans le cas contraire, la séquence attendue n'est pas détectée.

Supposons, par exemple, l'une des circonstances suivantes :

- Trois éléments <eventSelector> sont définis.
- Le premier élément <eventSelector> recherche l'événement `eventA`.
- Le deuxième élément <eventSelector> recherche tous les événements.
- Le troisième élément <eventSelector> recherche l'événement `eventB`.
- Les événements suivants sont présentés au système au cours de la plage temporelle définie : `eventA`, `eventB` et `eventC`.

Le comportement de la règle est le suivant, et la séquence attendue n'est pas détectée :

1. Le premier événement, `eventA`, est accepté par le premier élément <eventSelector>.
2. Le deuxième événement, `eventB`, est accepté par le deuxième élément <eventSelector>.
3. Le troisième événement, `eventC`, est ignoré.

Supposons les circonstances suivantes, où les éléments <eventSelector> sont codés de façon correcte, les critères de sélection d'événement les plus spécifiques précédant les moins spécifiques :

- Le premier élément <eventSelector> recherche l'événement `eventA`.
- Le deuxième élément <eventSelector> recherche l'événement `eventB`.
- Le troisième élément <eventSelector> recherche tous les événements.

Le comportement de la règle est le suivant, et la séquence est détectée :

1. Le premier événement, `eventA`, est accepté par le premier élément <eventSelector>.
2. Le deuxième événement, `eventB`, est accepté par le deuxième élément <eventSelector>.
3. Le troisième événement, `eventC`, est accepté par le troisième élément <eventSelector>.

Contenu dans

L'élément <sequenceRule> est contenu dans l'élément suivant :

- <ruleBlock>

Contient

L'élément <sequenceRule> contient les éléments suivants.

Ces éléments doivent être codés dans l'ordre indiqué. Il n'est pas nécessaire de coder les éléments facultatifs, mais tous les éléments codés doivent être placés dans le bon ordre.

Tableau 62. Eléments contenus dans l'élément <sequenceRule>

Elément	Obligatoire ou facultatif ?
<comment>	Facultatif. Le nombre d'occurrences autorisé est de 0 ou 1.
<variable>	Facultatif. Le nombre d'occurrences autorisé est de 0 ou plus.
<activationInterval>	Facultatif. Le nombre d'occurrences autorisé est de 0 ou 1.
<lifeCycleActions>	Facultatif. Le nombre d'occurrences autorisé est de 0 ou 1.
<eventSelector>	Deux occurrences de cet élément sont obligatoires pour la séquence. Des occurrences supplémentaires sont autorisées.
<groupingKey>	Facultatif. Le nombre d'occurrences autorisé est de 0 ou 1.
<timeWindow>	Obligatoire. Une seule occurrence autorisée.
<oonDetection>	Facultatif. Le nombre d'occurrences autorisé est de 0 ou 1.
<onTimeOut>	Facultatif. Le nombre d'occurrences autorisé est de 0 ou 1.

Concepts associés

«Modèle de séquence», à la page 14
Une règle de séquence est définie par le modèle de séquence. Elle détecte l'arrivée d'une séquence donnée d'événements au cours d'un intervalle de temps. La séquence peut être ordonnée ou aléatoire. Il s'agit d'une règle avec état.

Elément start

L'élément <start> indique si une règle est activée à une heure et une date données ou si elle est activée lorsqu'elle est chargée par le moteur Active Correlation Technology.

Détails

Si l'élément <start> n'est pas codé du tout, l'heure de début par défaut correspond à celle qui est définie par l'élément <whenLoaded>.

Attributs

L'élément <start> ne possède aucun attribut.

Contenu dans

L'élément <start> est contenu dans l'élément suivant :

- <activationTime>

Contient

L'élément <start> contient les éléments suivants :

Tableau 63. Eléments contenus dans l'élément <start>

Elément	Obligatoire ou facultatif ?
<dateTime>	L'un des ces éléments est obligatoire et une seule occurrence de l'élément choisi est autorisée.
<whenLoaded>	
<inactiveWhenLoaded>	

Elément stop

L'élément <stop> indique si une règle est désactivée à une heure et une date données, au terme d'une période donnée, ou si elle n'est jamais désactivée à une heure donnée.

Détails

Si l'élément <stop> n'est pas codé du tout, l'heure de fin par défaut correspond à celle qui est définie par l'élément <never>.

Attributs

L'élément <stop> ne possède aucun attribut.

Contenu dans

L'élément <stop> est contenu dans l'élément suivant :

- <activationTime>

Contient

L'élément <stop> contient les éléments suivants :

Tableau 64. Eléments contenus dans l'élément <stop>

Elément	Obligatoire ou facultatif ?
<dateTime>	L'un des ces éléments est obligatoire et une seule occurrence de l'élément choisi est autorisée.
<never>	
<after>	

Elément stopAfter

L'élément <stopAfter> spécifie la durée pendant laquelle une instance de règle, telle que définie par l'élément <groupingKey>, doit rester active après son activation. Au terme de cette durée, l'instance de règle doit être désactivée.

Attributs

L'élément <stopAfter> possède les attributs suivants :

Tableau 65. Attributs de l'élément <stopAfter>

Nom	Description	Type de données	Obligatoire ?
duration	Spécifie la quantité de temps pour la durée. Le type de données de cet attribut dépend de la valeur de l'attribut unit.	• Si la valeur de l'attribut unit est `ISO-8601`, ce type de données est xsd:duration. • Si la valeur de l'attribut unit est `milliseconds`, ce type de données est xsd:positiveInteger.	Oui
unit	Spécifie l'unité de temps à utiliser. Les valeurs correctes pour cet attribut sont : • `ISO-8601` • `milliseconds`	xsd:string	Oui

Utilisation de la norme ISO 8601 pour la durée

Lorsque le code `ISO-8601` est utilisé comme valeur de l'attribut unit (unité), la valeur de l'attribut duration (durée) utilise la norme ISO 8601 pour spécifier la durée sous forme de chaîne unique. La spécification du type de données selon le schéma XML standard utilise la norme ISO 8601 pour fournir un type de données appelé `duration`. Ce type de données est décrit plus en détail à l'adresse http://www.w3.org/TR/xmlschema-2/#duration.

La chaîne suivante correspond au format du type de données `duration` utilisé dans le schéma XML standard :

`PnYnMnDTnHnMnS`

- `P` est le caractère qui apparaît toujours en début de chaîne.

- *n*Y représente le nombre d'années. Une année équivaut à 365 jours. Ainsi, coder 1Y revient à coder 365D.
- *n*M représente le nombre de mois. Un mois équivaut à 30 jours. Ainsi, coder 1M revient à coder 30D.
- *n*D représente le nombre de jours.
- T est un séparateur placé entre les unités de date (années, mois et jour) et les unités d'heure (heures, minutes et secondes). Les unités d'heure suivent toujours le caractère T.
- *n*H représente le nombre d'heures.
- *n*M représente le nombre de mois.
- *n*S représente le nombre de secondes.

Exemples de ce format :

- P5DT12H indique 5 jours et demi.
- PT59M59S indique 59 minutes et 59 secondes.
- P1M indique un mois.

Contenu dans

L'élément <stopAfter> est contenu dans l'élément <activateOnEvent>, mais uniquement lorsque l'élément <activateOnEvent> est codé dans l'élément <activationByGroupingKey>.

Contient

L'élément <stopAfter> ne contient aucun élément.

Elément thresholdRule

L'élément <thresholdRule> définit une règle selon le modèle de seuil.

Attributs

L'élément <thresholdRule> possède les attributs suivants :

Tableau 66. Attributs de l'élément <thresholdRule>

Nom	Description	Type de données	Obligatoire ?
name	Identifie la règle. Cet identificateur doit être unique dans le bloc de règles qui contient la règle. Il ne doit pas contenir de point.	xsd:NMTOKEN	Oui
processOnlyForwardedEvents	Spécifie si la règle reçoit tous les événements ou seulement ceux qui sont acheminés depuis d'autres règles. La valeur par défaut est false, ce qui indique que la règle reçoit tous les événements, y compris ceux qui sont acheminés depuis d'autres règles.	xsd:boolean	Non

Contenu dans

L'élément <thresholdRule> est contenu dans l'élément suivant :

- <ruleBlock>

Contient

L'élément <thresholdRule> contient les éléments suivants.

Ces éléments doivent être codés dans l'ordre indiqué. Il n'est pas nécessaire de coder les éléments facultatifs, mais tous les éléments codés doivent être placés dans le bon ordre.

Tableau 67. Eléments contenus dans l'élément <thresholdRule>

Elément	Obligatoire ou facultatif ?
<comment>	Facultatif. Le nombre d'occurrences autorisé est de 0 ou 1.
<variable>	Facultatif. Le nombre d'occurrences autorisé est de 0 ou plus.
<activationInterval>	Facultatif. Le nombre d'occurrences autorisé est de 0 ou 1.
<lifeCycleActions>	Facultatif. Le nombre d'occurrences autorisé est de 0 ou 1.
<eventSelector>	Facultatif. Le nombre d'occurrences autorisé est de 0 ou 1.
<groupingKey>	Facultatif. Le nombre d'occurrences autorisé est de 0 ou 1.
<booleanThreshold>	L'un des ces éléments est obligatoire et une seule occurrence de l'élément choisi est autorisée.
<computedThreshold>	
<eventCountThreshold>	
<timeWindow>	Obligatoire. Une seule occurrence autorisée.
<onDetection>	Facultatif. Le nombre d'occurrences autorisé est de 0 ou 1.
<onTimeOut>	Facultatif. Le nombre d'occurrences autorisé est de 0 ou 1.

Concepts associés

«Modèle de seuil», à la page 17
Une règle de seuil est définie par le modèle de seuil. Elle collecte un groupe d'événements sélectionnés au cours d'un intervalle de temps et détermine, une fois que chaque événement a été reçu, si une condition de seuil est remplie. Il s'agit d'une règle avec état.

Elément timeInterval

L'élément <timeInterval> définit la durée de la plage temporelle.

Attributs

L'élément <timeInterval> possède les attributs suivants :

Tableau 68. Attributs de l'élément <timeInterval>

Nom	Description	Type de données	Obligatoire ?
duration	Spécifie la quantité de temps pour la durée. Le type de données de cet attribut dépend de la valeur de l'attribut unit.	• Si la valeur de l'attribut unit est `ISO-8601`, ce type de données est xsd:duration. • Si la valeur de l'attribut unit est `milliseconds`, ce type de données est xsd:positiveInteger.	Oui
unit	Spécifie l'unité de temps à utiliser. Les valeurs correctes pour cet attribut sont : • `ISO-8601` • `milliseconds`	xsd:string	Oui

Utilisation de la norme ISO 8601 pour la durée

Lorsque le code `ISO-8601` est utilisé comme valeur de l'attribut unit (unité), la valeur de l'attribut duration (durée) utilise la norme ISO 8601 pour spécifier la durée sous forme de chaîne unique. La spécification du type de données selon le schéma XML standard utilise la norme ISO 8601 pour fournir un type de données appelé `duration`. Ce type de données est décrit plus en détail à l'adresse http://www.w3.org/TR/xmlschema-2/#duration.

La chaîne suivante correspond au format du type de données `duration` utilisé dans le schéma XML standard :

`PnYnMnDTnHnMnS`

- `P` est le caractère qui apparaît toujours en début de chaîne.
- *n*`Y` représente le nombre d'années. Une année équivaut à 365 jours. Ainsi, coder `1Y` revient à coder `365D`.
- *n*`M` représente le nombre de mois. Un mois équivaut à 30 jours. Ainsi, coder `1M` revient à coder `30D`.
- *n*`D` représente le nombre de jours.
- `T` est un séparateur placé entre les unités de date (années, mois et jour) et les unités d'heure (heures, minutes et secondes). Les unités d'heure suivent toujours le caractère `T`.
- *n*`H` représente le nombre d'heures.
- *n*`M` représente le nombre de mois.
- *n*`S` représente le nombre de secondes.

Exemples de ce format :

- `P5DT12H` indique 5 jours et demi.
- `PT59M59S` indique 59 minutes et 59 secondes.
- `P1M` indique un mois.

Contenu dans

L'élément <timeInterval> est contenu dans l'élément suivant :

- <timeWindow>

Contient

L'élément <timeInterval> ne contient aucun élément.

Elément timerRule

L'élément <timerRule> définit une règle selon le modèle de temporisateur.

Attributs

L'élément <timerRule> possède les attributs suivants :

Tableau 69. Attributs de l'élément <timerRule>

Nom	Description	Type de données	Obligatoire ?
name	Identifie la règle. Cet identificateur doit être unique dans le bloc de règles qui contient la règle. Il ne doit pas contenir de point.	xsd:NMTOKEN	Oui
processOnlyForwardedEvents	Cet attribut est ignoré car la règle temporisée ne traite pas les événements.	xsd:boolean	Non
repeat	Définit si la règle temporisée s'exécute en mode répétition jusqu'à sa désactivation. Valeurs correctes : • `true` (vrai - valeur par défaut) • `false` (faux) Si la valeur est définie sur `false`, la règle ne s'exécute qu'une fois au cours de son intervalle de temps, exécute l'action de réponse à la règle lorsque la plage temporelle correspondante est terminée, puis s'arrête. Si l'élément <timeWindow> pour la règle temporisée contient l'élément <runUntilDeactivated>, l'attribut de répétition est ignoré.	xsd:boolean	Non

Contenu dans

L'élément <timerRule> est contenu dans l'élément suivant :

- <ruleBlock>

Contient

L'élément <timerRule> contient les éléments suivants.

Ces éléments doivent être codés dans l'ordre indiqué. Il n'est pas nécessaire de coder les éléments facultatifs, mais tous les éléments codés doivent être placés dans le bon ordre.

Tableau 70. Eléments contenus dans l'élément <timerRule>

Elément	Obligatoire ou facultatif ?
<comment>	Facultatif. Le nombre d'occurrences autorisé est de 0 ou 1.
<variable>	Facultatif. Le nombre d'occurrences autorisé est de 0 ou plus.
<activationInterval>	Facultatif. Le nombre d'occurrences autorisé est de 0 ou 1.
<lifeCycleActions>	Facultatif. Le nombre d'occurrences autorisé est de 0 ou 1.
<timeWindow>	Obligatoire. Une seule occurrence autorisée.
<onTimeWindowComplete>	Facultatif. Le nombre d'occurrences autorisé est de 0 ou 1.

Concepts associés

«Modèle de temporisateur», à la page 19
Une règle temporisée est définie par le modèle de temporisateur. Elle déclenche des actions à intervalles réguliers. Il s'agit d'une règle avec état. Bien que les règles temporisées ne traitent pas les événements, elles peuvent être activées ou désactivées par un événement.

Elément timeWindow

L'élément <timeWindow> contient des éléments qui définissent l'intervalle de temps au cours duquel la règle procède au traitement.

Détails

Par exemple, la plage temporelle d'une règle en double définit l'intervalle de temps pendant lequel la règle doit rechercher des événements correspondant à des doublons du premier événement accepté. Si la plage temporelle est de 30 secondes, la règle relative aux doublons traite tous les événements en double qui se produisent dans un délai de 30 secondes à partir du premier événement qu'elle accepte.

Attributs

L'élément <timeWindow> ne possède aucun attribut.

Contenu dans

L'élément <timeWindow> est contenu dans les éléments suivants :

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

Contient

L'élément <timeWindow> contient les éléments suivants :

Tableau 71. Eléments contenus dans l'élément <timeWindow>

Elément	Obligatoire ou facultatif ?
<timeInterval>	L'un des ces éléments est obligatoire et une seule occurrence de l'élément choisi est autorisée.
<runUntilDeactivated>	

Elément variable

L'élément <variable> définit une variable et contient des informations sous une forme qui peut être référencée par expressions. Une variable peut être définie au niveau du jeu de règles, d'un bloc de règles ou d'une règle.

Détails

variable de jeu de règles
S'applique de façon globale au jeu de règles et peut être référencée par toute expression présente dans ce jeu de règles.

variable de bloc de règles
S'applique uniquement à l'intérieur du bloc de règles (et à l'intérieur de tout bloc de règles contenu) et peut être référencée par toute expression présente dans ce bloc de règles.

variable de règle
S'applique uniquement aux expressions présentes dans cette règle.

Plusieurs variables peuvent avoir le même nom si elles se trouvent à différents niveaux de la hiérarchie de règles. Lorsqu'on accède à une variable, la définition la plus locale de la variable est utilisée. Par exemple, si une variable de même nom est définie au niveau du jeu de règles, au niveau d'un bloc de règles et au niveau d'une règle, c'est la définition de la variable au niveau de la règle qui est utilisée par les expressions présentes dans cette règle.

Lorsque des variables sont définies au niveau du jeu de règles ou au niveau d'un bloc de règles, plusieurs règles extraient et définissent ces variables à différents moments. Pour assurer une gestion correcte des valeurs de variable, apportez donc un soin particulier à la façon dont vous codez les interactions entre variables dans le jeu de règles.

Si la variable est définie au niveau du jeu de règles ou au niveau d'un bloc de règles, elle n'est pas réinitialisée après que le modèle de règle a trouvé une correspondance.

Dans les deux cas évoqués ci-après, utilisez un verrouillage sur l'extraction et la définition de variables de jeu de règles et de bloc de règles pour éviter que les valeurs de variable soient définies de façon incorrecte :

- Si la règle temporisée extrait ou définit une variable au cours d'une action <onTimeOut>
- Si l'application dans laquelle le moteur Active Correlation Technology est imbriqué est de type multiprocessus

Si une règle est définie avec une clé de groupement, des variables de règle définies par l'élément <variable> ne sont pas valables dans des actions de cycle de vie ou dans un élément <filteringPredicate> contenu dans un élément <activateOnEvent>, ou encore dans un élément <deactivateOnEvent> contenu dans un élément <activationInterval>. En effet, dans ce cas, les variables de règle s'appliquent uniquement à une instance de règle et il n'existe pas d'instance de règle au moment où ces expressions sont exécutées.

Attributs

<variable> a les attributs suivants :

Tableau 72. Attributs de l'élément <variable>

Nom	Description	Type de données	Obligatoire ?
name	Identifie une variable spécifique. Une variable est référencée par son nom.	xsd:NMTOKEN	Oui
dataType	Identifie le type d'informations contenues dans une variable. Il doit s'agir d'un type de données complet tel que java.lang.String.	xsd:NMTOKEN	Oui

Règles de dénomination des variables

L'attribution des noms de variable doit obéir à certaines règles. Ainsi, la valeur de l'attribut de nom de l'élément <variable> doit respecter les règles suivantes :

- Seuls les caractères suivants sont autorisés :
  - Lettres majuscules A à Z de l'alphabet latin, format ASCII. Représentation Unicode : `\u0041-\u005a`.
  - Lettres minuscules a à z de l'alphabet latin, format ASCII. Représentation Unicode : `\u0061-\u007a`.
  - Trait de soulignement ASCII (_). Représentation Unicode : `\u005f`.
  - Symbole du dollar ($). Représentation Unicode : `\u0024`.
  - Chiffres 0 à 9, format ASCII. Représentation Unicode : `\u0030-\u0039`.
- La valeur ne peut pas être de type NULL.
- Elle ne peut pas être une chaîne vide.
- Elle ne peut pas comporter d'espaces.
- Elle ne peut pas comporter de point.
- Le nom ne peut pas commencer par `act_`, sous quelque format que ce soit (ni en majuscules, ni en minuscules, ni dans un mélange de majuscules et de minuscules).

Contenu dans

L'élément <variable> est contenu dans les éléments suivants :

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

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

Contient

L'élément <variable> contient les éléments suivants.

Ces éléments doivent être codés dans l'ordre indiqué. Il n'est pas nécessaire de coder les éléments facultatifs, mais tous les éléments codés doivent être placés dans le bon ordre.

Tableau 73. Eléments contenus dans l'élément <variable>

Elément	Obligatoire ou facultatif ?
<comment>	Facultatif. Le nombre d'occurrences autorisé est de 0 ou 1.
<varInitializer>	Obligatoire. Une occurrence autorisée.

Concepts associés

«Variables», à la page 25
Dans le langage de règle, certaines variables sont utilisées pour stocker des informations relatives aux événements sur différentes occurrences d'événement ou règles. Ces informations sont ensuite accessibles au moyen d'expressions dans les règles. Certains types de variables sont définis par le programme d'écriture de règle, tandis que d'autres sont fournis par la technologie ACT. Certains types sont accessibles directement dans une expression, tandis que d'autres le sont uniquement par le biais de méthodes fournies par la technologie ACT.

Elément varInitializer

L'élément <varInitializer> contient une expression qui fournit la valeur initiale de la variable définie dans l'élément <variable> associé.

Détails

La variable pouvant être de tout type, le code d'expression peut renvoyer un objet tableau ou tout autre objet complexe spécifique à l'implémentation devant être stocké par le moteur Active Correlation Technology.

Voir «Variables», à la page 25 pour obtenir plus d'informations sur les variables qui peuvent être utilisées dans les expressions. L'emploi de certaines variables dépend du contexte d'expression.

Attributs

L'élément <varInitializer> possède les attributs suivants :

Tableau 74. Attributs de l'élément <varInitializer>

Nom	Description	Type de données	Obligatoire ?
expressionLanguage	Identifie le langage de programmation dans lequel l'expression est rédigée. Le langage de programmation Java étant le seul langage d'expression pris en charge, la seule valeur correcte pour cet attribut est `java`.	xsd:NMTOKEN	Oui

Contenu dans

L'élément <varInitializer> est contenu dans l'élément suivant :

- <variable>

Contient

L'élément <varInitializer> ne contient aucun élément.

Concepts associés

«Expressions», à la page 20
Une expression est un code qui contient une logique personnalisée pouvant être ajoutée à une règle. Elle peut également accéder à un code externe au moteur Active Correlation Technology. Dans le langage de règle, les expressions sont valides uniquement dans des contextes ou éléments de langage de règle spécifiques.

Elément whenLoaded

L'élément <whenLoaded> définit qu'une règle est activée lorsqu'elle est chargée par le moteur Active Correlation Technology.

Attributs

L'élément <whenLoaded> ne possède aucun attribut.

Contenu dans

L'élément <whenLoaded> est contenu dans l'élément suivant :

- <start>

Contient

L'élément <whenLoaded> ne contient aucun élément.

Chapitre 6. Glossaire

Ce glossaire contient les termes et définitions de concepts importants dans Active Correlation Technology.

ACT Voir Active Correlation Technology.

action Expression qui s'exécute en tant que réponse à une règle, ou lors du chargement, du déchargement, de l'activation ou de la désactivation d'une règle.

action de cycle de vie
Expression qui s'exécute lors du chargement, du déchargement, de l'activation ou de la désactivation d'une règle.

action de réponse à la règle
Voir action.

Active Correlation Technology
Technologie IBM qui permet de corréler des événements au moyen de règles.

bloc de règles
Unité organisationnelle des règles de groupement par fonction dans des domaines au sein du jeu de règles. Un bloc de règles peut contenir des règles, mais également d'autres bloc de règles.

clé de groupement
Méthode qui permet de conduire une règle à créer une instance de règle (ou une copie) pour chaque groupe d'événements possédant des caractéristiques communes.

Compilateur Active Correlation Technology
Composant Active Correlation Technology qui analyse un jeu de règles et tous les codes contenus dans ce dernier afin de générer les structures de données internes requises par le moteur Active Correlation Technology.

domaine
Catégorie sur laquelle est appliqué un groupe de règles selon sa fonction. Par exemple, un domaine peut représenter une zone géographique particulière, une discipline de gestion informatique (telles que la détection de sécurité ou la corrélation d'événements de réseau) ou une organisation d'entreprise (une entreprise particulière ou un service dans une entreprise).

Environnement d'exécution Active Correlation Technology
Application dans laquelle le moteur Active Correlation Technology est intégré, avec ou sans le compilateur.

événement externe
Evénement reçu par le moteur Active Correlation Technology depuis une source qui lui est extérieure.

événement interne
Evénement créé par une règle exécutée dans le moteur Active Correlation Technology. Cet événement peut être acheminé vers d'autres règles.

expression
Code qui contient une logique personnalisée pouvant être ajoutée à une règle. Les rédacteurs de règle peuvent utiliser des expressions dans

différents buts, par exemple : initialiser des variables, définir des critères de sélection d'événement ou spécifier des actions de réponse à la règle ou de cycle de vie.

fournisseur d'événements
Tout logiciel qui génère des événements traités par Active Correlation Technology.

fragment
Extrait du code source.

Générateur de règles Active Correlation Technology
Interface graphique permettant d'écrire des règles de corrélation dans le langage de règle Active Correlation Technology.

importation
Moyen spécifique aux langages de programmation de rendre un code externe accessible à des expressions.

instance de règle
Copie d'une règle, dans le contexte de la clé de groupement.

jeu de règles
Unité d'exécution de la règle dans le langage de règle Active Correlation Technology. Le jeu de règles contient des règles, organisées dans des blocs de règles, qui doivent être exécutées par le moteur Active Correlation Technology. Chaque moteur opère sur un seul jeu de règles à un certain moment.

Langage de règle Active Correlation Technology
Langage XML destiné à l'écriture de règles de corrélation d'événement. Ces règles peuvent ensuite être déployées dans les environnements d'exécution Active Correlation Technology.

langage d'expression
Langage de programmation dans lequel une expression est rédigée.

modèle de calcul
Modèle de règle qui définit une règle visant à effectuer (au moyen d'une expression) un calcul sur des événements collectés, chaque fois qu'un événement est reçu au cours d'un intervalle donné. Une règle définie par le modèle de calcul est une règle avec état.

modèle de collecte
Modèle de règle qui collecte un groupe d'événements sélectionnés au cours d'un intervalle de temps. Une règle définie par le modèle de collecte est une règle avec état.

modèle de duplication
Modèle de règle qui définit une règle destinée à compter le deuxième événement et les événements suivants acceptés dans l'intervalle de temps défini, tout en omettant leur traitement par le jeu de règles. Une règle définie par le modèle de duplication est une règle avec état.

modèle de filtrage
Modèle de règle qui définit une règle visant à appliquer une certaine action après avoir accepté un événement. Une règle définie par le modèle de filtrage agit sur un seul événement, et est par conséquent sans état.

modèle de règle
Représentation d'une situation de corrélation d'événements (telle qu'une condition de seuil ou la détection d'un événement en double). Le langage

de règle Active Correlation Technology comprend les modèles de règle suivants : collecte, calcul, duplication, filtrage, séquence, seuil et temporisateur. Le modèle d'une règle a trouvé une correspondance lorsque la situation définie par la règle se produit. Le cas échéant, la règle conclut ses opérations de traitement en appliquant les actions de réponse à la règle appropriées. Lorsqu'une règle est active, elle peut correspondre plusieurs fois au modèle de règle.

modèle de séquence
Modèle de règle qui définit une règle visant à détecter la présence ou l'absence d'une certaine séquence d'événements au cours d'un intervalle de temps spécifié. Cette séquence peut être ordonnée ou aléatoire. Une règle définie par le modèle de séquence est une règle avec état.

modèle de seuil
Modèle de règle qui définit une règle visant à collecter un groupe d'événements sélectionnés au cours d'un intervalle de temps et détermine, une fois que chaque événement a été reçu, si une condition de seuil est remplie. Une règle définie par le modèle de seuil est une règle avec état.

Moteur Active Correlation Technology
Composant Active Correlation Technology qui traite les événements selon la sortie du compilateur Active Correlation Technology.

modèle de temporisateur
Modèle de règle qui définit une règle visant à initier des actions à intervalles réguliers. Une règle définie par le modèle de temporisateur est une règle avec état. Bien que les règles temporisées ne traitent pas les événements, elles peuvent être activées ou désactivées par un événement.

noeud Objet situé dans la hiérarchie de la règle qui peut être ajouté, supprimé ou remplacé de manière indépendante et individuelle au sein d'un jeu de règles. En particulier, les objets suivants sont des noeuds :

- Règles
- Blocs de règles
- Variables de bloc de règles
- Variables de jeu de règles

Un objet ne pouvant être manipulé de manière indépendante et individuelle au-dessous du niveau de la règle, la variable de règle n'est pas un noeud.

objet externe
Objet créé par une application pour communiquer avec des expressions.

prédicat
Voir prédicat de filtrage.

prédicat de filtrage
Expression qui définit la condition dans laquelle une règle accepte de traiter un événement. Le prédicat de filtrage fait partie du sélecteur d'événement. Il renvoie une valeur booléenne.

règle Unité de corrélation utilisée pour reconnaître des relations parmi des événements et pour exécuter les réponses à la règle appropriées. Une règle est une implémentation d'un des sept modèles de règle et est organisée, selon sa fonction, dans un bloc de règles faisant partie d'un jeu de règles. Une règle accepte de traiter un événement si ce dernier répond aux critères de sélection d'événement.

règle avec état
Règle qui conserve des informations d'état, sur les caractéristiques d'une instance de règle, afin d'agir sur un ensemble d'événements au cours d'une période donnée. Les règles définies par l'un des modèles suivants sont des règles avec état : collecte, calcul, duplication, séquence, seuil et temporisateur.

règle sans état
Règle qui ne conserve pas d'informations d'état, et qui par conséquent ne peut agir que sur un événement à la fois. Une règle définie par le modèle de filtrage est une règle sans état.

réponse
Voir réponse à la règle.

réponse à la règle
Expression qui s'exécute lorsque le moteur Active Correlation Technology reconnaît qu'une condition de règle est remplie. Une réponse à la règle comprend une ou plusieurs actions.

sélecteur d'événement
Critères de sélection d'événement. Ils déterminent les événements autorisés à être traités par la règle. Le sélecteur d'événement comprend le type d'événement et le prédicat de filtrage.

Annexe. Remarques

Le présent document peut contenir des informations ou des références concernant certains produits, logiciels ou services IBM non annoncés dans ce pays. Pour plus de détails, référez-vous aux documents d'annonce disponibles dans votre pays, ou adressez-vous à votre partenaire commercial IBM. Toute référence à un produit, logiciel ou service IBM n'implique pas que seul ce produit, logiciel ou service puisse être utilisé. Tout autre élément fonctionnellement équivalent peut être utilisé, s'il n'enfreint aucun droit d'IBM. Il est de la responsabilité de l'utilisateur d'évaluer et de vérifier lui-même les installations et applications réalisées avec des produits, logiciels ou services non expressément référencés par IBM.

IBM peut détenir des brevets ou des demandes de brevet couvrant les produits mentionnés dans le présent document. La remise de ce document ne vous donne aucun droit de licence sur ces brevets ou demandes de brevet. Si vous désirez recevoir des informations concernant l'acquisition de licences, veuillez en faire la demande par écrit à l'adresse suivante :

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

Les informations sur les licences concernant les produits utilisant un jeu de caractères double octet peuvent être obtenues par écrit à l'adresse suivante :

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

Le paragraphe suivant ne s'applique ni au Royaume-Uni, ni dans aucun pays dans lequel il serait contraire aux lois locales.

LE PRESENT DOCUMENT EST LIVRE EN L'ETAT SANS AUCUNE GARANTIE EXPLICITE OU IMPLICITE. IBM DECLINE NOTAMMENT TOUTE RESPONSABILITE RELATIVE A CES INFORMATIONS EN CAS DE CONTREFACON AINSI QU'EN CAS DE DEFAUT D'APTITUDE A L'EXECUTION D'UN TRAVAIL DONNE.

Certaines juridictions n'autorisant pas l'exclusion des garanties explicites ou implicites, il se peut la déclaration ci-dessus ne vous concerne pas.

Le présent document peut contenir des inexactitudes ou des coquilles. Il est mis à jour périodiquement. Chaque nouvelle édition inclut les mises à jour. IBM peut, à tout moment et sans préavis, modifier les produits et logiciels décrits dans ce document.

Les références à des sites Web non IBM sont fournies à titre d'information uniquement et n'impliquent en aucun cas une adhésion aux données qu'ils contiennent. Les éléments figurant sur ces sites Web ne font pas partie des éléments du présent produit IBM et l'utilisation de ces sites relève de votre seule responsabilité.

IBM pourra utiliser ou diffuser, de toute manière qu'elle jugera appropriée et sans aucune obligation de sa part, tout ou partie des informations qui lui seront fournies.

Les licenciés souhaitant obtenir des informations permettant : (i) l'échange des données entre des logiciels créés de façon indépendante et d'autres logiciels (dont celui-ci), et (ii) l'utilisation mutuelle des données ainsi échangées, doivent adresser leur demande à :

IBM Corporation
2Z4A/101
11400 Burnet Road
Austin, TX 78758 U.S.A.

Ces informations peuvent être soumises à des conditions particulières, prévoyant notamment le paiement d'une redevance.

Le logiciel sous licence décrit dans ce document et tous les éléments sous licence disponibles s'y rapportant sont fournis par IBM conformément aux dispositions de l'ICA, des Conditions internationales d'utilisation des logiciels IBM ou de tout autre accord équivalent.

Marques

DB2, IBM, le logo IBM logo, Tivoli, le logo Tivoli logo, Tivoli Enterprise Console, et WebSphere sont des marques ou des marques déposées d'International Business Machines Corporation aux Etats-Unis et/ou dans certains autres pays.

Java ainsi que toutes les marques et tous les logos incluant Java sont des marques ou des marques déposées de Sun Microsystems, Inc. aux Etats-Unis et/ou dans certains autres pays.

Les autres noms de sociétés, de produits et de services peuvent appartenir à des tiers.

Imprimé en France