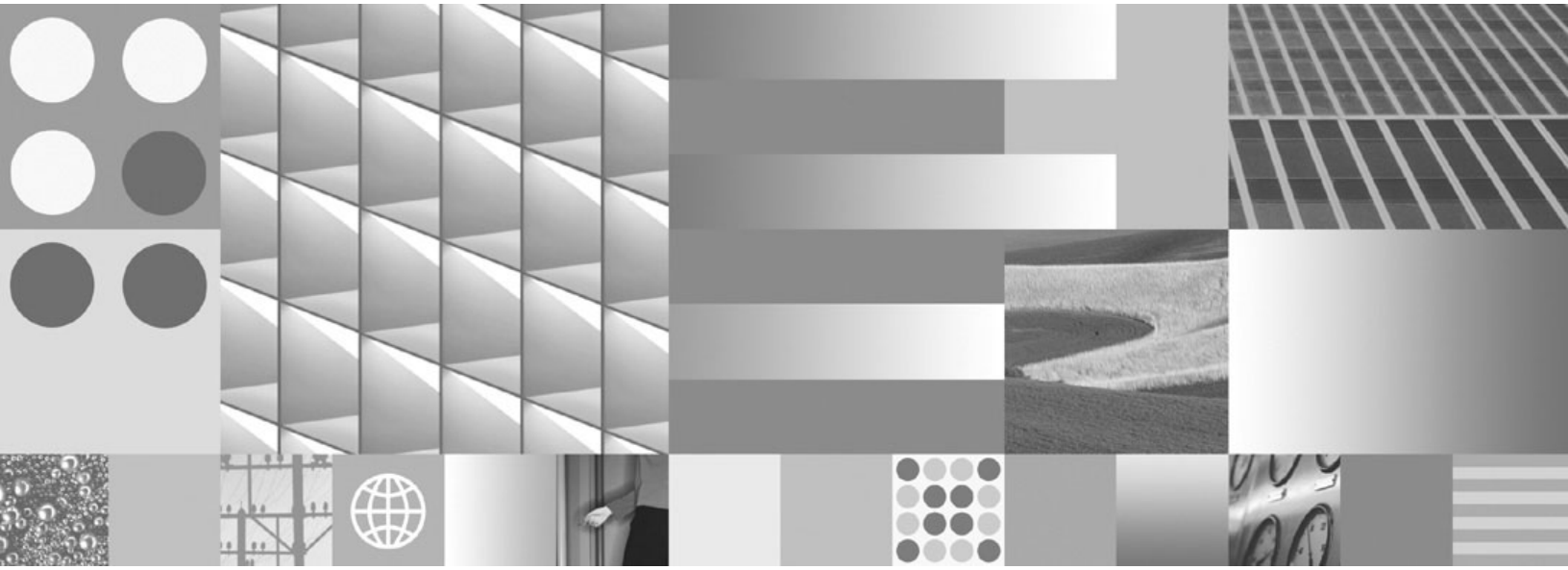
主動式相互關聯技術

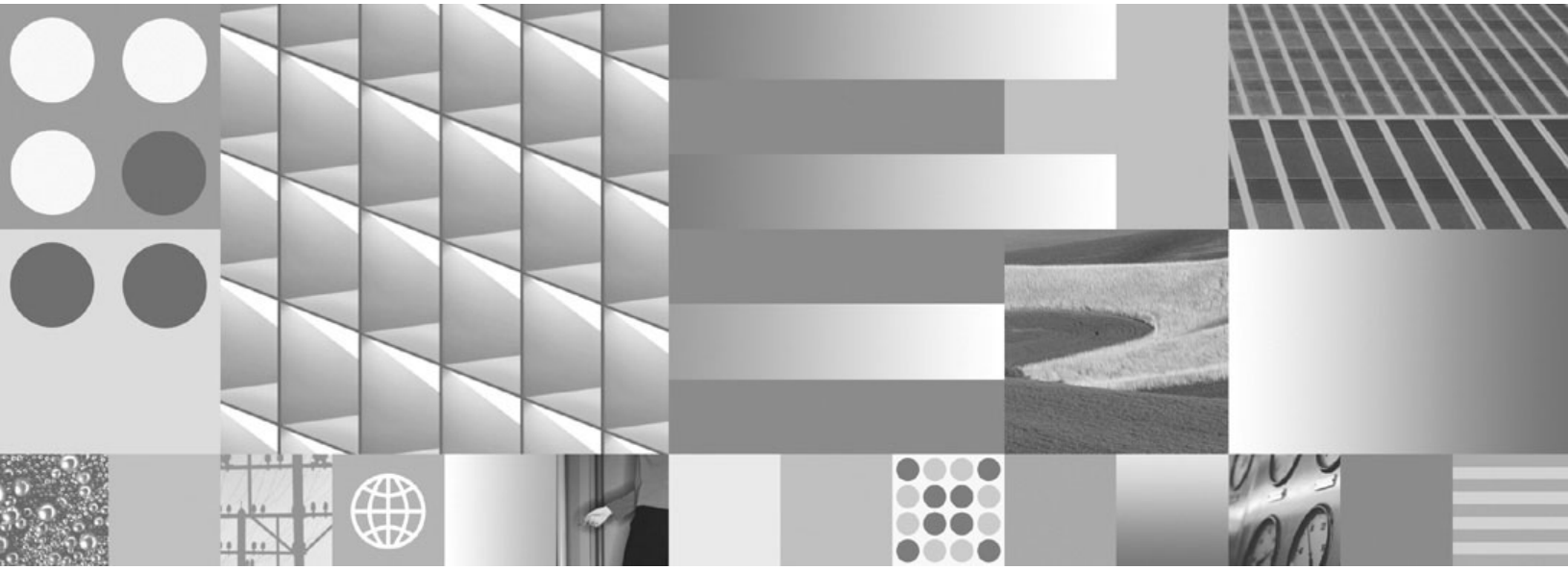
6.0 版



規則撰寫者手冊及參照

主動式相互關聯技術

6.0 版



規則撰寫者手冊及參照

注意

使用本資訊及其所支援的產品之前，請先詳閱第 111 頁的『注意事項』中的資訊。

第一版 (2006 年 6 月)

關於此資訊

此資訊提供複式事件處理程序中 IBM® Active Correlation Technology 及其角色的概觀。「主動式相互關聯技術」規則語言是用來撰寫相互關聯事件之規則的 XML 型語言。此資訊是爲規則撰寫者所設計的，這些規則撰寫者必須知道如何撰寫規則，將其企業組織事件相互關聯。

目錄

第 1 篇 規則撰寫者手冊

第 1 章 簡介

此簡介簡要地說明複式事件處理程序 (也稱爲 CEP)，並提供複式事件處理程序中之「主動式相互關聯技術」及其角色的概觀。

現今的商業環境

現今，商業及政府組織都依賴使用電腦網路 (尤其是網際網路) 來處理電子資訊。並使用其他技術 (如網格運算)，在世界各地隨時執行重要任務的應用程式。商業程序、活動、基礎架構、乃至我們整個社會都依賴組織的資訊技術 (IT) 層。

組織需要隨時瞭解其業務狀況。例如，它們需要瞭解重要任務的應用程式是否可供使用且正常運作，以及如何偵測及防止商業程序、活動或基礎架構中潛伏的危機。如果發生危機，它們需要立即瞭解問題、解決它的方式，以及問題的原因。

因爲當資訊以個別獨立的組件形式存在時，其數量會非常巨大且難以融會貫通，所以根本無法辨識或瞭解與商業程序、活動及基礎架構相關之大多數事件的重要性。然而，如果將事件聚集並讓它們相互關聯，便能容易地瞭解其關係，則它們可以生產大量的資訊。

複式事件處理程序的目的在於能即時取得事件較佳的資訊。

複式事件處理程序

*事件*只是已發生之事情的通知。

*複式事件處理程序*是在事件驅動系統中分析、關聯及彙總低層次事件而衍生的高層次事件。這些高層次事件 (稱爲複式事件) 適合以簡單易懂的術語來通知人們商業良機或問題，或者觸發自動化程序。然後，組織可以使用對可能的良機或問題的提前警告，藉由更瞭解其商業程序、活動或基礎架構中狀況變更的起因，更有效地進行運作。

*事件相關性*程序會即時定義及偵測事件串流中的型樣，並實施動作以回應相關事件。使用它來根據所偵測到的症狀去識別問題。可以依原因、時間、成員資格或所有項目的組合，讓事件相互關聯。事件相關性是複式事件處理程序不可或缺的部分。

主動式相互關聯技術

「主動式相互關聯技術」會使用規則來即時偵測事件串流中的型樣。此技術以瞭解爲基礎，在大多數情況下，不應使用單一低層次事件來觸發回應動作，而應使用在不同時間及不同環境定義中發生之事件的複式組合來觸發回應動作。「主動式相互關聯技術」會利用事件之間的關係來瞭解商業良機及問題。例如，組織可以根據即時從事件相互關係中察覺到的商業狀況，執行下列類型的動作：

- 在假期銷售期間向某些或所有客戶提供折扣的出貨。
- 在接下來的 30 天內，根據出貨承運商、訂購量及訂單數量來計算出貨成本。
- 對那些在 2005 年 7 月 1 日至 2005 年 12 月 31 日之間購買貨品價值超過 500 美元的客戶寄送價值 25 美元的禮券。
- 如果在 36 小時之內有任何尚未完成的訂購處理，則通知管理者。

- 如果在 30 秒內偵測到相同電腦上有超過四個登錄嘗試，則通知管理者。

「主動式相互關聯技術」包括下列主要項目：

「主動式相互關聯技術」規則語言 (Active Correlation Technology rule language)
用來撰寫讓事件相互關聯之規則的 XML 型語言。然後可以將這些規則部署到「主動式相互關聯技術」執行時期環境。

「主動式相互關聯技術」引擎 (Active Correlation Technology engine)
爲「主動式相互關聯技術」元件，會根據「主動式相互關聯技術」編譯器的輸出來處理事件。

「主動式相互關聯技術」規則建置器 (Active Correlation Technology rule builder)
使用「主動式相互關聯技術」規則語言來撰寫相關性規則的 GUI。

「主動式相互關聯技術」執行時期環境是內嵌「主動式相互關聯技術」引擎的應用程式。

第 2 章 規則語言概觀

此概觀說明「主動式相互關聯技術」規則語言的主要概念。

規則型樣代表事件相關性狀況 (例如臨界條件或偵測重複事件)。「主動式相互關聯技術」規則語言包括七種規則型樣，且已證實它們可代表 IBM 客戶需要處理的大部分事件相關性狀況。七種規則型樣中有六種是定義有狀態的規則，而有一種型樣是定義無狀態的規則。

有狀態的規則會將特定時段內發生的多個事件相互關聯，並產生這些事件的回應。無狀態的規則只會處理符合特定條件的單一事件，並產生該事件的回應。

有狀態的規則 (stateful rule)

保留狀態資訊的規則，它是規則實例之性質的相關資訊，目的是在一段時間內執行事件收集。由下列任何規則型樣定義的規則都是有狀態的規則：收集、計算、重複、序列、臨界值或計時器。

無狀態的規則 (stateless rule)

不保留狀態資訊的規則，因此一次只能在一個事件上執行。由過濾型樣定義的規則是無狀態的規則。

參考資訊

第 41 頁的第 4 章, 『規則集組織摘要』
此參照列出規則集、規則區塊，以及每種規則類型的所有語言元素。您可以將它視爲編碼規則集的簡易說明。

第 53 頁的第 5 章, 『語言元素參照』
此參照說明「主動式相互關聯技術」規則語言之 XML 綱目中語言元素的詳細資訊。語言元素會按字母順序列出， 如需每個元素可以使用之屬性的說明，請參閱該元素的主題。

第 107 頁的第 6 章, 『名詞解釋』
此名詞解釋包含「主動式相互關聯技術」中之重要概念的術語及定義。

規則剖析

規則最基本的組成部分爲事件選擇、分組鍵、有狀態之規則的時間範圍、規則回應、啓用間隔，以及生命週期動作。規則還包括表示式及變數。表示式是包含自訂邏輯的程式碼，您可以將這些自訂邏輯新增至規則。

事件選擇

事件選擇準則可決定規則會接受哪些事件來進行處理。<eventSelector> 元素會爲規則定義事件選擇準則。事件選擇會套用至所有規則 (但是由計時器型樣所定義的那些規則除外)。因爲計時器規則不處理事件，所以它不包含事件選擇準則。

分組鍵

通常，每個作用中的規則都有一個規則實例 (或複本) 在「主動式相互關聯技術」引擎中執行。然而，有時不同的事件群組會需要相同的規則，而這些事件群組通常與不同的資源群組有關。分組鍵是針對具有相同性質的每個事件群組，指引規則建立個別規則實例 (或其複本) 的方法。

分組鍵可視爲是另一種形式的事件選擇。若使用分組鍵來定義規則，且規則收到的事件具有由分組鍵來定義的性質，則會將該事件傳送到正在處理具有相同性質之事件的規則實例。例如，您可以定義一個規則來收集類型爲 `Audit Failure` 的所有安全事件，並將分組鍵定義爲事件的 hostname 屬性。現在可以多次使用規則，讓 hostname 屬性的每個唯一值，都執行個別的規則複本。您也可以監視接收 `Audit Failure` 事件的所有系統，以判定在 2 分鐘的時段內，每個主機名稱是否都會發生 10 次以上此類事件。

<groupingKey> 元素會定義規則的分組鍵，且該元素只對收集、計算、重複、序列及臨界值型樣所定義的規則有效。

有狀態之規則的時間範圍

因爲有狀態的規則會與在特定時段內發生的多個事件產生關聯，所以有狀態之規則的基本組成部分是時間範圍 (由 <timeWindow> 元素來定義)。時間範圍會指定一個時段，在此時段內，有狀態的規則會進行比對其型樣的處理程序。

規則回應

規則回應動作會定義規則完成其處理程序時要採取的動作。下列每種語言元素都定義不同類型的規則回應動作：

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

規則可以使用的規則回應動作類型視規則型樣而定。

啓用間隔

啓用間隔會定義規則處於作用中及非作用中的時間。<activationInterval> 元素會定義規則的啓用間隔。

規則可以在離散的時間點上或者依特定事件啓用或停用。

如果您指定規則在離散的時間點上*且* 依特定事件啓用或停用，則會根據最先發生的是時間點或事件的接收，來啓用或停用該規則。然而，在此情況下，規則在其整個生命週期中可能會被多個事件啓用或停用。例如，規則可能依事件啓用、停用、在定義的時間點上啓用、再停用，然後依其他事件啓用。

<activationByGroupingKey> 元素是內含在 <activationInterval> 元素中的一個元素。<activationByGroupingKey> 元素包含的元素可以指定事件來啓用及停用 <groupingKey> 元素所定義的規則實例。

生命週期動作

生命週期動作會定義在規則生命週期的下列四個主要階段要採取的動作：載入、啓用、停用及卸載。

<lifeCycleActions> 元素包含定義這些動作的下列元素：

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

規則的生命週期

規則生命週期中的每個階段都可以具有多重原因及影響。藉由將表示式寫入及包括在生命週期動作中 (如 <lifeCycleActions> 元素所定義)，規則撰寫者可以定義在每個階段要採取的動作。

規則生命週期中的階段

下列爲規則生命週期的四個主要階段：

載入 將規則載入執行中的「主動式相互關聯技術」引擎，可觸發 <onLoad> 元素中的動作。

啓用 啓用規則，可觸發 <onActivation> 元素中的動作。

停用 停用規則，可觸發 <onDeactivation> 元素中的動作。

卸載 從執行中的「主動式相互關聯技術」引擎卸載規則，可觸發 <onUnload> 元素中的動作。

在規則的生命週期中，啓用及停用階段可以出現多次，但是載入及卸載階段只會出現一次。

通常，您不需要定義生命週期動作。下列是您可能想要定義特定生命週期動作時的範例：

- 載入特定規則之後，您可能會想建立與在該規則內需要存取之外部系統 (如資料庫管理程式) 的連線。而在卸載此相同的規則時，則要捨棄該連線並執行所有必要的清除程序。
- 啓用特定規則之後，您可能會想驗證該規則是否可使用特定資源。
- 停用臨界值規則，但尚未到達臨界值且時段尚未結束時，您可能會想將含有此資訊的郵件轉寄給某人。

因爲在生命週期中可以發生多次啓用及停用規則的動作，所以會經常執行您爲這些階段編碼的任何動作。

每個生命週期階段的原因及影響

表 1列出每個生命週期階段的原因及影響。

表 1. 每個生命週期階段的原因及影響

生命週期階段	原因	影響
載入	適用於下列任何情況： • 新增或取代規則或規則區塊，會載入新的規則。 • 取代「主動式相互關聯技術」引擎中的規則集，會載入新規則集中的規則。	執行 <onLoad> 元素中的動作。
啓用	已啓用規則。可以採用下列任何一種方式來啓用規則： • 根據 <activationInterval> 元素中的定義 • 透過利用 act_lib 變數來存取的 activate() 方法 • 透過應用程式呼叫「主動式相互關聯技術」引擎中的 activate() 方法	如果規則不在作用中，則會執行 <onActivation> 元素中的動作。
停用	已停用規則。可以採用下列任何一種方式來停用規則： • 根據 <activationInterval> 元素中的定義 (只是 <activationByGroupingKey> 元素中的 <deactivateOnEvent> 元素不會造成規則停用) • 透過利用 act_lib 變數存取的 deactivate() 方法 • 透過應用程式呼叫「主動式相互關聯技術」引擎中的 deactivate() 方法	如果規則在作用中，則會執行 <onDeactivation> 元素中的動作。
卸載	適用於下列任何情況： • 關閉「主動式相互關聯技術」引擎，可卸載規則。 • 移除或取代規則或規則區塊，可卸載舊的規則。 • 移除或取代「主動式相互關聯技術」引擎中的規則集，可卸載舊規則集中的規則。	如果規則在作用中，則會執行 <onDeactivation> 元素中的動作，然後執行 <onUnload> 元素中的動作。否則，只會執行 <onUnload> 元素中的動作。

規則的組織

「主動式相互關聯技術」規則語言會將規則組織到屬於規則集之一部分的規則區塊中。

規則集

規則集包含「主動式相互關聯技術」引擎要執行的規則 (這些規則會組織到規則區塊中)。它是規則執行單位。每個「主動式相互關聯技術」引擎每次僅處理一個規則集。

傳送至「主動式相互關聯技術」引擎的事件會觸發規則集中包含的規則。然後根據每個規則的事件選擇準則將事件連續傳遞給適當的規則，而且每次只執行一個規則。相同的事件可套用，因此會觸發多個規則。這些規則不一定必須相關，但可能相關。

規則區塊及規則集內之規則的次序能決定規則集中的事件流程。

可以在規則集層次定義變數及匯入，以用於整個規則集範圍內的表示式 (包含自訂邏輯的程式碼)。匯入是存取外部程式碼的一種程式設計語言的方式。規則撰寫者可以定義匯入來匯入外部模組 (例如，Java™ 類別)，以用於規則內的表示式。

規則區塊

規則區塊是一種組織單位，可根據功能將規則分組成規則集中的網域。網域是一種種類，規則群組會根據規則的功能來套用至其中。例如，網域可以代表特定的地理區域、IT 管理紀律 (例如安全偵測或網路事件相互關聯)，或商業組織 (例如特定的公司或公司的部門)。

規則區塊可以包含規則及其他的規則區塊。因爲規則區塊可以是巢狀的，所以可以建構規則的階層。例如，規則集可能包含網路事件相互關聯的規則區塊，而網路事件相互關聯的規則區塊可能包含兩個其他規則區塊：一個屬於層次 2 相互關聯，另一個屬於 IP 相互關聯。

因此，規則集會爲多種網域提供事件相互關聯功能，而規則區塊會爲可能需要存取相似事件集合的組織提供不同的網域。

可以在規則區塊層次定義變數及匯入，以用於整個規則區塊範圍內的表示式。規則區塊的範圍包括規則區塊中包含的所有規則及其他規則區塊。

規則

規則是相互關聯單位，可用來識別事件之間的關係，以及執行適當的規則回應。規則是下列七種規則型樣之其中一種的實作，並且會根據其功能組織成屬於規則集之一部分的規則區塊：

- 收集型樣
- 計算型樣
- 重複型樣
- 過濾型樣
- 序列型樣
- 臨界值型樣
- 計時器型樣

每個規則都可以根據其型樣提供唯一的事件相互關聯功能，並且可以透過事件轉遞來鏈接規則。透過鏈接規則，可以將不同型樣的事件相互關聯功能相互結合或組成巢狀結構。

可以在規則層次定義變數，以用於整個規則範圍內的表示式。

摘要

簡而言之，規則集是執行單位、規則區塊是組織單位，而規則是相互關聯單位。規則集包含一或多個規則區塊，每個規則區塊還可以包含其他的規則區塊。每個規則區塊都包含特定網域的規則。規則區塊可以是巢狀的，來建構規則階層。規則區塊及規則集內之規則的次序能決定規則集中的事件流程。

可以在規則集或規則區塊層次定義變數及匯入，以用於規則內的表示式。變數或匯入的範圍是各規則集或規則區塊。也可以在規則層次定義變數，但是這會將它們的範圍限制爲規則。

規則型樣

規則型樣代表事件相關性狀況 (例如臨界條件或偵測重複事件)。「主動式相互關聯技術」規則語言會定義下列規則型樣：收集、計算、重複、過濾、序列、臨界值及計時器。

當發生規則定義的狀況時，就會比對規則型樣。比對型樣時，規則會採取適當的規則回應動作來結束其處理程序。當規則處於作用中時，可能會多次比對規則型樣。

過濾型樣定義的規則是規則語言中唯一無狀態的規則。所有其他規則都是有狀態的。

收集型樣

收集規則是由收集型樣來定義。它會在時間間隔內收集選定的事件群組。它是有狀態的規則。

概觀

收集型樣可用來彙集一段時間內的相似事件。時段是由必要的時間範圍來指示，如規則語言中的 <timeWindow> 元素所定義。

執行規則回應的條件

使用收集型樣，規則回應就會在時間範圍結束時執行，如 <onTimeWindowComplete> 元素所定義。

此規則型樣的範例用法

收集型樣的範例用法是執行下列作業的規則：

此規則會在規定的時段內收集符合特定事件選擇器準則的事件。當此時段結束時，它會將收集到的事件彙總成單一事件，其中包含事件的總數，以及有關所彙總之事件的性質資訊。

相關參考

第 42 頁的『收集規則摘要』
此摘要列出收集規則的所有語言元素。

計算型樣

計算規則是由計算型樣來定義。在某個時間間隔內，接收到每個事件時，此規則便會將計算 (透過表示式) 套用至收集到的事件。它是有狀態的規則。

概觀

計算型樣會對一段期間內接受的每個事件執行計算功能，如規則語言中的 <computeFunction> 元素所定義。時段是由必要的時間範圍來指示，如 <timeWindow> 元素所定義。

執行規則回應的條件

使用計算型樣，規則回應就會在時間範圍結束時執行，如 <onTimeWindowComplete> 元素所定義。在 <onTimeWindowComplete> 動作期間，可以使用計算的值。

此規則型樣的範例用法

假設應用程式正在處理客戶訂單事件。計算型樣的範例用法是執行下列作業的規則：

每次接收到事件時，都會將訂單總值加入指定時段內發生的所有訂單總值，並在使用者介面中發佈已更新的所有訂單總值。

相關參考

第 44 頁的『計算規則摘要』
此摘要列出計算規則的所有語言元素。

重複型樣

重複規則是由重複型樣來定義。此規則會計算在指定時間間隔內接受的第二個及後續事件數，但會略過這些事件的規則集處理程序。它是有狀態的規則。

概觀

通常，重複型樣可用來隔離一段時間內的相似 (重複) 事件。重複事件在某方面與前一個事件類似，但不一定完全是該事件的複本。只要事件符合規則的事件選擇準則，它們就是重複事件。時段是由必要的時間範圍來指示，如規則語言中的 <timeWindow> 元素所定義。

執行規則回應的條件

使用重複型樣，規則回應就會在下列時間執行：

- 偵測到第一個事件的時候，如 <onDetection> 元素所定義。
- 處理每個重複事件的時候，如 <onNextEvent> 元素所定義。
- 時間範圍結束的時候，如 <onTimeWindowComplete> 元素所定義。

即使未收到任何重複事件，第一個事件也會觸發 <onDetection> 動作。此行爲的原因是您可能想要轉遞第一個事件，並略過重複事件的規則集處理程序。在此情況下，您可以新增規則回應動作，在觸發規則的 <onDetection> 動作時轉遞第一個事件。

重複事件 (第二個及後續事件) 的預設處理是計算重複事件數，但略過重複事件的規則集處理程序。如果想要對重複事件採取其他動作，則可明確定義 <onNextEvent> 動作。例如，在某些情況下，重複事件代表可能已經記載到資料庫或其他儲存庫的事件。因此，您可能想要對 <onNextEvent> 動作編碼，以從這些其他位置移除重複事件。

<onTimeWindowComplete> 動作可用來爲所有重複事件建立摘要記錄，包括已處理的重複事件數目。

此規則型樣的範例用法

假設「拒絕服務」訊息不斷地從相同的資源類型 (安全監視器) 出現。這表示可能存在安全中斷。重複型樣的範例用法是執行下列作業的規則：

「拒絕服務」訊息從安全監視器出現之後，會計算 30 秒期間內發生之該事件的任何重複事件數，但不會將這些事件傳送至操作員主控台。另外，在 30 秒的時段結束時，規則會產生一個事件，指出在此時段內所發生的「拒絕服務」訊息數目。

相關參考

第 45 頁的『重複規則摘要』
此摘要列出重複規則的所有語言元素。

過濾型樣

過濾規則是由過濾型樣來定義。它會在接受事件時採取特定的動作。它只會在單一事件上執行，因此是無狀態的規則。

概觀

過濾型樣可用來在符合事件選擇準則的個別事件上執行。與其他規則型樣不同，它不會保留相關聯的狀態資訊 (例如，過去事件的歷程)。

執行規則回應的條件

使用過濾型樣，規則回應就會在接收到符合事件選擇準則的任何事件時執行，如 <onDetection> 元素所定義。

此規則型樣的範例用法

過濾型樣的範例用法是執行下列作業的規則：

如果 `ServerStatus` 事件指出 `serverLoad` 大於 95%，則規則會對管理者執行分頁動作。

相關參考

第 47 頁的『過濾規則摘要』
此摘要列出過濾規則的所有語言元素。

序列型樣

序列規則是由序列型樣來定義。此規則會偵測在時間間隔內是否有某個事件序列到達。序列可以是依序的，也可以是隨機的。序列規則是有狀態的規則。

概觀

序列型樣會檢查一段時間內的事件序列，並偵測序列是完整的還是不完整的。不完整的序列包含指定序列的一或多個事件，但並非所有的事件。

時段是由必要的時間範圍來指示，如規則語言中的 <timeWindow> 元素所定義。序列中的每個事件都是由規則中個別的 <eventSelector> 元素來定義。偵測到的事件序列可能會具有下列其中一種次序：

- 在規則中編碼 <eventSelector> 元素的次序。在此情況下，當規則偵測到第一個 <eventSelector> 元素所定義的事件時，會啓動序列偵測。此時，規則會等待第二個 <eventSelector> 元素定義的事件。
- 隨機次序。在此情況下，當規則偵測到 <eventSelector> 元素所定義的任何一個事件時，就會啓動序列偵測。此時，規則會等待 <eventSelector> 元素定義的另一個事件。

序列型樣不同於其他規則型樣，主要差異如下：

- 它使用多個 <eventSelector> 元素來定義規則接受的事件。它至少需要兩個 <eventSelector> 元素。
- 當事件符合其中一個 <eventSelector> 元素所定義的準則時，就會在該規則實例的進一步事件處理程序中，將該 <eventSelector> 元素排除在外。
- <eventSelector> 元素上的 alias 屬性僅在序列規則內有效，而且它會將特定事件選擇器在序列規則中所選取的事件命名爲唯一的名稱。在過濾述語或動作內的表示式中，您可以使用 act_eventList 變數，依別名來存取序列規則中的事件。

執行規則回應的條件

使用序列型樣，規則回應就會在下列時間執行：

- 在時間範圍內偵測到完整的事件序列時，如 <onDetection> 元素所定義。
- 在時間範圍內有一或多個事件到達但不是完整的序列時，如 <onTimeOut> 元素所定義。

  序列型樣可用來偵測給定的序列是否不完整。比方說，如果「系統關閉」事件發生後沒有發生後續的「系統啓動」事件，規則撰寫者可以編碼 <onTimeOut> 動作，以處理此類型的遺漏事件。

此規則型樣的範例用法

說明偵測完整序列的實務範例:

假設在 IT 環境中，管理者想要瞭解 DB2® 資料堆大小的值是否會影響 WebSphere® Application Server，如果會，則要更正此問題。因此，如果在指定的時段內發生下列事件且次序如下，則管理者會想要增加 DB2 資料堆大小的值，並重新啓動資料庫管理程式：

1. WebSphere Application Server 資源配置異常。假設此爲 `WASResourceAllocationException` 類型的事件。
2. DB2 錯誤訊息顯示：『沒有足夠的資料堆可處理陳述式』。假設此爲 `DB2NotEnoughHeap` 類型的事件。

針對此實務範例，在序列規則中定義兩個 <eventSelector> 元素，且事件必須依照編碼 <eventSelector> 元素的次序 (而不是隨機次序) 到達。第一個 <eventSelector> 元素檢查事件 `WASResourceAllocationException`，而第二個 <eventSelector> 元素則檢查事件 `DB2NotEnoughHeap`。假設會在指定的時間範圍內將下列事件呈現給系統：

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

規則行爲如下所示：

1. 接受第一個事件 `WASResourceAllocationException`。因爲符合第一個 <eventSelector> 元素的準則，所以現在會在此規則的進一步事件處理程序中，將第一個 <eventSelector> 元素排除在外。
2. 忽略第二個事件 `DB2BackupStarted`。

3. 忽略第三個事件 WASResourceAllocationException。
4. 忽略第四個事件 WASResourceAllocationException。
5. 接受第五個事件 DB2NotEnoughHeap，並完成序列。<onDetection> 規則回應動作會執行。定義此動作，以增加 DB2 資料堆大小的值，並重新啓動資料庫管理程式。規則回到起始狀態。

   現在，此規則會在進一步事件處理程序中包括第一個 <eventSelector> 元素。

說明偵測不完整序列的實務範例:
假設企業組織要在收到訂單要求的 1 個小時之內，準備好訂單交付給客戶，並且想要在未完成時獲得通知。

針對此實務範例，在序列規則中定義兩個 <eventSelector> 元素，且事件必須依照編碼 <eventSelector> 元素的次序 (而不是隨機次序) 到達。第一個 <eventSelector> 元素檢查 operationType=Order 的 Netsales 事件，而第二個 <eventSelector> 元素則檢查 operationType=Delivery 的 Netsales 事件。假設會在指定的 1 小時時間範圍內將下列事件呈現給系統：

1. operationType=Order 的 Netsales 事件
2. operationType=Order 的 Netsales 事件

規則行爲如下所示：

1. 接受第一個事件。因爲符合第一個 <eventSelector> 元素的準則，所以現在會在此規則的進一步事件處理程序中，將第一個 <eventSelector> 元素排除在外。
2. 忽略第二個事件。
3. 因爲 operationType=Delivery 的 Netsales 事件未在指定時間範圍內接收，<onTimeOut> 規則回應動作即會執行。定義此動作，以通知業務專員，在收到訂單要求的 1 個小時內尚未準備好訂單交付給客戶。規則回到起始狀態。

   現在，此規則會在進一步事件處理程序中包括第一個 <eventSelector> 元素。

   相關概念

   第 20 頁的『存取事件相關資訊』
   下列範例指出您如何透過「主動式相互關聯技術」提供的變數來存取事件相關資訊。

   相關參考

   第 48 頁的『序列規則摘要』
   此摘要列出序列規則的所有語言元素。

臨界值型樣

臨界值規則是由臨界值型樣來定義。此規則會收集某個時間間隔內的選定事件群組，並判定在收到每個事件之後，是否符合臨界條件。它是有狀態的規則。

概觀

臨界值型樣會收集某個時段內的事件，直到符合臨界值爲止。時段是由必要的時間範圍來指示，如規則語言中的 <timeWindow> 元素所定義。

臨界值型樣提供下列三個臨界值類型選項：

事件計數臨界值

使用這個臨界值類型，您可以定義某個時段內必須符合事件選擇準則的事件數目。定義的臨界值會與接受的事件數目進行比較。當事件計數與時間範圍內定義的限制相等時，便符合臨界值。

此類型的臨界值可用於非常簡單的事件計數檢查。例如，它可以回答問題：「1 分鐘內是否已發生 5 次登入失敗事件？」

此臨界值是由 <eventCountThreshold> 元素來定義。<eventCountThreshold> 元素還指定下列兩種可能的時間間隔模式之一作爲時間範圍：

固定間隔

固定間隔開始於接收到第一個符合事件選擇準則的事件時，並結束於發生下列其中一項狀況時：

- 規則符合其在指定持續期間內的臨界值。
- 已經過指定的持續期間。

可調整的間隔

可調整的間隔開始於接收到第一個符合事件選擇準則的事件時。不過，當規則尚未符合其臨界值且已經過指定的持續期間時，時間範圍會將開始時間調整爲新的「第一個」事件 (通常是下一個接受的事件) 的事件接收時間。可調整的間隔會繼續以此方式調整，直到發生下列其中一項狀況爲止：

- 規則符合其在指定持續期間內的臨界值。
- 接收到開始時間範圍的事件之後，在指定的持續期間內未接收到後續事件。

開始時間範圍的事件 (變成新的「第一個」事件) 的接收時間符合以下準則：新增至規則之時間間隔期間的接收時間大於目前的時間。下面是方程式形式的準則：

事件接收時間 + 規則的時間間隔期間 > 目前的時間

當不存在此類事件時，可調整的間隔就無法再調整時間，間隔便會結束。

計算的臨界值

使用此類型的臨界值，您可以撰寫程式碼 (或使用其他人撰寫的程式碼)，對每個接受的事件執行計算，並傳回計算的臨界值，而此值會儲存在先前定義的變數中。然後，將這個計算的臨界值與定義的臨界值進行比較，以判定是否符合臨界值。

因此，可以套用複式計算來建立 (或更新) 計算的臨界值 (可能使用先前事件中所儲存的資料)，因此規則撰寫者可以不依賴計算臨界值的計算邏輯來設定定義的臨界值。

此類型的臨界值可用於將值與定義的臨界值進行合計與比較。例如，您可以使用它來計算某個時段內，某位客戶的銷售金額總計，並將該總額與定義的臨界值進行比較。

此臨界值是由 <computedThreshold> 元素來定義。

布林臨界值

使用此類型的臨界值，您可以撰寫程式碼 (或使用其他人撰寫的程式碼)，針對

每個接受的事件傳回值 `true` 或 `false`。如果值爲 `true`，則符合臨界值。如果值爲 `false`，臨界值規則會繼續處理，直到時段結束，或它接受另一個事件爲止。

此類型的臨界值可用於檢查值的範圍。比方說，如果無論何時 CPU 的使用率都必須介於 30% 至 80% 之間，則此臨界值可能會不斷地驗證使用率是否保持在該範圍之內。

此臨界值是由 <booleanThreshold> 元素來定義。

執行規則回應的條件

使用臨界值型樣，規則回應就會在下列時間執行：

- 符合臨界值的時候，如 <onDetection> 元素所定義。
- 已接受一或多個事件，但在時間範圍內尚未符合臨界值的時候，如 <onTimeOut> 元素所定義。

此規則型樣的範例用法

臨界值型樣與事件計數臨界值搭配的範例用法是執行下列作業的規則：

如果在 30 秒的可調整時間間隔內，同一個子網路中產生 4 個以上的**無法存取伺服器**事件，則規則會執行動作來檢查路由器的狀態。

相關參考

第 49 頁的『臨界值規則摘要』
此摘要列出臨界值規則的所有語言元素。

計時器型樣

計時器規則是由計時器型樣來定義。此規則會定期起始動作。它是有狀態的規則。雖然計時器規則不處理事件，但可以使用事件來啓用或停用它。

概觀

計時器型樣與計時器類似，會在時段開始時啓動，並在時段結束時停止。時段是由必要的時間範圍來指示，如規則語言中的 <timeWindow> 元素所定義。

除非指定不重複，否則計時器型樣會一直重複，直到停用計時器規則爲止。因此，當計時器規則啓動時，會先等待指定的時段，然後再起始任何動作，並且一直重複此行爲，直到停用規則或「主動式相互關聯技術」引擎關閉爲止。

計時器規則是唯一的，因爲它不包含事件選擇準則。計時器規則會根據規則的啓用間隔來啓動處理程序，如 <activationInterval> 元素所定義。如果使用預設 <activationInterval> 元素，並將計時器型樣設爲重複，則計時器規則會在「主動式相互關聯技術」引擎將它載入後啓動，而在「主動式相互關聯技術」引擎關閉時停止。若要使用事件來啓用計時器規則，您必須在規則之 <activationInterval> 元素內的 <activateOnEvent> 元素中指定事件。

執行規則回應的條件

使用計時器型樣，規則回應就會在時間範圍結束時執行，如 <onTimeWindowComplete> 元素所定義。

此規則型樣的範例用法

計時器型樣可用於實施清除規則。計時器型樣的範例用法是執行下列作業的規則：

規則會每 30 分鐘執行一次動作，來清除開啓時間已超過 48 小時的無害參考事件。

相關參考

第 51 頁的『計時器規則摘要』
此摘要列出計時器規則的所有語言元素。

不同規則型樣的共同及唯一方面

此矩陣會提供不同規則型樣之共同及唯一方面的高階概觀。

表 2列出規則的主要語言元素，並在有效元素的每個規則類型直欄中顯示 X。主要語言元素是不同規則類型的直屬子元素。此清單不包括這些直屬子元素中所包含的元素，而且會視規則類型而改變。此外，某些元素屬性的有效性也會視規則類型而改變。

表 2. 顯示不同規則型樣之共同及唯一方面的矩陣

元素	收集	計算	重複	過濾	序列	臨界值	計時器
<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

表示式

表示式是包含自訂邏輯的程式碼，您可以將這些自訂邏輯新增至規則。表示式還可以存取「主動式相互關聯技術」引擎的外部程式碼。在規則語言中，表示式僅在特定環境定義或規則語言元素內才有效。

規則撰寫者可以根據環境定義及要取得的結果，編碼不同用途的表示式。表示式經常用來起始設定變數、定義事件選擇準則，及指定規則回應及生命週期動作。

包含表示式的語言元素

每個包含表示式的語言元素都具有 expressionLanguage 屬性，此屬性可識別用來撰寫表示式的程式設計語言。Java 程式設計語言是唯一受支援的表示式語言。

表示式可以內含在下列規則語言元素中。

- 規則集、規則區塊或規則變數的 <varInitializer>

- <eventSelector> 上的 <filteringPredicate>
- <groupingKey> 上的 <computedValue>
- 計算規則上的 <computeFunction>
- 臨界值規則上的 <booleanThreshold>
- 臨界值規則上的 <computedThreshold>
- 規則的規則回應動作：
  - <onDetection> 中的 <action>。此動作僅對重複、過濾、序列及臨界值規則有效。
  - <onNextEvent> 中的 <action>。此動作僅對重複規則有效。
  - <onTimeOut> 中的 <action>。此動作僅對序列及臨界值規則有效。
  - <onTimeWindowComplete> 中的 <action>。此動作僅對收集、計算、重複及計時器規則有效。
- 規則的生命週期動作：
  - <onLoad> 中的 <action>
  - <onActivation> 中的 <action>
  - <onDeactivation> 中的 <action>
  - <onUnload> 中的 <action>

主動式相互關聯技術為編碼表示式而提供的功能

爲了協助規則撰寫者編碼表示式，「主動式相互關聯技術」會提供執行下列動作的功能：

- 匯入用於表示式的外部模組 (例如，Java 類別) 及物件。
- 起始設定及存取規則集、規則區塊或規則變數。
- 透過 act_event 變數存取規則正在處理的目前事件。
- 透過 act_eventCount 變數存取規則已接受的事件數目。
- 透過 act_eventList 變數存取規則已接受的事件清單。這包括存取事件的各種屬性，以及依別名存取序列規則中每個事件的功能。
- 透過 act_lib 變數存取包括取得及設定變數，以及透過規則集控制事件流程之功能的方法。
- 透過 act_location 變數存取表示式之規則階層內的位置。
- 透過 act_nodeName 變數存取節點的完整名稱。
- 透過 act_threshold 變數存取臨界值規則的已定義臨界值。

匯入及存取外部模組及物件

此範例指出您如何讓表示式可以存取外部程式碼 (例如，Java 類別) 及外部物件。外部物件爲應用程式建立用來與表示式通訊的物件。

在您從表示式存取外部程式碼之前，您必須讓表示式可以存取該程式碼。

匯入是讓表示式可以存取外部程式碼的程式設計語言專用方式。<import> 元素包含特殊類型的表示式，此表示式可以指定要匯入以在規則內的其他表示式中使用的外部模組 (例如，Java 類別)。您可以在規則集或規則區塊層次定義匯入。

下列 <import> 元素包含以 Java 程式設計語言撰寫的表示式，此表示式會匯入可從其他表示式參照的 StaticHelper 類別及 Queue 類別：

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

雖然在 import 陳述式中不需要使用完整的類別名稱，您仍然應該指定完整名稱以防止編譯時間過長。例如，應該將 Java 類別指定爲 com.ibm.act.sample.StaticHelper，而不是 com.ibm.act.sample.* 或 com.ibm.act.*。

存取靜態方法

下列範例指出規則回應動作內的表示式如何在匯入 StaticHelper 類別之後參照它：

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

存取物件的實例方法

下列範例指出規則回應動作內的表示式如何在匯入 Queue 類別之後參照它。在此範例中，會取得名稱爲 OutputQueueOne 及類型爲 Queue 的外部物件，可用來將事件放置在特定佇列中。

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

起始設定及存取變數

此範例指出您可以如何起始設定及存取規則集、規則區塊或規則變數。

您可以在規則集、規則區塊或規則層次定義變數。必須先使用起始設定表示式來起始設定變數，才能存取它。下列表示式會起始設定兩個變數，其名稱分別是 hostsList 及 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>
```

可以透過表示式存取所有變數。下列範例顯示如何在規則回應動作內，透過表示式存取在前述範例中起始設定的 hostsList 及 hostsString 變數。在此範例中，會修改 hostsList，並且會爲 hostsString 給定新值。

```
<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>
```

存取事件相關資訊

下列範例指出您如何透過「主動式相互關聯技術」提供的變數來存取事件相關資訊。

存取目前事件的範例:

下列程式碼顯示如何使用 act_event 變數，以取得事件的 hostname 屬性：

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

透過依索引排列之事件清單來存取事件的範例:

下列程式碼顯示如何使用 act_eventList 變數，以取得事件清單中的第一個事件：

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

透過依別名排列之事件清單來存取事件的範例:

與其他規則類型不同，序列規則容許使用多個事件選擇器，而且實際上至少需要兩個事件選擇器。<eventSelector> 元素上的 alias 屬性僅在序列規則內有效，而且它會將特定事件選擇器在序列規則中所選取的事件命名爲唯一的名稱。在過濾述語或動作內的表示式中，您可以使用 act_eventList 變數，依別名來存取序列規則中的事件。

下列程式碼顯示序列規則的兩個事件選擇器。別名分別是 TECevent 及 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>
```

下列程式碼顯示如何使用 act_eventList 變數，以取得第一個事件選擇器 (名爲 TECevent) 已接受的事件：

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

編碼表示式的最佳實務範例

此資訊包括有效率地編碼表示式的一些最佳實務範例、提示及秘訣。

- 爲了便於理解，此資訊所提供的大部份表示式範例包括直接在 XML 建構內的 Java 程式碼。然而，當您建立規則時，最佳實務範例是使用外部模組來包含 Java 程式碼，並將這些外部模組作爲表示式的一部分進行呼叫。

  您還可以在規則建置器中使用或編輯現有片段，或建立新片段，以提供用來呼叫外部模組的程式碼。片段是可以用於表示式之原始碼的摘錄。在規則建置器中，可透過「片段」檢視畫面取得片段。

  透過這種方式，您可以選擇在整合開發環境 (IDE) 中，完成設計、開發、編輯、測試及除錯 Java 程式碼的作業，並且可以將其作爲開發處理程序的標準部分來管制。

- 若要避免將表示式代碼當作 XML 來剖析，請將代碼含括在 CDATA 區段中，其中 <![CDATA[緊接在代碼之前，而]]> 則緊隨在代碼之後，如下列範例所示：

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

  XML Parser 會忽略 CDATA 區段內的所有內容。
- 如果您在表示式中使用 act_lib.getExternalContext() 方法，請不要儲存從規則集或規則區塊變數的方法所傳回的物件。原因在於應用程式可以變更物件實例的參照，而相關的規則集或規則區塊變數卻不會更新。
- 如果您在 <action> 元素內的表示式中使用 return 陳述式 (return;)，而其他 <action> 元素是爲各自的規則回應或生命週期動作編碼，則代碼執行會在編碼 return 陳述式的地方結束，並在下一個 <action> 元素內的表示式中重新開始。
- 無法從規則回應或生命週期動作內呼叫規則管理及其他「主動式相互關聯技術」引擎方法。

相關資訊

第 18 頁的『匯入及存取外部模組及物件』
此範例指出您如何讓表示式可以存取外部程式碼 (例如，Java 類別) 及外部物件。外部物件爲應用程式建立用來與表示式通訊的物件。

第 37 頁的『在規則內的表示式中併入片段』
在規則內的表示式中，您可以併入「Eclipse 工作台」之「片段」檢視畫面中的片段。

變數

在規則語言中，會使用特定的變數，儲存在不同次數出現的事件或規則之中的事件相關資訊。這個與事件相關的資訊可從規則內的表示式來存取。部分類型的變數是由規則撰寫者來定義，而其他類型的變數則由「主動式相互關聯技術」提供。部分類型可以直接從表示式存取，而其他類型則只能透過「主動式相互關聯技術」所提供的方法來存取。

在 <variable> 元素中定義並透過方法來存取的變數

您可以在規則、規則區塊或規則集的 <variable> 元素中定義變數。然後，您可以使用下列其中一個方法，存取表示式中的這個變數：

- getVariable() 方法或其中一個 get*javatype*Variable() 方法
- setVariable() 方法或其中一個 set*javatype*Variable() 方法

例如，如果您在規則的 <variable> 元素中定義變數 *rule_writer_variable*，您就可以使用下列程式碼來存取這個變數：

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

由「主動式相互關聯技術」提供且可直接在表示式中存取的變數

下列變數是由「主動式相互關聯技術」提供。您可以在表示式中直接使用這些變數。

- act_event
- act_eventList

- act_lib

例如，使用下列程式碼，您可以存取 act_event 變數，以取得事件的 hostname 屬性：

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

由「主動式相互關聯技術」提供並透過方法來存取的變數

下列變數是由「主動式相互關聯技術」提供。使用 getVariable() 方法或其中一個 get*javatype*Variable() 方法，您可以在表示式中存取這些變數。

- act_eventCount
- act_location
- act_nodeName
- act_threshold

例如，使用下列程式碼，您可以存取 act_eventCount 變數：

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

表 3顯示 IACTLibrary 介面爲這些變數提供的常數。在程式碼中，爲了確保可在編譯期間而不是在執行時期找到拼字或打字錯誤，請一律使用代表這些變數的常數，而不要使用變數本身。例如，使用 `act_lib.getIntVariable(IACTLibrary.EVENTCOUNT);` 而不用 `act_lib.getIntVariable("act_eventCount");`。

表 3. 具有相關常數的變數

變數	相關常數
act_eventCount	EVENTCOUNT
act_location	LOCATION
act_nodeName	NODENAME
act_threshold	THRESHOLD

相關參考

第 103 頁的『variable 元素』
<variable> 元素會定義變數，並包含可被表示式參照的格式資訊。您可以在規則集、規則區塊或規則層次定義變數。

「主動式相互關聯技術」變數的資料類型

「主動式相互關聯技術」所提供的變數具有不同資料類型。

表 4顯示這些變數的資料類型。

表 4. 「主動式相互關聯技術」變數的資料類型

變數	資料類型
act_ event	由 com.ibm.correlation.IEvent 介面所定義的類型
act_ eventCount	int
act_ eventList	由 com.ibm.correlation.IEventList 介面所定義的類型
act_ lib	由 com.ibm.correlation.IACTLibrary 介面所定義的類型
act_ location	java.lang.String
act_ nodeName	java.lang.String

表 4. 「主動式相互關聯技術」變數的資料類型 (繼續)

變數	資料類型
act_ threshold	此變數是臨界值規則之 <computedThreshold> 或 <eventCountThreshold> 元素的臨界值屬性值，因此資料類型必須與臨界值屬性值的資料類型相同。

變數在當中有效的表示式環境定義

「主動式相互關聯技術」提供的變數只有在特定表示式環境定義中有效。

表 5列出這些變數在當中有效的表示式環境定義。針對在個別表示式環境定義內有效的每個變數，該表格會在直欄中顯示 X。在各個變數的主題中說明了適用於這些變數的其他用法限制。

表 5. 變數在當中有效的表示式環境定義

表示式環境定義	act_ event	act_ eventCount	act_ eventList	act_ lib	act_ location	act_ nodeName	act_ threshold
<onActivation> 中的 <action>	X			X	X	X	X
<onDeactivation> 中的 <action>	X	X	X	X	X	X	X
<onDetection> 中的 <action>	X	X	X	X	X	X	X
<onLoad> 中的 <action>				X	X	X	X
<onNextEvent> 中的 <action>	X	X	X	X	X	X	
<onTimeOut> 中的 <action>		X	X	X	X	X	X
<onTimeWindowComplete> 中的 <action>		X	X	X	X	X	
<onUnload> 中的 <action>				X	X	X	X
<booleanThreshold>	X	X	X	X	X	X	
<computedThreshold>	X	X	X	X	X	X	X
<computedValue>	X			X	X	X	
<computeFunction>	X	X	X	X	X	X	
<filteringPredicate>	X	X	X	X	X	X	X
規則的 <varInitializer>				X	X	X	X

act_event 變數

act_event 變數可讓您存取適用於目前事件的方法。

詳細資訊

因爲計時器規則不處理事件，所以計時器規則中的 act_event 變數僅適用於啓用或停用規則的事件。

只有當事件啓用或停用規則時，act_event 變數才適用於 <onActivation> 及 <onDeactivation> 動作。否則，此變數爲空值。

編碼範例

下列程式碼將存取 act_event 變數，以取得事件的 hostname 屬性：

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

可存取的方法

act_event 變數可讓您存取的方法是在 IEvent 介面中定義，如表 6所示。

表 6. 含有相對應的方法及 Javadoc 方法說明位置的 IEvent 介面

介面	方法	Javadoc 方法說明的位置
IEvent	• get • getAttribute • getBooleanAttribute • getByteAttribute • getShortAttribute • getIntAttribute • getLongAttribute • getFloatAttribute • getDoubleAttribute • getStringAttribute • set • getTimeStamp • setTimeStamp • getType • getOriginal	com.ibm.correlation.IEvent

act_eventCount 變數

act_eventCount 變數是整數，它等於規則已接受的事件數目。

詳細資訊

若爲重複的規則，則 act_eventCount 變數的值是已接受事件的總數，其中包括原始事件及任何副本。若爲所有其他規則類型，此值便與事件清單的大小相同，您可以使用 act_eventList.size() 方法透過 act_eventList 變數取得該清單。

act_eventCount 及 act_eventList 變數在計時器規則內無效，因爲計時器規則不處理事件。

如果是以分組鍵來定義規則，則 act_eventCount、act_eventList 及 act_threshold 變數在下列表示式環境定義內無效：

- 生命週期動作
- <activateOnEvent> 內的 <filteringPredicate> 或 <activationInterval> 內的 <deactivateOnEvent>
- <computedValue>

這是因爲在此情況下，規則變數僅適用於規則實例，而規則實例在這些表示式執行時不存在。

編碼範例

下列程式碼將存取 act_lib 變數，以取得規則已接受的事件數目：

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

act_eventList 變數

act_eventList 變數可讓您存取適用於規則已接受之事件清單的方法。

詳細資訊

過濾規則及重複規則的清單永遠只包含一個事件，因爲過濾規則是無狀態的規則，且重複規則只保留第一個分析的事件。

act_eventCount 及 act_eventList 變數在計時器規則內無效，因爲計時器規則不處理事件。

如果是以分組鍵來定義規則，則 act_eventCount、act_eventList 及 act_threshold 變數在下列表示式環境定義內無效：

- 生命週期動作
- <activateOnEvent> 內的 <filteringPredicate> 或 <activationInterval> 內的 <deactivateOnEvent>
- <computedValue>

這是因爲在此情況下，規則變數僅適用於規則實例，而規則實例在這些表示式執行時不存在。

編碼範例

下列程式碼將存取 act_eventList 變數，以取得事件清單中的第二個事件：

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

可存取的方法

act_eventList 變數提供的存取方法是在 IEventList 介面中定義，如表 7所示。

表 7. 含有相對應的方法及 Javadoc 方法說明位置的 IEventList 介面

介面	方法	Javadoc 方法說明的位置
IEventList	• get • size • isEmpty • listIterator	com.ibm.correlation.IEventList

act_lib 變數

act_lib 變數提供對「主動式相互關聯技術」中之檔案庫方法的存取權。

詳細資訊

act_lib 變數可以存取的方法視包含使用該變數之表示式的規則語言元素而定。請參閱表 8。

表 8. act_lib 變數可存取的方法 (根據其包含表示式的內容)

表示式環境定義	IACTLibrary 方法	IExitableActionLibrary 方法	IActionLibrary 方法
<onActivation> 中的 <action>	X		
<onDeactivation> 中的 <action>	X		

表 8. act_lib 變數可存取的方法 (根據其包含表示式的內容) (繼續)

表示式環境定義	IACTLibrary 方法	IExitableActionLibrary 方法	IActionLibrary 方法
<onDetection> 中的 <action>	X	X	X
<onLoad> 中的 <action>	X		
<onNextEvent> 中的 <action>	X	X	X
<onTimeOut> 中的 <action>	X		X
<onTimeWindowComplete> 中的 <action>	X		X
<onUnload> 中的 <action>	X		
<booleanThreshold>	X		
<computedThreshold>	X		
<computedValue>	X		
<computeFunction>	X		
<filteringPredicate>	X		
<varInitializer>	X		

編碼範例

下列程式碼將存取 act_lib 變數，以呼叫用來結束規則集的方法：

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

可存取的方法

IACTLibrary、IExitableActionLibrary 及 IActionLibrary 介面中定義了 act_lib 變數可以存取的方法，如第 27 頁的表 9所示。

表 9. 具有相對應的方法及 Javadoc 方法說明位置的介面

介面	方法	Javadoc 方法說明的位置
IACTLibrary	• trace • getVariable • getBooleanVariable • getShortVariable • getIntVariable • getLongVariable • getFloatVariable • getDoubleVariable • getStringVariable • setVariable • setBooleanVariable • setShortVariable • setIntVariable • setLongVariable • setFloatVariable • setDoubleVariable • setStringVariable • getExternalContext	com.ibm.correlation.IACTLibrary
IActionLibrary	• forward • forwardOnCompletion • activate • deactivate IACTLibrary 介面中定義的方法也可以用於 IActionLibrary 介面。	com.ibm.correlation.IActionLibrary
IExitableActionLibrary	• exitRuleSet • exitRuleBlock IACTLibrary 及 IActionLibrary 介面中定義的方法也可以用於 IExitableActionLibrary 介面。	com.ibm.correlation.IExitableActionLibrary

act_location 變數

act_location 變數是字串，可識別表示式在規則階層中的位置。

詳細資訊

位置是完整名稱，可指出表示式在規則階層中的位置。它的格式爲 *identifier.identifier...*，其中每個 *identifier* 可以爲下列其中一項：

- 各個階層中之 XML 元素的名稱屬性值。
- 若爲在規則區塊或規則中出現多次且沒有名稱屬性的元素：包含表示式的 XML 元素，後面接著以方括弧包住的索引編號。此索引編號指出表示式在它的包含元素中

的位置。指定索引編號的計數器從 0 而不是 1 開始。因此，例如當元素包含在第三個 <action> 元素中時，索引編號顯示爲 action[2]。

這些 ID 以遞減次序排列，從最高層次的規則區塊至包含表示式的最低層次元素。

編碼範例

下列程式碼將存取 act_lib 變數，以取得表示式的位置：

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

從變數傳回的位置範例

下列值爲從 act_location 變數傳回的位置範例。

ruleBlockA.ruleA.eventSelector[3].filteringPredicate

此表示式包含在下列項目中：

- 名稱屬性值爲 ruleBlockA 的規則區塊
- 名稱屬性值爲 ruleA 的規則
- 第四個 <eventSelector> 元素
- <filteringPredicate> 元素

ruleBlockA.ruleA.onDetection.action[5]

此表示式包含在下列項目中：

- 名稱屬性值爲 ruleBlockA 的規則區塊
- 名稱屬性值爲 ruleA 的規則
- <onDetection> 元素
- 第六個 <action> 元素

ruleBlockA.ruleA.variableA.varInitializer

此表示式包含在下列項目中：

- 名稱屬性值爲 ruleBlockA 的規則區塊
- 名稱屬性值爲 ruleA 的規則
- 名稱屬性值爲 variableA 的變數
- <varInitializer> 元素

act_nodeName 變數

act_nodeName 變數是識別節點完整名稱的字串。

詳細資訊

在「主動式相互關聯技術」中，節點是規則階層內的物件，您可以從規則集內個別獨立地新增、移除或取代此規則階層。具體而言，下列物件爲節點：規則、規則區塊、規則區塊變數及規則集變數。因爲物件不能在規則層次下個別獨立地操作，所以規則變數不是節點。

名稱屬性值爲 rule1 的規則位於名稱屬性值爲 ruleBlockA 的規則區塊內，它的完整節點名稱是 ruleBlockA.rule1。因爲規則在規則集內是按階層組織的，所以在完整的節點名稱內會使用句點 (.) 來表示向下過渡到下層節點。

編碼範例

下列程式碼將存取 act_nodeName 變數，以取得節點的完整名稱：

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

act_threshold 變數

act_threshold 變數是臨界值規則之 <computedThreshold> 或 <eventCountThreshold> 元素上定義的臨界值屬性值。

詳細資訊

act_threshold 變數只在臨界值規則中有效。

如果是以分組鍵來定義規則，則 act_eventCount、act_eventList 及 act_threshold 變數在下列表示式環境定義內無效：

- 生命週期動作
- <activateOnEvent> 內的 <filteringPredicate> 或 <activationInterval> 內的 <deactivateOnEvent>
- <computedValue>

這是因爲在此情況下，規則變數僅適用於規則實例，而規則實例在這些表示式執行時不存在。

編碼範例

下列程式碼將存取 act_lib 變數，以取得已定義臨界值的值：

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

規則集中的事件流程

規則集中的事件流程會依照編碼規則區塊及規則的次序來進行。當「主動式相互關聯技術」引擎收到事件時，引擎會判定事件類型，並識別使用此事件類型的規則，以啓用規則、處理事件或停用規則。

規則使用事件的方式

使用事件的每個規則都會先判定事件是否符合啓用規則、處理事件或停用規則的所有指定準則。如果符合，規則會執行下列動作：

若為啓用規則
如果已編碼規則之 <onActivation> 元素中的動作，則會執行這些動作。

若為處理事件
規則會處理事件。當符合該規則型樣時，如果已編碼規則回應動作，則會執行這些動作。在某些情況下，規則回應動作可以執行下列作業：

- 動作可以讓事件略過在規則區塊或規則集剩餘部分中的處理。
- 動作可以將新的或現有的事件傳送至其他規則或規則區塊以進行處理。

若為停用規則
如果已編碼規則之 <onDeactivation> 元素中的動作，則會執行這些動作。

可影響事件流程的方法

「主動式相互關聯技術」會提供下列方法，您可以呼叫它們來影響規則集中的事件流程。可透過 act_lib 變數來使用這些方法。

exitRuleSet

此方法指定目前的事件不會由規則集中的任何其他規則來處理。

exitRuleBlock

此方法指定目前的事件不會由目前規則區塊或此規則區塊所含的任何規則區塊中的任何其他規則來處理。然而，會由目前規則區塊範圍之外的其他規則來處理。

forward

此方法指定即使目前的規則尚未完成事件處理，也會將事件傳送至其他規則及規則區塊。然後，每個其他規則及規則區塊都會在處理完事件後，將事件傳回至呼叫 forward 方法的規則。

forwardOnCompletion

此方法指定在目前的規則完成事件處理之後，將事件傳送至其他規則及規則區塊。

第 3 章 規則撰寫概觀

在撰寫規則以建立事件相關性之前，您必須瞭解及規劃事件相關性，並設計規則。您可以使用「主動式相互關聯技術」規則建置器來撰寫規則並編譯規則集。

「主動式相互關聯技術」規則建置器是用來撰寫規則的 GUI。它包括線上說明。在規則建置器中，產生的規則集檔案是檔案類型爲 actl 的 XML 檔案。

「主動式相互關聯技術」不提供變更管理或版本控制系統。

規劃事件相關性

規劃事件相關性包括瞭解 (或學習) 何謂事件相關性，以及如何在應用程式中套用事件相關性。

請確保您瞭解下列概念：

- 第 3 頁的第 1 章, 『簡介』及第 5 頁的第 2 章, 『規則語言概觀』中的資訊
- 應用程式處理的事件

每個應用程式都可以處理不同的事件集，如下列範例所述：

保險業務範例

在保險業務中，您可以產生事件來追蹤整個索賠程序中的工作流程，並建立事件相關性，以判定業務處理是否及時完成。

銷售範例

在不同類型的業務中，您可以定期彙總及報告銷售業績，並與目標相互比較，以指出某個時段銷售目標的達成狀態。

IT 環境範例

在 IT 環境中，擁有重要任務的系統可以每分鐘產生一個事件，以指出資料庫伺服器的執行狀態正常。您可以撰寫相關性規則，以監視這些活動訊號事件的接收作業，並在未收到預期的活動訊號事件時採取特定的規則回應動作。

您還應該瞭解應用程式處理的事件格式。「主動式相互關聯技術」提供Java 類別及方法來存取「主動式相互關聯技術」引擎正在處理之事件中的資料。然而，當在處理這些事件時，如果您想要使用這些類別及方法來存取它們，瞭解基礎事件物件就非常重要。

若要規劃事件相關性，請執行下列步驟：

1. 決定應用程式中要產生關聯的事件。
2. 決定規則型樣以建立事件相關性。

   規則型樣代表特定的事件相關性狀況，您可以使用它來針對該狀況以某種方式將提出的事件相互關聯。請考慮一下，應用程式處理的事件與「主動式相互關聯技術」規則語言定義的規則型樣之間如何相互關聯。這樣可協助您決定需要使用的規則型樣。

針對事件相關性狀況，請永遠使用最適當的型樣。比方說，如果您想讓規則偵測事件的特定序列，請不要撰寫程式碼來將序列型樣行爲併入過濾規則的規則回應動作中；而是使用序列型樣以建立序列規則。

3. 識別要使用之每個規則型樣的建構。

   下列資訊彙總規則語言中的主要建構，然而每個主要建構的詳細資訊對於規則型樣來說都是唯一的。本資訊的組織方式與透過規則建置器 GUI 來呈現的方式大致相同：

   性質 規則性質的定義，包括規則名稱、說明及型樣。如需詳細資訊，請參閱下列主題：

   - 第 67 頁的『collectionRule 元素』
   - 第 68 頁的『computationRule 元素』
   - 第 74 頁的『duplicateRule 元素』
   - 第 81 頁的『filterRule 元素』
   - 第 94 頁的『sequenceRule 元素』
   - 第 98 頁的『thresholdRule 元素』
   - 第 101 頁的『timerRule 元素』

   變數 規則變數的定義，包括每個變數的名稱、類型、說明，以及起始設定表示式。如需詳細資訊，請參閱第 103 頁的『variable 元素』。

   事件選擇

   決定規則會接受哪些事件來進行處理之準則的定義。如需詳細資訊，請參閱第 78 頁的『eventSelector 元素』。

   分組鍵 分組鍵的定義，分組鍵是針對具有分享共用性質的每個事件群組，指引規則建立個別規則實例 (或其複本) 的方法。如需詳細資訊，請參閱第 82 頁的『groupingKey 元素』。

   型樣明細

   指定有狀態的規則處理比對其型樣的時段，以及定義某些有狀態之規則型樣的唯一特性。如需詳細資訊，請參閱第 102 頁的『timeWindow 元素』。

   若爲計算規則，這包括要套用至已收集事件的計算定義。如需詳細資訊，請參閱第 71 頁的『computeFunction 元素』。

   若爲臨界值規則，這包括臨界值類型的定義，以及臨界值類型特有的其他相關資訊。如需詳細資訊，請參閱下列主題：

   - 第 66 頁的『booleanThreshold 元素』
   - 第 69 頁的『computedThreshold 元素』
   - 第 76 頁的『eventCountThreshold 元素』

   規則回應

   規則完成處理時要採取的動作定義。

   如需詳細資訊，請參閱下列主題：

   - 若爲重複、過濾、序列或臨界值規則，請參閱：第 87 頁的『onDetection 元素』
   - 若爲重複規則，請參閱：第 89 頁的『onNextEvent 元素』
   - 若爲序列或臨界值規則，請參閱：第 89 頁的『onTimeOut 元素』

- 若爲收集、計算、重複或計時器規則，請參閱：第 90 頁的『onTimeWindowComplete 元素』

啓用間隔

規則處於作用中及非作用中的時間定義。如需詳細資訊，請參閱第 61 頁的『activationInterval 元素』。

生命週期

在規則生命週期的下列四個主要階段要採取的動作定義：載入、啓用、停用及卸載。通常，無需定義這些動作。如需詳細資訊，請參閱第 85 頁的『lifeCycleActions 元素』。

4. 指定要在規則表示式中呼叫的 Java 方法及相關片段。規則撰寫者應該使用 Java 方法來呼叫外部模組，而不是在規則表示式中撰寫大量 Java 程式碼。這些外部模組可由內含「主動式相互關聯技術」的應用程式提供，或視需要由規則撰寫者建立。此外，還需指定與每一個 Java 方法相關聯的片段。如需相關資訊，請參閱第 20 頁的『編碼表示式的最佳實務範例』。

繼續進行『設計規則以關聯事件』。

設計規則以關聯事件

在規劃事件相關性之後，您必須設計規則，讓事件相互關聯。

首先，完成第 31 頁的『規劃事件相關性』。

若要設計規則，請執行下列步驟：

1. 設計規則集中之規則及規則區塊的組織。
2. 設計定義不同變數時要使用的層次，例如，在規則集、規則區塊或規則層次定義。
3. 根據規則型樣，設計每個規則的元素。

   此步驟會利用您在規劃階段所識別之每個規則型樣的建構。
4. 設計規則之間的互動作業，特別是轉遞事件以及略過重複事件的規則集處理程序。

   如需詳細資訊，請參閱第 29 頁的『規則集中的事件流程』。
5. 設計、開發及測試您所建立以便在表示式中呼叫的所有 Java 方法及相關片段。

繼續進行『規則建置器入門』。

規則建置器入門

設計讓事件相互關聯的規則之後，您可以使用「主動式相互關聯技術」規則建置器來撰寫規則。

下列步驟是使用規則建置器來撰寫規則的主要步驟。

1. 開啓「Eclipse 工作台」。
2. 在「Eclipse 工作台」中設定視景。
3. 設定「主動式相互關聯技術」的喜好設定，或接受預設喜好設定。
4. 建立用來儲存規則集檔案的專案，或使用現有的專案。
5. 建立規則集檔案，並將它儲存在您選擇的專案中。

6. 在規則集內至少建立一個規則區塊。此外，您還可以在規則集內建立其他規則區塊，並且可以在規則區塊內建立規則區塊。
7. 在規則區塊內建立規則。
8. 驗證規則集。
9. 編譯規則集。
10. 視需要更新規則集。

在規則內的表示式中，您還可以併入「Eclipse 工作台」之「片段」檢視畫面中的片段。

在「Eclipse 工作台」設定視景

在進行任何操作之前，必須先在「Eclipse 工作台」中設定視景。此主題說明如何開啓「規則建置器」視景。

首先，開啓「Eclipse 工作台」(如果尚未開啓)。

雖然可使用「規則建置器」視景之外的視景，但執行各項作業的步驟會視您選擇的視景而稍有不同。

若要開啓「規則建置器」視景，請執行下列步驟：

1. 在「工作台」中，按一下**視窗 → 開啓視景 → 其他...**。
2. 按一下**規則建置器**，再按一下**確定**。 然後會顯示「規則建置器」視景。

繼續進行『設定喜好設定』。

設定喜好設定

建立規則集檔案之前，請確保已在「Eclipse 工作台」正確設定「主動式相互關聯技術」的喜好設定。此主題說明如何設定這些喜好設定。

首先，開啓「Eclipse 工作台」(如果尚未開啓)。

若要設定「主動式相互關聯技術」的喜好設定，請執行下列步驟：

1. 在「工作台」中，按一下**視窗 → 喜好設定...**。
2. 按一下**主動式相互關聯技術**，然後在「主動式相互關聯技術」頁面上，指定您是否要新增 『act』 作爲規則建置器中新建立的規則及規則區塊的字首。 根據預設值，『act』 *不會*作爲字首新增。
3. 展開**主動式相互關聯技術**。 是否會顯示下列其他項目將視您的應用程式而定。按一下這些項目，以設定相關的喜好設定。

項目	相關的喜好設定
公用基礎事件定義支援模組	您可以指定提供一或多個事件類型 (包括指定事件類型可以包含的屬性名稱及類型) 結構的 XML 檔案，其中事件類型會符合「公用基礎事件」規格。然後，在建立規則時便可使用這些事件類型及屬性。

項目	相關的喜好設定
編譯器	您可以指定下列主要項目： • 編譯過的規則集是否需要儲存 • 編譯過的規則集檔案類型 • 在編譯時間要使用之程式碼的類別路徑 根據預設值，編譯過的規則集會儲存並顯示在「導覽器」檢視畫面中，且檔案類型爲 `acts`，這表示此編譯器輸出爲已序列化的規則集。

繼續進行『建立專案以儲存規則集檔案』。

建立專案以儲存規則集檔案

在「Eclipse 工作台」中，建立規則集檔案時，必須指定要儲存檔案的專案。因此，您必須在建立規則集檔案之前建立專案，或是使用現有的專案。此主題說明在「Eclipse 工作台」中建立專案的方法。

首先，開啓「Eclipse 工作台」(如果尚未開啓)。同時，也開啓「規則建置器」視景。

若要在「工作台」中建立簡式專案，請執行下列步驟：

1. 按一下**檔案 → 開新檔案 → 專案...**。
2. 在「新增專案」精靈中，按一下**簡式 → 專案**，再按**下一步**。
3. 在**專案名稱**欄位中，鍵入專案的唯一名稱。另外，可以使用專案的預設位置，也可以選擇其他位置。
4. 按一下**完成**。 然後，新專案便會顯示在「規則建置器」視景的「導覽器」檢視畫面中。

繼續進行『建立規則集』。

建立規則集

此主題說明如何建立規則集。

首先，開啓「Eclipse 工作台」(如果尚未開啓)。同時，也開啓「規則建置器」視景。

若要建立規則集檔案，請執行下列步驟：

1. 按一下**檔案 → 開新檔案 → 規則集檔案**。
2. 在「新增規則集檔案」精靈中，按一下與要儲存規則集檔案之專案相關聯的資料夾名稱。這很可能就是您在『建立專案以儲存規則集檔案』中建立的專案。 然後，資料夾名稱會顯示在第一個欄位中。
3. 在**檔案名稱**欄位中，鍵入規則集檔案的名稱。 檔案類型必須爲 actl。
4. 按**下一步**。
5. 鍵入規則集的唯一名稱及選擇性說明。如果不想使用 **XML 編碼**欄位中的預設值，也可以指定規則集檔案 (XML 檔案) 的編碼樣式。
6. 按一下**完成**。 然後，在「導覽器」檢視畫面中，規則集檔案便會顯示在其專案資料夾中。儲存檔案時會對其進行驗證，因此檔案旁邊會顯示『已驗證』字樣。規則集檔案也會顯示在「概要」檢視畫面中。

繼續進行『建立規則區塊』。

建立規則區塊

此主題說明如何在規則集或其他規則區塊中建立規則區塊。

首先，開啓「Eclipse 工作台」(如果尚未開啓)。同時，也開啓「規則建置器」視景。

如果您是第一次建立規則集，則必須先在規則集中建立至少一個規則區塊，才能建立規則。建立第一個規則區塊之後，就可以建立其他規則區塊，包括規則區塊中的規則區塊。

若要建立規則區塊，請執行下列步驟：

1. 在「導覽器」檢視畫面中，按兩下要更新的規則集檔名。 便會在「概要」檢視畫面中開啓檔案。當您在「概要」檢視畫面中按一下規則集時，該規則集的目前資訊便會顯示在編輯區中。
2. 在「概要」檢視畫面中，以滑鼠右鍵按一下規則集。
3. 按一下**新增子項 → 規則區塊**。 然後，規則區塊便會顯示在「概要」檢視畫面的規則集中，且規則區塊目前的資訊也會顯示在編輯區中。

現在，您可以使用下列方法，建立其他規則區塊：

- 若要建立現有規則區塊的對等規則區塊，請以滑鼠右鍵按一下現有的規則區塊，再按一下**新增同層級項 → 規則區塊**。
- 若要在現有的規則區塊中建立規則區塊，請以滑鼠右鍵按一下現有的規則區塊，再按一下**新增子項 → 規則區塊**。

另外，還可以使用編輯器，更新規則集及每個規則區塊的資訊。繼續進行『建立規則』。

建立規則

此主題說明如何建立規則。

首先，開啓「Eclipse 工作台」(如果尚未開啓)。同時，也開啓「規則建置器」視景。

您必須在規則區塊中建立規則。若要建立規則，請執行下列步驟：

1. 在「概要」檢視畫面中，以滑鼠右鍵按一下要更新的規則區塊。
2. 按一下**新增子項**，以及要建立的規則類型。 然後，規則便會顯示在「概要」檢視畫面的規則區塊中，且規則目前的資訊也會顯示在編輯區中。

您可以使用同樣的方法，將其他規則新增至規則區塊。另外，還可以使用編輯器，更新每個規則的資訊。繼續進行『驗證規則集』。

驗證規則集

此主題說明如何在編譯規則集之前對其進行驗證。

首先，開啓「Eclipse 工作台」(如果尚未開啓)。同時，也開啓「規則建置器」視景。

若要驗證規則集，請執行下列步驟：

1. 在「概要」檢視畫面中，以滑鼠右鍵按一下規則、規則區塊或規則集。

2. 按一下**驗證規則集**。 當驗證完成時會顯示訊息視窗，指出是否有發生問題。如果順利完成驗證，則在「導覽器」檢視畫面中，規則集檔名的右邊會顯示『已驗證』字樣。

當驗證順利完成時，請繼續執行『編譯規則集』。

編譯規則集

此主題說明如何編譯規則集。

首先，開啓「Eclipse 工作台」(如果尚未開啓)。同時，也開啓「規則建置器」視景。

在「導覽器」或「概要」檢視畫面中，以滑鼠右鍵按一下要編譯的規則集，再按一下**編譯規則集**。所有的編譯錯誤都會顯示在「問題」檢視畫面中。

根據預設值，如果沒有編譯錯誤，則編譯過的規則集會顯示在「導覽器」檢視畫面中，且預設檔案類型爲 acts，這表示此爲已序列化的規則集。同時，在「導覽器」檢視畫面中，規則集檔名的右邊會顯示『已編譯』字樣。

更新規則集

此主題說明如何更新規則集。

首先，開啓「Eclipse 工作台」(如果尚未開啓)。同時，也開啓「規則建置器」視景。

在「概要」檢視畫面中，按一下您要更新的項目 (規則、規則區塊或規則集)。項目的現行資訊便會顯示在編輯區，您可以使用編輯器來更新此資訊。

若要建立與現有規則區塊或規則對等的規則區塊或規則，請執行下列步驟：

1. 以滑鼠右鍵按一下現有的規則區塊或規則。
2. 按一下**新增同層級項**及您要新增的項目 (規則區塊或規則類型)。

若要在現有的規則區塊內建立規則區塊或規則，請執行下列步驟：

1. 以滑鼠右鍵按一下現有的規則區塊。
2. 按一下**新增子項**及您要新增的項目 (規則區塊或規則類型)。

若要存取「概要」檢視畫面中的其他更新功能，請執行下列步驟：

1. 以滑鼠右鍵按一下您要更新的項目 (規則區塊或規則)。
2. 按一下功能的功能表項目，例如**刪除**、**複製**或**貼上**。

在規則內的表示式中併入片段

在規則內的表示式中，您可以併入「Eclipse 工作台」之「片段」檢視畫面中的片段。

首先，開啓「Eclipse 工作台」(如果尚未開啓)。同時，也開啓「規則建置器」視景。

「片段」檢視畫面會顯示在「規則建置器」視景中。依據其功能，片段分爲幾個種類。

若要併入「片段」檢視畫面中的片段，請執行下列步驟：

1. 按一下片段種類以檢視種類中的片段名稱。
2. 按一下您想要在表示式中併入的片段。

3. 將片段拖曳至各自的**表示式**欄位。 程式碼會置於**表示式**欄位中的游標所在位置。如果程式碼需要規則撰寫者輸入內容 (例如特定訊息文字或變數值)，則會提示您在表示式中併入程式碼之前提供該輸入內容。

繼續進行第 36 頁的『驗證規則集』。

第 2 篇 規則撰寫者參照

第 4 章 規則集組織摘要

此參照列出規則集、規則區塊，以及每種規則類型的所有語言元素。您可以將它視爲編碼規則集的簡易說明。

表 10會解釋每個語言元素後面的表示法具有何種意義。*n* 代表無界限數字。

表 10. 定義語言元素出現次數的表示法說明

表示法	意義
(0, 1)	0 表示語言元素是選用的。1 表示編碼時只容許出現 1 次。
(0, *n*)	0 表示語言元素是選用的。*n* 表示編碼時容許出現多次。
(1, 1)	第一個 1 表示語言元素是必要的。第二個 1 表示只容許出現 1 次。
(1, *n*)	1 表示語言元素是必要的。*n* 表示容許出現多次。
(2, *n*)	2 表示語言元素必須出現 2 次。*n* 表示容許出現更多次。

元素必須按照所顯示的次序來編碼。如果元素是選用的，則無需進行編碼，但所有編碼的元素都必須遵循正確的次序。

規則集摘要

此摘要列出規則集的語言元素。

規則集元素

<ruleSet> 包含下列元素：

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

相關參考

『規則區塊摘要』
此摘要列出規則區塊的語言元素。

規則區塊摘要

此摘要列出規則區塊的語言元素。

規則區塊元素

<ruleBlock> 包含下列元素。

如果編碼這些元素，則必須依照顯示的次序來編碼 <comment>、<import> 及 <variable> 元素。剩餘的元素則可以依照任何次序來編碼。

- <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)

相關參考

『收集規則摘要』
此摘要列出收集規則的所有語言元素。

第 44 頁的『計算規則摘要』
此摘要列出計算規則的所有語言元素。

第 45 頁的『重複規則摘要』
此摘要列出重複規則的所有語言元素。

第 47 頁的『過濾規則摘要』
此摘要列出過濾規則的所有語言元素。

第 48 頁的『序列規則摘要』
此摘要列出序列規則的所有語言元素。

第 49 頁的『臨界值規則摘要』
此摘要列出臨界值規則的所有語言元素。

第 51 頁的『計時器規則摘要』
此摘要列出計時器規則的所有語言元素。

收集規則摘要

此摘要列出收集規則的所有語言元素。

規則元素

<collectionRule> 包含下列元素：

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

        - 下列三個元素的其中一個 (1, 1)：
          - <dateTime>
          - <whenLoaded>
          - <inactiveWhenLoaded>
      - <stop> (0, 1)
        - 下列三個元素的其中一個 (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)
  - 下列三個元素，依任何次序 (1, n)：
    - <attributeAlias>
      - <eventAttribute> (2, n)
    - <attributeName>

- <computedValue>

- <timeWindow> (1, 1)
  - 下列兩個元素的其中一個 (1, 1)：
    - <timeInterval>
    - <runUntilDeactivated>
- <onTimeWindowComplete> (0, 1)
  - <action> (0, n)

計算規則摘要

此摘要列出計算規則的所有語言元素。

規則元素

<computationRule> 包含下列元素：

- <comment> (0, 1)
- <variable> (0, n)
  - <comment> (0, 1)
  - <varInitializer> (1, 1)
- <activationInterval> (0, 1)
  - <activationTime> (0, 1)
    - <start> (0, 1)
      - 下列三個元素的其中一個 (1, 1)：
        - <dateTime>
        - <whenLoaded>
        - <inactiveWhenLoaded>
    - <stop> (0, 1)
      - 下列三個元素的其中一個 (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)
  - 下列三個元素，依任何次序 (1, n)：
    - <attributeAlias>
      - <eventAttribute> (2, n)
    - <attributeName>
    - <computedValue>
- <computeFunction> (1, 1)
- <timeWindow> (1, 1)
  - 下列兩個元素的其中一個 (1, 1)：
    - <timeInterval>
    - <runUntilDeactivated>
- <onTimeWindowComplete> (0, 1)
  - <action> (0, n)

重複規則摘要

此摘要列出重複規則的所有語言元素。

規則元素

<duplicateRule> 包含下列元素：

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

- <start> (0, 1)
  - 下列三個元素的其中一個 (1, 1)：
    - <dateTime>
    - <whenLoaded>
    - <inactiveWhenLoaded>
- <stop> (0, 1)
  - 下列三個元素的其中一個 (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)
– 下列三個元素，依任何次序 (1, n)：
- <attributeAlias>
  - <eventAttribute> (2, n)

- <attributeName>
- <computedValue>

- <timeWindow> (1, 1)
  - 下列兩個元素的其中一個 (1, 1)：
    - <timeInterval>
    - <runUntilDeactivated>
- <onDetection> (0, 1)
  - <action> (0, n)
- <onNextEvent> (0, 1)
  - <action> (0, n)
- <onTimeWindowComplete> (0, 1)
  - <action> (0, n)

過濾規則摘要

此摘要列出過濾規則的所有語言元素。

規則元素

<filterRule> 包含下列元素：

- <comment> (0, 1)
- <variable> (0, n)
  - <comment> (0, 1)
  - <varInitializer> (1, 1)
- <activationInterval> (0, 1)
  - <activationTime> (0, 1)
    - <start> (0, 1)
      - 下列三個元素的其中一個 (1, 1)：
        - <dateTime>
        - <whenLoaded>
        - <inactiveWhenLoaded>
    - <stop> (0, 1)
      - 下列三個元素的其中一個 (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)

序列規則摘要

此摘要列出序列規則的所有語言元素。

規則元素

<sequenceRule> 包含下列元素：

- <comment> (0, 1)
- <variable> (0, n)
  - <comment> (0, 1)
  - <varInitializer> (1, 1)
- <activationInterval> (0, 1)
  - <activationTime> (0, 1)
    - <start> (0, 1)
      - 下列三個元素的其中一個 (1, 1)：
        - <dateTime>
        - <whenLoaded>
        - <inactiveWhenLoaded>
    - <stop> (0, 1)
      - 下列三個元素的其中一個 (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)
    – 下列三個元素，依任何次序 (1, n)：
      - <attributeAlias>
        • <eventAttribute> (2, n)
      - <attributeName>
      - <computedValue>
  • <timeWindow> (1, 1)
    – 下列兩個元素的其中一個 (1, 1)：
      - <timeInterval>
      - <runUntilDeactivated>
  • <onDetection> (0, 1)
    – <action> (0, n)
  • <onTimeOut> (0, 1)
    – <action> (0, n)

臨界值規則摘要

此摘要列出臨界值規則的所有語言元素。

規則元素

<thresholdRule> 包含下列元素：

- <comment> (0, 1)
- <variable> (0, n)
  - <comment> (0, 1)
  - <varInitializer> (1, 1)
- <activationInterval> (0, 1)
  - <activationTime> (0, 1)
    - <start> (0, 1)
      - 下列三個元素的其中一個 (1, 1)：
        - <dateTime>
        - <whenLoaded>
        - <inactiveWhenLoaded>
    - <stop> (0, 1)
      - 下列三個元素的其中一個 (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)
  - 下列三個元素，依任何次序 (1, n)：
    - <attributeAlias>
      - <eventAttribute> (2, n)
    - <attributeName>
    - <computedValue>
- 下列三個元素的其中一個 (1, 1)：
  - <booleanThreshold>
  - <computedThreshold>
  - <eventCountThreshold>
- <timeWindow> (1, 1)
  - 下列兩個元素的其中一個 (1, 1)：
    - <timeInterval>
    - <runUntilDeactivated>
- <onDetection> (0, 1)
  - <action> (0, n)
- <onTimeOut> (0, 1)
  - <action> (0, n)

計時器規則摘要

此摘要列出計時器規則的所有語言元素。

規則元素

<timerRule> 包含下列元素：

- <comment> (0, 1)
- <variable> (0, n)
  - <comment> (0, 1)
  - <varInitializer> (1, 1)
- <activationInterval> (0, 1)
  - <activationTime> (0, 1)
    - <start> (0, 1)
      - 下列三個元素的其中一個 (1, 1)：
        - <dateTime>
        - <whenLoaded>
        - <inactiveWhenLoaded>

- <stop> (0, 1)
  - 下列三個元素的其中一個 (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)
– 下列兩個元素的其中一個 (1, 1)：
- <timeInterval>
- <runUntilDeactivated>

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

第 5 章 語言元素參照

此參照說明「主動式相互關聯技術」規則語言之 XML 綱目中語言元素的詳細資訊。語言元素會按字母順序列出，如需每個元素可以使用之屬性的說明，請參閱該元素的主題。

在 XML 及其他標記語言 (例如，SGML 及 HTML) 中，元素是由下述項目組成的基本單位：起始標示、結束標示、相關屬性及其值，以及起始與結束標示之間包含的任何文字。屬性是名稱值組，您可以在元素上編碼屬性，以定義元素的某種性質。屬性具有資料類型，可識別其值中提供的資訊類型 (例如，數字、文字或布林資訊)。

在 XML 中，名稱空間是制式資源識別字 (URI)，它提供唯一的名稱，以與綱目中的元素及類型定義產生關聯。URI 會指出包含元素定義的 XML 綱目。名稱空間是以字首字串及其後面的冒號來指定。「主動式相互關聯技術」規則語言綱目是在三個不同的檔案中定義，並且使用下列三個名稱空間：

xsd: 此名稱空間指出語言元素是使用標準 XML 綱目定義的，如需其說明，請造訪 http://www.w3.org。

br: 此名稱空間指出語言元素是使用「主動式相互關聯技術」基本規則集綱目定義的，它位於 com/ibm/correlation/ruleparser/xml/RuleSetBase.xsd 子目錄中的 ACTparser.jar 檔案。例如，`br:ruleSet` 是指在 RuleSetBase.xsd 檔案中定義的 ruleSet 元素。

act: 此名稱空間指出語言元素是使用「主動式相互關聯技術」語言綱目定義的，它位於 com/ibm/correlation/ruleparser/xml/ACTL.xsd 子目錄中的 ACTparser.jar 檔案。例如，`act:ruleSet` 是指在 ACTL.xsd 檔案中定義的 ruleSet 元素。

在規則語言綱目中，會將語言元素定義爲元素或複式類型，例如：

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

在綱目中，minOccurs 及 maxOccurs 屬性會分別定義語言元素的最小及最大出現次數。表 11說明 minOccurs 及 maxOccurs 屬性不同值的意義。

表 11. 綱目中定義語言元素出現次數的屬性

屬性	屬性值	意義
minOccurs	0	語言元素是選用的。
minOccurs	1	語言元素必須至少出現一次。1 是 minOccurs 屬性的預設值。
minOccurs	2	語言元素必須至少出現兩次。
maxOccurs	1	語言元素不能出現多次。1 是 maxOccurs 屬性的預設值。
maxOccurs	unbounded	語言元素可以出現任意次數。

動作元素

<action> 元素包含定義規則回應動作或生命週期動作的表示式。

詳細資訊

如需可在表示式中使用之變數的相關資訊，請參閱第 21 頁的『變數』。某些變數的使用取決於表示式的環境定義。

屬性

<action> 具有下列屬性：

表 12. <action> 元素的屬性

名稱	說明	資料類型	必要的嗎？
expressionLanguage	識別用來撰寫表示式的程式設計語言。因爲 Java 程式設計語言是唯一受支援的表示式語言 ，所以此屬性唯一的有效值是 `java`。	xsd:NMTOKEN	是
名稱	識別動作。在進行疑難排解時，此 ID 非常有用，尤其當它是針對特定的規則回應或生命週期動作來定義的所有 <action> 元素中的唯一名稱時。	xsd:NMTOKEN	否

包含範圍

<action> 內含在下列元素中：

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

包含

<action> 不包含任何元素。

相關概念

第 17 頁的『表示式』
表示式是包含自訂邏輯的程式碼，您可以將這些自訂邏輯新增至規則。表示式還可以存取「主動式相互關聯技術」引擎的外部程式碼。在規則語言中， 表示式僅在特定環境定義或規則語言元素內才有效。

activateOnEvent 元素

<activateOnEvent> 元素會定義可以啓用規則或 (若爲使用 <groupingKey> 元素來定義的規則) 規則實例的事件。

下列是選取事件的三種可能方法：

- 使用一或多個具有 <filteringPredicate> 元素的 <eventType> 元素

- 使用一或多個不具有 <filteringPredicate> 元素的 <eventType> 元素
- 使用不具有任何 <eventType> 元素的 <filteringPredicate> 元素

如果規則處於不作用狀態，而且未編碼任何 <eventType> 或 <filteringPredicate> 元素，則會選取發生的任何事件。

不編碼任何 <eventType> 元素會對系統效能產生負面影響。

假設您要選取類型爲 `Audit Failure` 的所有事件。您可以使用過濾述語來進一步修正選取準則，僅包括其事件屬性具有固定值的事件。例如，您可以編碼 <eventType> 元素，以選取類型爲 `Audit Failure` 的所有事件，並編碼 <filteringPredicate> 元素，僅選取其 hostname 屬性值爲 `MyCriticalSystem` 的那些事件。

屬性

<activateOnEvent> 沒有屬性。

包含範圍

<activateOnEvent> 內含在下列元素中：

- <activationInterval>
- <activationByGroupingKey>

包含

<activateOnEvent> 包含下列元素。

元素必須按照所顯示的次序來編碼。如果元素是選用的，則無需進行編碼，但所有編碼的元素都必須遵循正確的次序。

表 13. <activateOnEvent> 元素中包含的元素

元素	必要的或選用的？
<eventType>	選用的。出現次數 0 或更多是可接受的。
<filteringPredicate>	選用的。出現次數 0 或 1 是可接受的。
<stopAfter>	只有當 <activateOnEvent> 元素包含在 <activationByGroupingKey> 元素中時，此元素才有效。 選用的。出現次數 0 或 1 是可接受的。

activationByGroupingKey 元素

<activationByGroupingKey> 元素包含的元素可以指定事件來啓用及停用 <groupingKey> 元素所定義的規則實例。因爲 <groupingKey> 元素對於過濾及計時器規則無效，所以 <activationByGroupingKey> 元素不會套用至這些規則。

詳細資訊

<activationByGroupingKey> 元素提供的功能是在您定義分組鍵的規則中使用。它可讓您根據分組鍵來控制規則實例的啓用及停用。編碼 <activationByGroupingKey> 元素時，可以根據 <activationByGroupingKey> 內的 <activateOnEvent> 及 <deactivateOnEvent> 條件個別啓用及停用每一個規則實例。

下列範例說明計算規則內之 <activationByGroupingKey> 元素的使用。

- 下列計算規則接受 StockSharesTraded 類型的事件。這些事件指出許多不同公司進行交易的股份數量。
- 分組鍵是事件的 stockSymbol 屬性。stockSymbol 屬性的值是特定公司的名稱。
- 事件之 sharesTraded 屬性的值是各個公司已完成交易的股份數量 (以 stockSymbol 屬性的值來指出該公司)。
- 會爲特定公司建立報表，指出該公司每 10 分鐘交易的股份數量。然而，計算規則必須先接收到 StartReporting 類型的事件 (以各公司的名稱作爲 stockSymbol 屬性的值)，才能建立此報表。

```
<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>
```

屬性

<activationByGroupingKey> 沒有屬性。

包含範圍

<activationByGroupingKey> 內含在下列元素中：

- <activationInterval>

包含

<activationByGroupingKey> 包含下列元素。

元素必須按照所顯示的次序來編碼。如果元素是選用的，則無需進行編碼，但所有編碼的元素都必須遵循正確的次序。

表 14. <activationByGroupingKey> 元素中包含的元素

元素	必要的或選用的？
<activateOnEvent>	選用的。出現次數 0 或 1 是可接受的。
<deactivateOnEvent>	選用的。出現次數 0 或 1 是可接受的。

<activationInterval> 及 <activationByGroupingKey> 元素之間的關係

<activateOnEvent> 及 <deactivateOnEvent> 元素包含在這兩個元素中：

- <activationInterval>
- <activationByGroupingKey>，它也包含在 <activationInterval> 中

根據目前的規則活動，以及 <activationInterval> 及 <activationByGroupingKey> 元素中之 <activateOnEvent> 與 <deactivateOnEvent> 定義之間的互動，規則行爲可能有所不同。下列範例說明這些定義的互動方式。

在此範例中，會定義重複的規則，以抑制維護模式下的系統中的事件，並在維護期間結束時，提供抑制事件數量的摘要報告。

依預設，其中定義了分組鍵的規則容許處理所有分組鍵值。因此，當事件符合規則的事件選擇準則時，所有的規則實例都處於作用中狀態，並且可以根據分組鍵的任何值來接受這些事件。規則的啓用間隔與規則不具有分組鍵時的情況相同 (因爲在實際情況下，會處理所有符合規則之事件選擇準則的事件)。

在下列範例中，分組鍵爲 `hostname`，且 <activationInterval> 元素中的定義會指定下列動作：

1. 接收到 `StartMaintenanceModeAllHosts` 類型的事件時，啓用所有規則實例。
2. 在 2 小時之後或接收到 `StopMaintenanceModeAllHosts` 類型的事件時，停用所有規則實例。

```
<duplicateRule name="Maintenance_Supression">
   <activationInterval>
      <activationTime>
         <start>
            <inactiveWhenLoaded/>
         </start>
      <stop>
        <after duration="PT2H" unit="ISO-8601"/>
      </stop>
      </activationTime>
    <activateOnEvent>
      <eventType type="StartMaintenanceModeAllHosts"/>
    </activateOnEvent>
    <deactivateOnEvent>
      <eventType type="StopMaintenanceModeAllHosts"/>
```

```
    </deactivateOnEvent>
   </activationInterval>
  <groupingKey missingAttributeHandling="ignoreEvent">
    <attributeName>hostname</attributeName>
   </groupingKey>
   <timeWindow>
    <runUntilDeactivated/>
   </timeWindow>
  <onDetection>
    <action expressionLanguage="java" name="DropEvent">
      <![CDATA[act_lib.exitRuleSet();]]>
      </action>
  </onDetection>   <onTimeWindowComplete>
    <action expressionLanguage="java" name="CreateSummaryOfSupressedEvents">
      <![CDATA[Helper.createSummaryEvent("MaintenanceSummary", act_eventList, act_lib);]]>
      </action>
   </onTimeWindowComplete>
</duplicateRule>
```

在某些狀況下，您可能要控制變成作用中狀態的規則實例，以及它們變成作用中狀態的時間。對於這些狀況，您應該編碼 <activationByGroupingKey> 元素。

下列範例是先前範例的延伸，說明如何使用分組鍵值來選取容許處理的規則實例。<activationByGroupingKey> 元素中的定義指定下列動作：

1. 容許在特定主機名稱接收到 StartMaintenanceMode 類型的事件時，處理該主機名稱的規則實例。
2. 在啓用 2 小時之後或各主機名稱接收到 StopMaintenanceMode 類型的事件時，停用這些規則實例。

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

下列陳述式彙總編碼 <activationByGroupingKey> 元素期間發生的事件：

- 在 <activationByGroupingKey> 元素中編碼 <activateOnEvent> 元素時，只能處理共用符合 <activationByGroupingKey> <activateOnEvent> 條件之事件分組鍵值的事件。
- 在 <activationByGroupingKey> 元素中編碼 <deactivateOnEvent> 元素時，不 容許處理共用符合 <activationByGroupingKey> <deactivateOnEvent> 條件之事件分組鍵值的事件。

不同的啓用及停用定義對規則狀態的影響

第 59 頁的表 15及第 60 頁的表 16顯示不同的啓用及停用定義如何影響規則的狀態。在這些表中使用下列慣例：

- *A* 是正在啓用的事件。
- 在 『*A[x]*』 表示法中，*x* 代表分組鍵值。例如，*A[1]* 是正在啓用的事件，其分組鍵值爲 1。
- *D* 是正在停用的事件。
- 在 『*D[x]*』 表示法中，*x* 代表分組鍵值。例如，*D[1]* 是正在停用的事件，其分組鍵值爲 1。

表 15. 規則狀態會根據不同的啓用定義而變更

起始規則狀態	規則狀態會受下列潛在因素影響	結束規則狀態
不作用	<activationInterval> <activationTime> <start> 內定義的時間	1. 已啓用規則。 2. 執行 <onActivation> 動作。 3. 接受所有分組鍵值。
	activate() 方法	
	事件 *A*，在 <activationInterval> <activateOnEvent> 內定義	
	事件 *A[1]*，在 <activationByGroupingKey> <activateOnEvent> 內定義 (*無* <stopAfter>)	1. 已啓用規則。 2. 執行 <onActivation> 動作。 3. 只接受值爲 1 的分組鍵值。當規則型樣符合此規則實例時，不再接受值爲 1 的分組鍵值。
	事件 *A[2]*，在 <activateOnEvent> 內定義 (*具有* <stopAfter>)	1. 已啓用規則。 2. 執行 <onActivation> 動作。 3. 只接受值爲 2 的分組鍵值，而且它只能用於 <stopAfter> 元素所指定的期間。此規則實例的規則型樣可在此期間內多次符合。
• 作用中 • 接受所有分組鍵值	<activationInterval> <activationTime> <start> 內定義的時間	規則狀態未發生任何變更。與起始規則狀態相同。
	activate() 方法	
	事件 *A*，在 <activationInterval> <activateOnEvent> 內定義	
	事件 *A[1]*，在 <activationByGroupingKey> <activateOnEvent> 內定義 (*無* <stopAfter>)	
	事件 *A[2]*，在 <activateOnEvent> 內定義 (*具有* <stopAfter>)	
• 作用中 • 只接受依據 <activationByGroupingKey> <activateOnEvent> 定義觸發規則實例的分組鍵值	<activationInterval> <activationTime> <start> 內定義的時間	規則狀態未發生任何變更。與起始規則狀態相同。
	activate() 方法	
	事件 *A*，在 <activationInterval> <activateOnEvent> 內定義	接受所有分組鍵值。
	事件 *A[1]*，在 <activationByGroupingKey> <activateOnEvent> 內定義 (*無* <stopAfter>)	• 除了先前接受的分組鍵值，現在也接受值爲 1 的分組鍵值。 • 當規則型樣符合此規則實例時，不再接受值爲 1 的分組鍵值。
	事件 *A[2]*，在 <activateOnEvent> 內定義 (*具有* <stopAfter>)	• 除了先前接受的分組鍵值，現在也接受值爲 2 的分組鍵值。 • 此值只能用於 <stopAfter> 元素所指定的期間。 • 此規則實例的規則型樣可在此期間內多次符合。

表 15. 規則狀態會根據不同的啓用定義而變更 (繼續)

起始規則狀態	規則狀態會受下列潛在因素影響	結束規則狀態
• 作用中 • 接受所有的分組鍵值 (除了基於 <activationByGroupingKey> <deactivateOnEvent> 定義而不接受的值之外)	<activationInterval> <activationTime> <start> 內定義的時間	規則狀態未發生任何變更。與起始規則狀態相同。
	activate() 方法	
	事件 *A*，在 <activationInterval> <activateOnEvent> 內定義	接受所有分組鍵值。
	事件 *A[1]*，在 <activationByGroupingKey> <activateOnEvent> 內定義 (*無* <stopAfter>)	除了先前接受的分組鍵值，現在也接受值爲 1 的分組鍵值。
	事件 *A[2]*，在 <activateOnEvent> 內定義 (*具有* <stopAfter>)	除了先前接受的分組鍵值，現在也接受值爲 2 的分組鍵值。

表 16. 規則狀態會根據不同的停用定義而變更

起始規則狀態	規則狀態會受下列潛在因素影響	結束規則狀態
不作用	<activationInterval> <activationTime> <stop> 內定義的時間	規則狀態未發生任何變更。與起始規則狀態相同。
	deactivate() 方法	
	事件 *D*，在 <activationInterval> <deactivateOnEvent> 內定義	
	事件 *D[1]*，在 <activationByGroupingKey> <deactivateOnEvent> 內定義	
	在 <activationByGroupingKey> <activateOnEvent> <stopAfter> 內定義的期間遇到事件 *A[2]* 時結束	
• 作用中 • 接受所有分組鍵值	<activationInterval> <activationTime> <stop> 內定義的時間	1. 已停用所有規則實例。 2. 執行 <onDeactivation> 動作。 3. 已停用規則。
	deactivate() 方法	
	事件 *D*，在 <activationInterval> <deactivateOnEvent> 內定義	
	事件 *D[1]*，在 <activationByGroupingKey> <deactivateOnEvent> 內定義	• 不再接受值爲 1 的分組鍵值。 • 如果分組鍵值爲 1 的規則實例處於作用中狀態，則會停用它。
	在 <activationByGroupingKey> <activateOnEvent> <stopAfter> 內定義的期間遇到事件 *A[2]* 時結束	已停用分組鍵值爲 2 的規則實例。

表 16. 規則狀態會根據不同的停用定義而變更 (繼續)

起始規則狀態	規則狀態會受下列潛在因素影響	結束規則狀態
• 作用中 • 只接受依據 <activationByGroupingKey> <activateOnEvent> 定義觸發規則實例的分組鍵值	<activationInterval> <activationTime> <stop> 內定義的時間	1. 已停用所有規則實例。 2. 執行 <onDeactivation> 動作。 3. 已停用規則。
	deactivate() 方法	
	事件 *D*，在 <activationInterval> <deactivateOnEvent> 內定義	
	事件 *D[1]*，在 <activationByGroupingKey> <deactivateOnEvent> 內定義	• 不再接受值爲 1 的分組鍵值。 • 如果分組鍵值爲 1 的規則實例處於作用中狀態，則會停用它。
	在 <activationByGroupingKey> <activateOnEvent> <stopAfter> 內定義的期間遇到事件 *A[2]* 時結束	• 不再接受值爲 2 的分組鍵值。 • 已停用分組鍵值爲 2 的規則實例。
• 作用中 • 接受所有的分組鍵值 (除了基於 <activationByGroupingKey> <deactivateOnEvent> 定義而不接受的值之外)	<activationInterval> <activationTime> <stop> 內定義的時間	1. 已停用所有規則實例。 2. 執行 <onDeactivation> 動作。 3. 已停用規則。
	deactivate() 方法	
	事件 *D*，在 <activationInterval> <deactivateOnEvent> 內定義	
	事件 *D[1]*，在 <activationByGroupingKey> <deactivateOnEvent> 內定義	• 不再接受值爲 1 的分組鍵值。 • 如果分組鍵值爲 1 的規則實例處於作用中狀態，則會停用它。
	在 <activationByGroupingKey> <activateOnEvent> <stopAfter> 內定義的期間遇到事件 *A[2]* 時結束	已停用分組鍵值爲 2 的規則實例。

activationInterval 元素

<activationInterval> 元素包含的元素會定義規則何時處於作用中及不作用中狀態。

詳細資訊

規則可以在離散的時間點上或者依特定事件啓用或停用。

如果您指定規則在離散的時間點上*且* 依特定事件啓用或停用，則會根據最先發生的是時間點或事件的接收，來啓用或停用該規則。然而，在此情況下，規則在其整個生命週期中可能會被多個事件啓用或停用。例如，規則可能依事件啓用、停用、在定義的時間點上啓用、再停用，然後依其他事件啓用。

在企業環境中，您可能要在接收到指示企業股票交換已開始的事件時啓用規則。在 IT 環境中，您可能要在 2005 年 10 月 29 日 06:00 啓動維護視窗，並根據下列最先發生的狀況來結束它：

- 2005 年 10 月 30 日 11:30
- 接收到指示維護工作已完成的事件時

屬性

<activationInterval> 沒有屬性。

包含範圍

<activationInterval> 內含在下列元素中：

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

包含

<activationInterval> 包含下列元素。

元素必須按照所顯示的次序來編碼。如果元素是選用的，則無需進行編碼，但所有編碼的元素都必須遵循正確的次序。

表 17. <activationInterval> 元素中包含的元素

元素	必要的或選用的？
<activationTime>	選用的。出現次數 0 或 1 是可接受的。
<activateOnEvent>	選用的。出現次數 0 或 1 是可接受的。
<deactivateOnEvent>	選用的。出現次數 0 或 1 是可接受的。
<activationByGroupingKey>	選用的。出現次數 0 或 1 是可接受的。

內含元素之間的關係

<activationTime> 元素中包含的 <start> 及 <stop> 元素是啓用及停用規則的靜態方法。透過這些元素，可在離散的時間點上啓用或停用規則。相反地，<activateOnEvent> 及 <deactivateOnEvent> 元素是啓用及停用規則的動態方法。透過這些元素，可在發生某些事件時啓用或停用規則。例如，如果規則尚未處於作用中，則符合爲 <activateOnEvent> 元素所定義之準則的任何事件都會啓用該規則。如果規則尚未處於不作用狀態，則符合爲 <deactivateOnEvent> 元素所定義之準則的任何事件都會停用該規則。因此，某些事件可以變更何時啓用或停用規則的靜態定義。

第 63 頁的表 18說明如何以及何時根據下列元素的某些編碼組合方式，啓用或停用規則：

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

在第 63 頁的表 18中，*X* 代表啓用規則的事件名稱，而 *Y* 代表停用規則的事件名稱。

如果 <start> 元素完全未編碼，預設開始時間便與 <whenLoaded> 元素所定義的時間相同。

如果 <stop> 元素完全未編碼，預設停止時間便與 <never> 元素所定義的時間相同。

表 18. 基於 <activationInterval> 內含元素之不同編碼組合的規則活動

<activationTime>		<activateOnEvent>	<deactivateOnEvent>	規則活動
<start>	<stop>			
<whenLoaded>	<never>			規則載入時處於作用中狀態，而且會在「主動式相互關聯技術」引擎正在執行時保留作用中狀態。
<whenLoaded>	<never>		*Y*	規則載入時處於作用中狀態。事件 *Y* 會停用該規則。
<whenLoaded>	<never>	*X*	*Y*	規則載入時處於作用中狀態。事件 *Y* 會停用該規則，而事件 *X* 會重新啓用它。停用及重新啓用可以發生多次。
<whenLoaded>	<after>			規則載入時處於作用中狀態，而且會在指定的時間間隔之後停用。
<whenLoaded>	<dateTime>			規則載入時處於作用中狀態，而且會在指定的日期及時間停用。
<inactiveWhenLoaded>	<never>	*X*		規則載入時處於不作用狀態。事件 *X* 會啓用該規則，而且會在「主動式相互關聯技術」引擎正在執行時保留作用中狀態。
<inactiveWhenLoaded>	<never>	*X*	*Y*	規則載入時處於不作用狀態。事件 *X* 會啓用該規則，而事件 *Y* 會停用它。啓用及停用可以發生多次。
<dateTime>	<dateTime>			規則是在指定的日期及時間啓用，並在指定的日期及時間停用。
<dateTime>	<dateTime>	*X*	*Y*	規則是在指定的日期及時間啓用，並在指定的日期及時間停用。事件 *X* 會啓用該規則，而事件 *Y* 會停用它。事件 *X* 及 *Y* 可以多次啓用及停用該規則。
<dateTime>	<never>			規則是在指定的日期及時間啓用，而且會在「主動式相互關聯技術」引擎正在執行時保留作用中狀態。
<dateTime>	<never>		*Y*	在指定的日期及時間啓用規則。事件 *Y* 會停用該規則。
<dateTime>	<never>	*X*	*Y*	在指定的日期及時間啓用規則。事件 *Y* 會停用該規則，而事件 *X* 會重新啓用它。停用及重新啓用可以發生多次。
<dateTime>	<after>			規則是在指定的日期及時間啓用，並在指定的時間間隔之後停用。
<dateTime>	<after>	*X*	*Y*	規則是在指定的日期及時間啓用，並在指定的時間間隔之後停用。事件 *X* 會啓用該規則，而事件 *Y* 會停用它。啓用及停用可以發生多次。

activationTime 元素

<activationTime> 元素會定義啓用或停用規則的離散時間點。

屬性

<activationTime> 沒有屬性。

包含範圍

<activationTime> 內含在下列元素中：

- <activationInterval>

包含

<activationTime> 包含下列元素。

元素必須按照所顯示的次序來編碼。如果元素是選用的，則無需進行編碼，但所有編碼的元素都必須遵循正確的次序。

表 19. <activationTime> 元素中包含的元素

元素	必要的或選用的？
<start>	選用的。出現次數 0 或 1 是可接受的。
<stop>	選用的。出現次數 0 或 1 是可接受的。

after 元素

<after> 元素會指定規則變成作用中之後保持作用中的持續期間。在此持續期間之後，便會停用規則。

屬性

<after> 具有下列屬性：

表 20. <after> 元素的屬性

名稱	說明	資料類型	必要的嗎？
duration	指定持續的時間量。此屬性的資料類型是根據 unit 屬性的值而定。	• 如果 unit 屬性的值爲 ISO-8601，則資料類型爲 xsd:duration。 • 如果 unit 屬性的值爲 milliseconds，則資料類型爲 xsd:positiveInteger。	是
unit	指定要使用的時間單位。此屬性的有效值爲： • ISO-8601 • milliseconds	xsd:string	是

對持續期間使用 ISO 8601 標準

將 ISO-8601 編碼爲 unit 屬性的值，表示會根據 ISO 8601 標準來編碼 duration 屬性的值，以將持續期間指定爲一個字串。標準 XML 綱目資料類型規格會使用 ISO 8601 來提供稱爲 duration 的資料類型。如需此資料類型的詳細說明，請參閱 http://www.w3.org/TR/xmlschema-2/#duration。

標準 XML 綱目中之 duration 資料類型的格式爲下列字串：

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

- P 是永遠位於字串開頭的字元。
- *n*Y 代表年數。一年等於 365 天。因此，1Y 與 365D 的編碼相同。
- *n*M 代表月數。一個月等於 30 天。因此，1M 與 30D 的編碼相同。
- *n*D 代表天數。
- T 是分隔日單位 (年、月及日) 與時間單位 (時、分及秒) 的分隔符號。時間單位永遠都在 T 的後面。
- *n*H 代表小時數。
- *n*M 代表分鐘數。
- *n*S 代表秒數。

下列爲格式範例：

- P5DT12H 爲 5.5 天。
- PT59M59S 爲 59 分鐘 59 秒。
- P1M 爲 1 個月。

包含範圍

<after> 內含在下列元素中：

- <stop>

包含

<after> 不包含任何元素。

attributeAlias 元素

<attributeAlias> 元素會提供別名，爲不同事件中意義相同但名稱不同的事件屬性建立關聯性。例如，三個不同的事件可能會對指出事件來源之系統名稱的事件屬性使用下列三種不同的名稱：host、hostname 及 source。<attributeAlias> 元素包含 <eventAttribute> 元素，該元素會說明個別事件屬性，而這些事件屬性必須相互關聯成爲分組鍵的一個事件屬性。

詳細資訊

<attributeAlias> 元素及其 aliasName 屬性只在分組鍵環境定義中有效。您無法在任何表示式中 (包括 <computedValue> 元素中的表示式) 參照此元素及其屬性。

屬性

<attributeAlias> 具有下列屬性：

表 21. <attributeAlias> 元素的屬性

名稱	說明	資料類型	必要的嗎？
aliasName	定義事件屬性的名稱，這些事件屬性會在 <eventAttribute> 元素中予以說明，而且必須相互關聯成爲分組鍵的一個事件屬性。此名稱必須是規則中唯一的。	xsd:NMTOKEN	是

包含範圍

<attributeAlias> 內含在下列元素中：

- <groupingKey>

包含

<attributeAlias> 包含下列元素：

表 22. <attributeAlias> 元素中包含的元素

元素	必要的或選用的？
<eventAttribute>	此元素必須出現 2 次。可出現更多次。

attributeName 元素

<attributeName> 元素包含了屬於分組鍵部分之特定事件屬性的名稱。此名稱必須符合在 act_event 變數之 getAttribute 方法呼叫中所使用的名稱。

屬性

<attributeName> 沒有屬性。

包含範圍

<attributeName> 內含在下列元素中：

- <groupingKey>

包含

<attributeName> 不包含任何元素。

booleanThreshold 元素

<booleanThreshold> 元素只對臨界值規則有效。它包含了接收每個事件時都會呼叫的表示式。此表示式會根據目前的事件以及規則已接受的任何其他事件，計算或比較臨界值。表示式會傳回布林值 `true` 或 `false`，以指出是否符合臨界值。

詳細資訊

如需可在表示式中使用之變數的相關資訊，請參閱第 21 頁的『變數』。某些變數的使用取決於表示式的環境定義。

屬性

<booleanThreshold> 具有下列屬性：

表 23. <booleanThreshold> 元素的屬性

名稱	說明	資料類型	必要的嗎？
expressionLanguage	識別用來撰寫表示式的程式設計語言。因爲 Java 程式設計語言是唯一受支援的表示式語言 ，所以此屬性唯一的有效值是 `java`。	xsd:NMTOKEN	是

包含範圍

<booleanThreshold> 內含在下列元素中：

- <thresholdRule>

包含

<booleanThreshold> 不包含任何元素。

相關概念

第 17 頁的『表示式』
表示式是包含自訂邏輯的程式碼，您可以將這些自訂邏輯新增至規則。表示式還可以存取「主動式相互關聯技術」引擎的外部程式碼。在規則語言中， 表示式僅在特定環境定義或規則語言元素內才有效。

collectionRule 元素

<collectionRule> 元素會根據收集型樣來定義規則。

屬性

<collectionRule> 具有下列屬性：

表 24. <collectionRule> 元素的屬性

名稱	說明	資料類型	必要的嗎？
name	識別規則。此 ID 在包含此規則的規則區塊中必須是唯一的。它不能包含句點。	xsd:NMTOKEN	是
processOnlyForwardedEvents	決定規則會接收所有事件，或是只接收從其他規則轉遞來的事件。預設值爲 `false`，這表示規則會接收所有事件，包括那些從其他規則轉遞來的事件。	xsd:boolean	否

包含範圍

<collectionRule> 內含在下列元素中：

- <ruleBlock>

包含

<collectionRule> 包含下列元素。

元素必須按照所顯示的次序來編碼。如果元素是選用的，則無需進行編碼，但所有編碼的元素都必須遵循正確的次序。

表 25. <collectionRule> 元素中包含的元素

元素	必要的或選用的？
<comment>	選用的。出現次數 0 或 1 是可接受的。
<variable>	選用的。出現次數 0 或更多是可接受的。
<activationInterval>	選用的。出現次數 0 或 1 是可接受的。
<lifeCycleActions>	選用的。出現次數 0 或 1 是可接受的。
<eventSelector>	選用的。出現次數 0 或 1 是可接受的。
<groupingKey>	選用的。出現次數 0 或 1 是可接受的。
<timeWindow>	必要的。只允許出現 1 次。
<onTimeWindowComplete>	選用的。出現次數 0 或 1 是可接受的。

相關概念

第 10 頁的『收集型樣』
收集規則是由收集型樣來定義。它會在時間間隔內收集選定的事件群組。它是有狀態的規則。

comment 元素

<comment> 元素可以包含其內含規則集、規則區塊、規則或變數的功能及用途說明。

屬性

<comment> 沒有屬性。

包含範圍

<comment> 內含在下列元素中：

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

包含

<comment> 不包含任何元素。

computationRule 元素

<computationRule> 元素會根據計算型樣來定義規則。

屬性

<computationRule> 具有下列屬性：

表 26. <computationRule> 元素的屬性

名稱	說明	資料類型	必要的嗎？
name	識別規則。此 ID 在包含此規則的規則區塊中必須是唯一的。它不能包含句點。	xsd:NMTOKEN	是
processOnlyForwardedEvents	決定規則會接收所有事件，或是只接收從其他規則轉遞來的事件。預設值爲 `false`，這表示規則會接收所有事件，包括那些從其他規則轉遞來的事件。	xsd:boolean	否

包含範圍

<computationRule> 內含在下列元素中：

- <ruleBlock>

包含

<computationRule> 包含下列元素。

元素必須按照所顯示的次序來編碼。如果元素是選用的，則無需進行編碼，但所有編碼的元素都必須遵循正確的次序。

表 27. <computationRule> 元素中包含的元素

元素	必要的或選用的？
<comment>	選用的。出現次數 0 或 1 是可接受的。
<variable>	選用的。出現次數 0 或更多是可接受的。
<activationInterval>	選用的。出現次數 0 或 1 是可接受的。
<lifeCycleActions>	選用的。出現次數 0 或 1 是可接受的。
<eventSelector>	選用的。出現次數 0 或 1 是可接受的。
<groupingKey>	選用的。出現次數 0 或 1 是可接受的。
<computeFunction>	必要的。只允許出現 1 次。
<timeWindow>	必要的。只允許出現 1 次。
<onTimeWindowComplete>	選用的。出現次數 0 或 1 是可接受的。

相關概念

第 10 頁的『計算型樣』
計算規則是由計算型樣來定義。在某個時間間隔內，接收到每個事件時，此規則便會將計算 (透過表示式) 套用至收集到的事件。它是有狀態的規則。

computedThreshold 元素

<computedThreshold> 元素只對臨界值規則有效。它包含一個在接收到每個事件時都會呼叫的表示式，此表示式會根據目前的事件以及符合規則事件選擇準則的任何其他事件來計算臨界值。此表示式會傳回計算的臨界值，此值會儲存在爲規則定義的變數中。然後，規則會使用計算的臨界值，與定義的臨界值相互比較。

詳細資訊

如需可在表示式中使用之變數的相關資訊，請參閱第 21 頁的『變數』。某些變數的使用取決於表示式的環境定義。

屬性

<computedThreshold> 具有下列屬性：

表 28. <computedThreshold> 元素的屬性

名稱	說明	資料類型	必要的嗎？
expressionLanguage	識別用來撰寫表示式的程式設計語言。因爲 Java 程式設計語言是唯一受支援的表示式語言 ，所以此屬性唯一的有效值是 `java`。	xsd:NMTOKEN	是
threshold	定義要符合的臨界值。這個定義的臨界值必須是數值的字串表示法，且可轉換爲對規則變數有效的資料類型。	xsd:string	是
assignTo	識別變數的名稱，該變數包含從此表示式傳回的計算臨界值。您必須已使用 <variable> 元素，爲規則 (在規則集、規則區塊或規則層級) 定義此變數。而且，必須將它定義爲下列其中一個數值資料類型： • java.lang.Double • java.lang.Float • java.lang.Integer • java.lang.Long • java.lang.String 如果是在規則集或規則區塊層次定義變數，則在符合規則型樣之後，不會重新起始設定此變數。	xsd:NMTOKEN	是
thresholdComparison	定義用於比較計算臨界值與定義臨界值的運算子。此運算子的有效值爲： • `lessThan` • `lessThanOrEqualTo` • `greaterThan` • `greaterThanOrEqualTo`	xsd:string	是

包含範圍

<computedThreshold> 內含在下列元素中：

- <thresholdRule>

包含

<computedThreshold> 不包含任何元素。

相關概念

第 17 頁的『表示式』
表示式是包含自訂邏輯的程式碼，您可以將這些自訂邏輯新增至規則。表示式還可以存取「主動式相互關聯技術」引擎的外部程式碼。在規則語言中， 表示式僅在特定環境定義或規則語言元素內才有效。

computedValue 元素

<computedValue> 元素包含一個在規則收到事件時就會執行的表示式，此表示式會根據事件的一或多個屬性值來建立字串值。然後，就可以在分組鍵中使用此字串值。

詳細資訊

有時，規則撰寫者可能想要在分組鍵中使用下列項目：

- 事件屬性值的子字串。例如，如果事件屬性值包含內含的 IP 位址，則 <computedValue> 元素中的表示式可以擷取該 IP 位址，作爲要在分組鍵中使用的唯一值。
- 數個不同事件屬性值的子字串。例如，<computedValue> 元素中的表示式可以擷取子字串，並將它們合併，以建立要在分組鍵中使用的唯一值。

如果 <computedValue> 元素中的表示式傳回空值，則規則會將此空值視爲遺漏的屬性值。

如需可在表示式中使用之變數的相關資訊，請參閱第 21 頁的『變數』。某些變數的使用取決於表示式的環境定義。

屬性

<computedValue> 具有下列屬性：

表 29. <computedValue> 元素的屬性

名稱	說明	資料類型	必要的嗎？
expressionLanguage	識別用來撰寫表示式的程式設計語言。因爲 Java 程式設計語言是唯一受支援的表示式語言 ，所以此屬性唯一的有效值是 `java`。	xsd:NMTOKEN	是

包含範圍

<computedValue> 內含在下列元素中：

- <groupingKey>

包含

<computedValue> 不包含任何元素。

相關概念

第 17 頁的『表示式』
表示式是包含自訂邏輯的程式碼，您可以將這些自訂邏輯新增至規則。表示式還可以存取「主動式相互關聯技術」引擎的外部程式碼。在規則語言中， 表示式僅在特定環境定義或規則語言元素內才有效。

computeFunction 元素

<computeFunction> 元素只對計算規則有效。它包含一個在接收到每個事件時都會呼叫的表示式，此表示式會傳回要儲存在針對規則所定義之變數中的值。從此表示式傳回的值，必須符合在 <computeFunction> 元素之 assignTo 屬性中命名之變數的資料類型。

詳細資訊

如需可在表示式中使用之變數的相關資訊，請參閱第 21 頁的『變數』。某些變數的使用取決於表示式的環境定義。

屬性

<computeFunction> 具有下列屬性：

表 30. <computeFunction> 元素的屬性

名稱	說明	資料類型	必要的嗎？
expressionLanguage	識別用來撰寫表示式的程式設計語言。因爲 Java 程式設計語言是唯一受支援的表示式語言 ，所以此屬性唯一的有效值是 `java`。	xsd:NMTOKEN	是
assignTo	識別變數的名稱，該變數包含從此表示式傳回的值。您必須已使用 <variable> 元素，爲規則 (在規則集、規則區塊或規則層級) 定義此變數。如果是在規則集或規則區塊層次定義變數，則在符合規則型樣之後，不會重新起始設定此變數。	xsd:NMTOKEN	是

包含範圍

<computeFunction> 內含在下列元素中：

- <computationRule>

包含

<computeFunction> 不包含任何元素。

相關概念

第 17 頁的『表示式』
表示式是包含自訂邏輯的程式碼，您可以將這些自訂邏輯新增至規則。表示式還可以存取「主動式相互關聯技術」引擎的外部程式碼。在規則語言中， 表示式僅在特定環境定義或規則語言元素內才有效。

dateTime 元素

<dateTime> 元素會指定啓用或停用規則的日期及時間。然而，只有當規則已在這個指定的時間之前載入執行中的「主動式相互關聯技術」引擎時，才能啓用或停用此規則。

詳細資訊

如果規則未在指定的啓用時間之前載入執行中的「主動式相互關聯技術」引擎，便永遠無法啓用此規則。如果規則未在指定的停用時間之前載入執行中的「主動式相互關聯技術」引擎，則會將此規則設爲 <start> 元素所定義的狀態，且 <stop> 元素永遠無法停用它。

<dateTime> 元素的內容必須是字串，並使用標準 XML 綱目中之 `dateTime` 資料類型的格式。例如，`dateTime` 由有限長度的字元順序組成，格式如下：

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

- *yyyy* 是一個四位 (或以上) 數字，代表年份。如果超過四位，則會禁止使用前導零，並且會禁止使用 0000。
- 剩餘的 '-' 是各個日期部分之間的分隔符號。
- 第一個 *mm* 是一個兩位數字，代表月份，並從 01 開始。
- *dd* 是一個兩位數字，代表日期，並從 01 開始。
- T 是分隔符號，指出後面是時間。
- *hh* 是一個兩位數字，代表小時 (24 小時制)，從 00 開始，到 23 結束。
- : 是各個時間部分之間的分隔符號。
- 第二個 *mm* 是一個兩位數字，代表分鐘，從 00 開始，到 59 結束。
- *ss* 是一個兩位數字，代表整數秒，從 00 開始，到 59 結束。
- '.' *s*+ (如果有的話) 代表小數秒。
- *zzzzzz* (如果有的話) 代表時區。時區由有限長度的字元順序組成，格式爲 (('+' | '-') *hh* ':' *mm*) | 'Z'，其中：
  - '+' (如果有的話) 代表非負數的持續時間，且不得出現 '-'。
  - '-' (如果有的話) 代表非正數的持續時間，且不得出現 '+'。
  - *hh* 是一個兩位數字，代表小時，從 00 開始，到 14 結束。
  - *mm* 是一個兩位數字，代表分鐘，從 00 開始，到 59 結束。然而，如果小時值爲 14，則分鐘值必須爲 00
  - Z 是 UTC (+00:00 或 -00:00) 的速記表示法，如此一來，就不得存在任何其他的時區元素。

下面是 <dateTime> 元素內容的兩個範例：

- 2005-06-01T13:05:06.07 是當地時間 2005 年 6 月 1 日下午 1 點 05 分 6.07 秒。
- 2005-06-01T13:05:06.07Z 是 UTC 時間 2005 年 6 月 1 日下午 1 點 05 分 6.07 秒，即 EDT 時間 2005 年 6 月 1 日上午 9 點 05 分 6.07 秒 (或 2005-06-01T09:05:06.07-04:00)

屬性

<dateTime> 沒有屬性。

包含範圍

<dateTime> 內含在下列元素中：

- <start>
- <stop>

包含

<dateTime> 不包含任何元素。

deactivateOnEvent 元素

<deactivateOnEvent> 元素會定義可以停用規則或 (若爲使用 <groupingKey> 元素來定義的規則) 規則實例的事件。

下列是選取事件的三種可能方法：

- 使用一或多個具有 <filteringPredicate> 元素的 <eventType> 元素
- 使用一或多個不具有 <filteringPredicate> 元素的 <eventType> 元素
- 使用不具有任何 <eventType> 元素的 <filteringPredicate> 元素

如果規則處於作用中，且沒有編碼 <eventType> 或 <filteringPredicate> 元素，則會選取發生的所有事件。

不編碼任何 <eventType> 元素會對系統效能產生負面影響。

假設您要選取類型爲 `Audit Failure` 的所有事件。您可以使用過濾述語來進一步修正選取準則，僅包括其事件屬性具有固定值的事件。例如，您可以編碼 <eventType> 元素，以選取類型爲 `Audit Failure` 的所有事件，並編碼 <filteringPredicate> 元素，僅選取其 hostname 屬性值爲 `MyCriticalSystem` 的那些事件。

屬性

<deactivateOnEvent> 沒有屬性。

包含範圍

<deactivateOnEvent> 內含在下列元素中：

- <activationInterval>
- <activationByGroupingKey>

包含

<deactivateOnEvent> 包含下列元素。

元素必須按照所顯示的次序來編碼。如果元素是選用的，則無需進行編碼，但所有編碼的元素都必須遵循正確的次序。

表 31. <deactivateOnEvent> 元素中包含的元素

元素	必要的或選用的？
<eventType>	選用的。出現次數 0 或更多是可接受的。
<filteringPredicate>	選用的。出現次數 0 或 1 是可接受的。

duplicateRule 元素

<duplicateRule> 元素會根據重複型樣來定義規則。

屬性

<duplicateRule> 具有下列屬性：

表 32. <duplicateRule> 元素的屬性

名稱	說明	資料類型	必要的嗎？
name	識別規則。此 ID 在包含此規則的規則區塊中必須是唯一的。它不能包含句點。	xsd:NMTOKEN	是
processOnlyForwardedEvents	決定規則會接收所有事件，或是只接收從其他規則轉遞來的事件。預設值爲 `false`，這表示規則會接收所有事件，包括那些從其他規則轉遞來的事件。	xsd:boolean	否

包含範圍

<duplicateRule> 內含在下列元素中：

- <ruleBlock>

包含

<duplicateRule> 包含下列元素。

元素必須按照所顯示的次序來編碼。如果元素是選用的，則無需進行編碼，但所有編碼的元素都必須遵循正確的次序。

表 33. <duplicateRule> 元素中包含的元素

元素	必要的或選用的？
<comment>	選用的。出現次數 0 或 1 是可接受的。
<variable>	選用的。出現次數 0 或更多是可接受的。
<activationInterval>	選用的。出現次數 0 或 1 是可接受的。
<lifeCycleActions>	選用的。出現次數 0 或 1 是可接受的。
<eventSelector>	選用的。出現次數 0 或 1 是可接受的。
<groupingKey>	選用的。出現次數 0 或 1 是可接受的。
<timeWindow>	必要的。只允許出現 1 次。
<onDetection>	選用的。出現次數 0 或 1 是可接受的。
<onNextEvent>	選用的。出現次數 0 或 1 是可接受的。
<onTimeWindowComplete>	選用的。出現次數 0 或 1 是可接受的。

相關概念

第 11 頁的『重複型樣』

重複規則是由重複型樣來定義。此規則會計算在指定時間間隔內接受的第二個及後續事件數， 但會略過這些事件的規則集處理程序。它是有狀態的規則。

eventAttribute 元素

<eventAttribute> 元素會提供方法，將事件類型與事件屬性相關聯，作爲 <attributeAlias> 元素所定義之屬性別名的一部分。

屬性

<eventAttribute> 具有下列屬性：

表 34. <eventAttribute> 元素的屬性

名稱	說明	資料類型	必要的嗎？
type	會定義事件類型的名稱。此名稱與 <eventType> 元素上的 type 屬性所使用的名稱相同。	xsd:NMTOKEN	是
attributeName	會指定透過屬性別名與其他事件屬性相關聯之事件屬性的完整名稱。此名稱必須符合 act_event 變數中所使用的名稱，才能呼叫 getAttribute 方法。	xsd:string	是

包含範圍

<eventAttribute> 內含在下列元素中：

- <attributeAlias>

包含

<eventAttribute> 不包含任何元素。

eventCountThreshold 元素

<eventCountThreshold> 元素僅對臨界值規則有效。它會定義在特定時段中必須符合事件選擇準則的事件數目。<eventCountThreshold> 元素還會指定時間範圍之兩種可能的時間間隔模式的其中之一：固定或可調整。

詳細資訊

固定間隔

固定間隔開始於接收到第一個符合事件選擇準則的事件時，並結束於發生下列其中一項狀況時：

- 規則符合其在指定持續期間內的臨界值。
- 已經過指定的持續期間。

可調整的間隔

可調整的間隔開始於接收到第一個符合事件選擇準則的事件時。不過，當規則尚未符合其臨界值且已經過指定的持續期間時，時間範圍會將開始時間調整爲新的「第一個」事件 (通常是下一個接受的事件) 的事件接收時間。可調整的間隔會繼續以此方式調整，直到發生下列其中一項狀況爲止：

- 規則符合其在指定持續期間內的臨界值。
- 接收到開始時間範圍的事件之後，在指定的持續期間內未接收到後續事件。

開始時間範圍的事件 (變成新的「第一個」事件) 的接收時間符合以下準則：新增至規則之時間間隔期間的接收時間大於目前的時間。下面是方程式形式的準則：

事件接收時間 + 規則的時間間隔期間 > 目前的時間

當不存在此類事件時，可調整的間隔就無法再調整時間，間隔便會結束。

臨界值規則會計算每個接受的事件，直到達到臨界值或時段結束爲止。然後，它會視需要執行在 <onDetection> 元素或 <onTimeOut> 元素內定義的動作。

<onDetection> 動作

當事件計數等於由 <eventCountThreshold> 元素之臨界值屬性所定義的值 (這指出符合臨界值) 時，會執行這些動作。

<onTimeOut> 動作

視時間間隔模式是固定還是可調整，來決定何時執行這些動作。

固定模式

使用固定模式，會在時間範圍到期時執行這些動作。

可調整模式

使用可調整模式，若在接收到開始時間範圍的事件之後，未在指定的持續期間內接收到任何後續事件，就會執行這些動作。換句話說，新增到規則的時間間隔期間後，所接收到之任何事件的接收時間都不會大於目前的時間。

時間範圍的時間間隔模式是由 <eventCountThreshold> 元素的 timeIntervalMode 屬性來定義。下列實務範例說明兩種可能時間間隔模式的行爲及其差異。

說明固定及可調整模式的實務範例

假設規則接收到四個符合事件選擇準則的事件，其接收時間分別爲：8:00、8:04、8:06 及 8:07。事件計數臨界值是 3，且時間範圍的期間是 5 分鐘。

具有 `fixed` 模式的規則行為

使用此時間間隔模式，臨界值規則會於 8:00 開始處理，並於 8:05 執行 <onTimeOut> 動作，因爲它在 5 分鐘內僅收到 2 個事件。因此，它不符合其在時間範圍內的臨界值。於 8:06 接收到第三個事件時，臨界值規則會重新開始處理，並於 8:11 執行 <onTimeOut> 動作，因爲它在 5 分鐘內僅接收到 2 個事件。

固定模式是靜態的。

具有 `sliding` 模式的規則行為

使用此時間間隔模式，臨界值規則會於 8:00 開始處理。在 8:05 (排定時間範圍結束的時間) 時，規則判定它僅接收到 2 個事件。然後，規則會捨棄它於 8:00 接收到的事件，並重新計算期間於 8:09 結束 (因爲現在的第一個事件是它於 8:04 接收到的事件)。當規則於 8:07 接收到事件時，它會執行 <onDetection> 動作，因爲它現在已符合其最新時間範圍 (8:04 - 8:09) 內的臨界值 (分別於 8:04、8:06 及 8:07 接收到 3 個事件)。

可調整模式是動態的，所以它會繼續調整開始時間，以嘗試符合其在時間範圍內的臨界值。

現在假設規則接收到 4 個符合事件選擇準則的事件，其接收時間分別爲：8:00、8:04、8:06 及 8:10。事件計數臨界值是 3，且時間範圍的期間是 5 分鐘。

具有 `sliding` 模式的規則行為

在此情況下，臨界值規則會於 8:00 開始處理。在 8:05 (排定時間範圍結束的時間) 時，規則判定它僅接收到 2 個事件。然後，規則會捨棄它於 8:00 接收到的事件，並重新計算期間於 8:09 結束 (因爲現在的第一個事件是它於 8:04 接收到的事件)。

在 8:09 (現在排定時間範圍結束的時間) 時，規則判定它僅接收到 2 個事件。然後，規則會捨棄它於 8:04 接收到的事件，並重新計算期間於 8:11 結束 (因爲現在的第一個事件是它於 8:06 接收到的事件)。

在 8:11 (現在排定時間範圍結束的時間) 時，規則判定它僅接收到 2 個事件。然後，規則會捨棄它於 8:06 接收到的事件，並重新計算期間於 8:15 結束 (因爲現在的第一個事件是它於 8:10 接收到的事件)。

在 8:15 (現在排定時間範圍結束的時間) 時，規則判定自從 8:10 (開始時間範圍) 接收到的事件之後，未接收到任何事件。然後，規則會執行 <onTimeOut> 動作。

屬性

<eventCountThreshold> 具有下列屬性：

表 35. <eventCountThreshold> 元素的屬性

名稱	說明	資料類型	必要的嗎？
臨界值	會定義在特定時段內必須符合事件選擇準則的事件數目。這是要符合的事件計數臨界值。此值必須是正整數。	xsd:positiveInteger	是
timeIntervalMode	會定義時間範圍的時間間隔是固定還是可調整。此屬性的有效值爲： • `fixed` (預設值) • `sliding`	xsd:string	否

包含範圍

<eventCountThreshold> 內含在下列元素中：

- <thresholdRule>

包含

<eventCountThreshold> 不包含任何元素。

eventSelector 元素

<eventSelector> 元素會定義規則所選取以進行處理的事件。

詳細資訊

下列是選取事件的三種可能方法：

- 使用一或多個具有 <filteringPredicate> 元素的 <eventType> 元素
- 使用一或多個不具有 <filteringPredicate> 元素的 <eventType> 元素
- 使用不具有任何 <eventType> 元素的 <filteringPredicate> 元素

在要規則處理所有事件的特殊情況下，您具有下列選項：

- 不編碼 <eventSelector> 元素。
- 編碼不含任何元素的 <eventSelector> 元素。

不編碼任何 <eventType> 元素會對系統效能產生負面影響。

假設您要選取類型爲 `Audit Failure` 的所有事件。您可以使用過濾述語來進一步修正選取準則，僅包括其事件屬性具有固定值的事件。例如，您可以編碼 <eventType> 元素，以選取類型爲 `Audit Failure` 的所有事件，並編碼 <filteringPredicate> 元素，僅選取其 hostname 屬性值爲 `MyCriticalSystem` 的那些事件。

屬性

<eventSelector> 具有下列屬性：

表 36. <eventSelector> 元素的屬性

名稱	說明	資料類型	必要的嗎？
alias	此屬性僅在序列規則內有效，序列規則是唯一具有多個 <eventSelector> 元素的規則。它會將此特定事件選擇器在序列規則中所選取的事件命名爲唯一的名稱。然後，過濾述語及動作可以使用此別名來存取該事件。	xsd:NMTOKEN	否

包含範圍

<eventSelector> 內含在下列元素中：

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

包含

<eventSelector> 包含下列元素。

元素必須按照所顯示的次序來編碼。如果元素是選用的，則無需進行編碼，但所有編碼的元素都必須遵循正確的次序。

表 37. <eventSelector> 元素中包含的元素

元素	必要的或選用的？
<eventType>	選用的。出現次數 0 或更多是可接受的。
<filteringPredicate>	選用的。出現次數 0 或 1 是可接受的。

eventType 元素

<eventType> 元素會定義由規則選取以進行處理或者啓用或停用規則的事件類型。

屬性

<eventType> 具有下列屬性：

表 38. <eventType> 元素的屬性

名稱	說明	資料類型	必要的嗎？
type	會定義事件類型。 若爲符合「公用基礎事件」規格的事件，則此名稱是 extensionName 屬性的值。 若爲 IBM Tivoli Enterprise Console® 事件，則此名稱是在 BAROC 檔案中定義的事件類別名稱。 基於其他格式的事件可能會使用不同的屬性來指定事件類型。	xsd:NMTOKEN	是

包含範圍

<eventType> 內含在下列元素中：

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

包含

<eventType> 不包含任何元素。

filteringPredicate 元素

<filteringPredicate> 元素包含表示式，此表示式會進一步限制規則所選取以進行處理的事件，或限制選取以啓用或停用規則的事件。因此，您可以更全面地過濾事件，而不是僅透過 <eventType> 元素依事件類型來過濾。

詳細資訊

表示式會定義條件並傳回布林值 `true` (表示符合條件) 或 `false` (表示不符合條件)。

如需可在表示式中使用之變數的相關資訊，請參閱第 21 頁的『變數』。某些變數的使用取決於表示式的環境定義。

屬性

<filteringPredicate> 具有下列屬性：

表 39. <filteringPredicate> 元素的屬性

名稱	說明	資料類型	必要的嗎？
expressionLanguage	識別用來撰寫表示式的程式設計語言。因爲 Java 程式設計語言是唯一受支援的表示式語言 ，所以此屬性唯一的有效值是 `java`。	xsd:NMTOKEN	是

包含範圍

<filteringPredicate> 內含在下列元素中：

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

包含

<filteringPredicate> 不包含任何元素。

相關概念

第 17 頁的『表示式』
表示式是包含自訂邏輯的程式碼，您可以將這些自訂邏輯新增至規則。表示式還可以存取「主動式相互關聯技術」引擎的外部程式碼。在規則語言中， 表示式僅在特定環境定義或規則語言元素內才有效。

filterRule 元素

<filterRule> 元素會根據過濾型樣來定義規則。

屬性

<filterRule> 具有下列屬性：

表 40. <filterRule> 元素的屬性

名稱	說明	資料類型	必要的嗎？
name	識別規則。此 ID 在包含此規則的規則區塊中必須是唯一的。它不能包含句點。	xsd:NMTOKEN	是
processOnlyForwardedEvents	決定規則會接收所有事件，或是只接收從其他規則轉遞來的事件。預設值爲 `false`，這表示規則會接收所有事件，包括那些從其他規則轉遞來的事件。	xsd:boolean	否

包含範圍

<filterRule> 內含在下列元素中：

- <ruleBlock>

包含

<filterRule> 包含下列元素。

元素必須按照所顯示的次序來編碼。如果元素是選用的，則無需進行編碼，但所有編碼的元素都必須遵循正確的次序。

表 41. <filterRule> 元素中包含的元素

元素	必要的或選用的？
<comment>	選用的。出現次數 0 或 1 是可接受的。
<variable>	選用的。出現次數 0 或更多是可接受的。
<activationInterval>	選用的。出現次數 0 或 1 是可接受的。
<lifeCycleActions>	選用的。出現次數 0 或 1 是可接受的。
<eventSelector>	選用的。出現次數 0 或 1 是可接受的。
<onDetection>	選用的。出現次數 0 或 1 是可接受的。

相關概念

第 12 頁的『過濾型樣』
過濾規則是由過濾型樣來定義。它會在接受事件時採取特定的動作。它只會在單一事件上執行，因此是無狀態的規則。

groupingKey 元素

通常，每個作用中的規則都有一個規則實例 (或複本) 在「主動式相互關聯技術」引擎中執行。然而，有時不同的事件群組會需要相同的規則，而這些事件群組通常與不同的資源群組有關。分組鍵是一或多個事件屬性，或事件屬性的一部分，可以用來將已選取的事件分隔成不同群組，以作爲群組進行唯一處理。<groupingKey> 元素會定義規則的分組鍵。<groupingKey> 元素的目的是針對分享共用性質 (如包含分組鍵之屬性的值所定義) 的每個事件群組，指引規則來建立個別規則實例 (或其複本)。

詳細資訊

下列兩個實務範例說明分組鍵的重要性。

實務範例 1:
發生兩個事件，DB2down 事件及 DB2up 事件。DB2 程式在名稱爲 A、B 及 C 的三台電腦上執行。A 序列規則定義爲讓 DB2down 事件與 DB2up 事件相互關聯，並在 DB2 程式停止及不重新啓動時警示操作員。

如果不使用分組鍵來定義序列規則，且從電腦 A 接收到 DB2down 事件，則來自任何電腦的 DB2up 事件都會完成該序列，但這不能達成預期的目的。然而，如果將分組鍵定義爲 hostname 屬性，則會爲已選取事件中 hostname 屬性的每個唯一值建立規則的唯一複本或實例。「主動式相互關聯技術」引擎會將每個事件傳送至正確的規則實例 (該事件之主機名稱值的規則實例)。因此，如果從電腦 A 接收到 DB2down 事件，則「主動式相互關聯技術」引擎會爲電腦 A 建立規則實例。如果從電腦 B 接收到 DB2down

事件，則「主動式相互關聯技術」引擎會爲電腦 B 建立第二個規則實例。當從電腦 B 接收到 DB2up 事件時，「主動式相互關聯技術」引擎會處理第二個規則實例中的該事件。序列完成，且因爲來自電腦 B 的 DB2down 及 DB2up 事件已正確地相互關聯，所以會警示操作員。

實務範例 2:
`Incorrect login attempted` 訊息的事件在特定環境中的所有電腦上發生。事件包含使用者 ID。臨界值規則定義爲如果此事件在 5 分鐘之內發生 10 次以上，則向操作員發出警告。

可將分組鍵定義爲使用者 ID。然後，會爲每個唯一使用者 ID 建立新的規則實例。在唯一臨界值規則實例中會追蹤每個使用者的登入嘗試，每個實例都具有該使用者登入嘗試數目的不同計數。如果任何使用者 ID 在 5 分鐘內不正確的登入 10 次以上，則操作員會接收到警告。

此方法的其他變化包括：

- 可以將分組鍵定義爲主機名稱而不是使用者 ID。此選項可以偵測單一電腦上大量不正確的登入嘗試。
- 可以將分組鍵定義爲主機名稱與使用者 ID 的組合。此選項可以偵測特定使用者 ID 對特定電腦可能進行的攻擊嘗試。

如果針對規則所指定的所有事件類型中包含相同的屬性，則使用 <attributeName> 元素是定義分組鍵之最簡單且常用的方法。

屬性

<groupingKey> 具有下列屬性：

表 42. <groupingKey> 元素的屬性

名稱	說明	資料類型	必要的嗎？
missingAttributeHandling	定義規則在下列任一條件下必須執行的動作： • 當已選取的事件具有參與分組鍵的屬性，但該屬性的值遺失時 • 當 <computedValue> 元素中的表示式傳回空值時。規則會將此空值當作遺失的屬性值。 missingAttributeHandling 屬性的有效值爲： • `ignoreEvent` (預設值)，這表示規則會忽略事件，而不會對它執行任何動作。 • `ignoreAttribute`，這表示規則會接受事件，但會忽略值已遺失的屬性。然後，「主動式相互關聯技術」引擎會包括屬性的替換值。	xsd:string	否

包含範圍

<groupingKey> 內含在下列元素中：

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

包含

<groupingKey> 包含下列元素。

表 43. <groupingKey> 元素中包含的元素

元素	必要的或選用的？
<attributeAlias>	下列其中一個元素是必要的。編碼下列多個元素是選用的。容許這三個元素出現多次。可以按任何次序編碼這些元素。
<attributeName>	
<computedValue>	

import 元素

<import> 元素包含表示式，它可以指定要匯入以在規則內的其他表示式中使用的外部模組 (例如，Java 類別)。

詳細資訊

表示式代碼是 <import> 元素中的字串。「主動式相互關聯技術」編譯器使用 <import> 元素提供的 import 陳述式，在呼叫外部方法的規則內編譯表示式代碼。

屬性

<import> 具有下列屬性：

表 44. <import> 元素的屬性

名稱	說明	資料類型	必要的嗎？
expressionLanguage	識別用來撰寫表示式的程式設計語言。因爲 Java 程式設計語言是唯一受支援的表示式語言 ，所以此屬性唯一的有效值是 `java`。	xsd:NMTOKEN	是

包含範圍

<import> 內含在下列元素中：

- <ruleSet>
- <ruleBlock>

包含

<import> 不包含任何元素。

相關概念

第 18 頁的『匯入及存取外部模組及物件』
此範例指出您如何讓表示式可以存取外部程式碼 (例如，Java 類別) 及外部物件。外部物件爲應用程式建立用來與表示式通訊的物件。

inactiveWhenLoaded 元素

<inactiveWhenLoaded> 元素指定，當「主動式相互關聯技術」引擎載入規則時，規則處於非作用中狀態。此規則會保持非作用中，直到使用另一種方法來啓用它爲止。

屬性

<inactiveWhenLoaded> 沒有屬性。

包含範圍

<inactiveWhenLoaded> 內含在下列元素中：

- <start>

包含

<inactiveWhenLoaded> 不包含任何元素。

lifeCycleActions 元素

<lifeCycleActions> 元素包含定義要在規則生命週期的四個主要階段中採取動作的元素。

詳細資訊

會在實際載入或啓用規則之後，且在規則開始進行處理之前，呼叫爲載入及啓動階段定義的動作。會在實際停用或卸載規則之前，呼叫爲停用及卸載階段定義的動作。

屬性

<lifeCycleActions> 沒有屬性。

包含範圍

<lifeCycleActions> 內含在下列元素中：

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

包含

<lifeCycleActions> 包含下列元素。

元素必須按照所顯示的次序來編碼。如果元素是選用的，則無需進行編碼，但所有編碼的元素都必須遵循正確的次序。

表 45. <lifeCycleActions> 元素中包含的元素

元素	必要的或選用的？
<onLoad>	選用的。出現次數 0 或 1 是可接受的。
<onActivation>	選用的。出現次數 0 或 1 是可接受的。
<onDeactivation>	選用的。出現次數 0 或 1 是可接受的。
<onUnload>	選用的。出現次數 0 或 1 是可接受的。

never 元素

<never> 元素指定永不在某個時間內停用的規則。但是仍然可以透過事件或其他方法來停用規則。

屬性

<never> 沒有屬性。

包含範圍

<never> 內含在下列元素中：

- <stop>

包含

<never> 不包含任何元素。

onActivation 元素

<onActivation> 元素指定啓用規則時要採取的動作或動作集。會在啓用規則之後且在規則開始進行處理之前呼叫 <onActivation> 動作。

詳細資訊

如果規則集包含的多個規則是在相同日期及時間啓用，或是利用相同的事件來啓用，且它們具有相同的時間範圍，則不會在完全相同的時間對這些規則執行下列動作：

- <onTimeOut> 及 <onTimeWindowComplete> 元素中的規則回應動作
- <onActivation> 及 <onDeactivation> 元素中的生命週期動作

這些動作會以任意次序連續執行。不必依照在規則集中編碼這些動作的次序來執行它們。因爲在序列中，每個動作都必須在下一個動作開始前完成，所以動作不會同時執行。

屬性

<onActivation> 沒有屬性。

包含範圍

<onActivation> 內含在下列元素中：

- <lifeCycleActions>

包含

<onActivation> 包含下列元素：

表 46. <onActivation> 元素中包含的元素

元素	必要的或選用的？
<action>	選用的。出現次數 0 或更多是可接受的。

onDeactivation 元素

<onDeactivation> 元素指定在停用規則時要採取的動作或動作集。呼叫 <onDeactivation> 動作後會立即停用規則。

詳細資訊

如果規則集包含的多個規則是在相同日期及時間啓用，或是利用相同的事件來啓用，且它們具有相同的時間範圍，則不會在完全相同的時間對這些規則執行下列動作：

- <onTimeOut> 及 <onTimeWindowComplete> 元素中的規則回應動作
- <onActivation> 及 <onDeactivation> 元素中的生命週期動作

這些動作會以任意次序連續執行。不必依照在規則集中編碼這些動作的次序來執行它們。因爲在序列中，每個動作都必須在下一個動作開始前完成，所以動作不會同時執行。

屬性

<onDeactivation> 沒有屬性。

包含範圍

<onDeactivation> 內含在下列元素中：

- <lifeCycleActions>

包含

<onDeactivation> 包含下列元素：

表 47. <onDeactivation> 元素中包含的元素

元素	必要的或選用的？
<action>	選用的。出現次數 0 或更多是可接受的。

onDetection 元素

<onDetection> 元素只對重複、過濾、序列及臨界值規則有效。它指定在偵測到規則型樣時要採取的動作或動作集。

詳細資訊

表 48會針對 <onDetection> 動作適用的每個規則類型，說明如何偵測規則型樣。

表 48. 如何根據規則類型來偵測規則型樣

規則類型	如何偵測規則型樣
重複	當接收到符合事件選擇準則的第一個事件時，會偵測到此規則型樣。
過濾	當接收到符合事件選擇準則的任何事件時，會偵測到此規則型樣。
序列	當在時間範圍內以適當的次序接收到符合事件選擇準則的事件序列時，會偵測到此規則型樣。
臨界值	當在時間範圍內接收到符合事件選擇準則的事件並且到達臨界值時，會偵測到此規則型樣。

屬性

<onDetection> 沒有屬性。

包含範圍

<onDetection> 內含在下列元素中：

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

包含

<onDetection> 包含下列元素：

表 49. <onDetection> 元素中包含的元素

元素	必要的或選用的？
<action>	選用的。出現次數 0 或更多是可接受的。

onLoad 元素

<onLoad> 元素指定在執行中之「主動式相互關聯技術」引擎中載入 (或部署) 規則時要採取的動作或動作集。會在載入規則之後且在規則開始進行處理之前呼叫 <onLoad> 動作。

屬性

<onLoad> 沒有屬性。

包含範圍

<onLoad> 內含在下列元素中：

- <lifeCycleActions>

包含

<onLoad> 包含此元素：

表 50. <onLoad> 元素中包含的元素

元素	必要的或選用的？
<action>	選用的。出現次數 0 或更多是可接受的。

onNextEvent 元素

<onNextEvent> 元素只對重複規則有效。它指定當重複規則在指定的時間範圍內接收到符合事件選擇準則的第二個及每一個後續事件時要採取的動作或動作集。

詳細資訊

針對重複規則，「主動式相互關聯技術」引擎會在指定的時間範圍內，略過符合事件選擇準則之第二個及每一個後續事件的規則集處理程序。因此，編碼 <onNextEvent> 動作的唯一原因是指定第二個及每一個後續事件的替代處理程序。

屬性

<onNextEvent> 沒有屬性。

包含範圍

<onNextEvent> 內含在下列元素中：

- <duplicateRule>

包含

<onNextEvent> 包含下列元素：

表 51. <onNextEvent> 元素中包含的元素

元素	必要的或選用的？
<action>	選用的。出現次數 0 或更多是可接受的。

onTimeOut 元素

<onTimeOut> 元素僅適用於序列及臨界值規則。它指定當規則的時間範圍到期時要採取的動作或動作集。

詳細資訊

表 52會針對 <onTimeOut> 動作適用的每個規則類型，說明時間範圍的到期情況。

表 52. 根據規則類型，時間範圍的到期情況

規則類型	時間範圍的到期情況
序列	如果接受一或多個事件，但是未在時間範圍內接收到事件的完整序列，時間範圍便會到期。
臨界值	如果接受一或多個事件，但是未在時間範圍內符合臨界值，時間範圍便會到期。

如果規則集包含的多個規則是在相同日期及時間啓用，或是利用相同的事件來啓用，且它們具有相同的時間範圍，則不會在完全相同的時間對這些規則執行下列動作：

- <onTimeOut> 及 <onTimeWindowComplete> 元素中的規則回應動作
- <onActivation> 及 <onDeactivation> 元素中的生命週期動作

這些動作會以任意次序連續執行。不必依照在規則集中編碼這些動作的次序來執行它們。因爲在序列中，每個動作都必須在下一個動作開始前完成，所以動作不會同時執行。

屬性

<onTimeOut> 沒有屬性。

包含範圍

<onTimeOut> 內含在下列元素中：

- <sequenceRule>
- <thresholdRule>

包含

<onTimeOut> 包含下列元素：

表 53. <onTimeOut> 元素中包含的元素

元素	必要的或選用的？
<action>	選用的。出現次數 0 或更多是可接受的。

onTimeWindowComplete 元素

<onTimeWindowComplete> 元素僅對收集、計算、重複及計時器規則有效。它指定在規則的時間範圍結束時要採取的動作或動作集。

詳細資訊

如果規則集包含的多個規則是在相同日期及時間啓用，或是利用相同的事件來啓用，且它們具有相同的時間範圍，則不會在完全相同的時間對這些規則執行下列動作：

- <onTimeOut> 及 <onTimeWindowComplete> 元素中的規則回應動作
- <onActivation> 及 <onDeactivation> 元素中的生命週期動作

這些動作會以任意次序連續執行。不必依照在規則集中編碼這些動作的次序來執行它們。因爲在序列中，每個動作都必須在下一個動作開始前完成，所以動作不會同時執行。

屬性

<onTimeWindowComplete> 沒有屬性。

包含範圍

<onTimeWindowComplete> 內含在下列元素中：

- <collectionRule>

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

包含

<onTimeWindowComplete> 包含下列元素：

表 54. <onTimeWindowComplete> 元素中包含的元素

元素	必要的或選用的？
<action>	選用的。出現次數 0 或更多是可接受的。

onUnload 元素

<onUnload> 元素指定從執行中的「主動式相互關聯技術」引擎卸載或移除規則時要採取的動作或動作集。呼叫 <onUnload> 動作後會立即卸載規則。

屬性

<onUnload> 沒有屬性。

包含範圍

<onUnload> 內含在下列元素中：

- <lifeCycleActions>

包含

<onUnload> 包含下列元素：

表 55. <onUnload> 元素中包含的元素

元素	必要的或選用的？
<action>	選用的。出現次數 0 或更多是可接受的。

ruleBlock 元素

<ruleBlock> 元素提供一種方法，將相關的規則組成群組，並將規則組織到階層中。

屬性

<ruleBlock> 具有下列屬性：

表 56. <ruleBlock> 元素的屬性

名稱	說明	資料類型	必要的嗎？
name	識別規則區塊。此 ID 在包含此規則區塊的規則集或規則區塊中必須是唯一的。它不能包含句點。	xsd:NMTOKEN	是

包含範圍

<ruleBlock> 內含在下列元素中：

- <ruleSet>
- <ruleBlock>

包含

<ruleBlock> 包含下列元素。

如果編碼這些元素，則必須依照顯示的次序來編碼 <comment>、<import> 及 <variable> 元素。剩餘的元素則可以依照任何次序來編碼。

表 57. <ruleBlock> 元素中包含的元素

元素	必要的或選用的？
<comment>	選用的。出現次數 0 或 1 是可接受的。
<import>	選用的。出現次數 0 或更多是可接受的。
<variable>	選用的。出現次數 0 或更多是可接受的。
<ruleBlock>	選用的。出現次數 0 或更多是可接受的。
<collectionRule>	選用的。出現次數 0 或更多是可接受的。
<computationRule>	選用的。出現次數 0 或更多是可接受的。
<duplicateRule>	選用的。出現次數 0 或更多是可接受的。
<filterRule>	選用的。出現次數 0 或更多是可接受的。
<sequenceRule>	選用的。出現次數 0 或更多是可接受的。
<thresholdRule>	選用的。出現次數 0 或更多是可接受的。
<timerRule>	選用的。出現次數 0 或更多是可接受的。

ruleSet 元素

由 act:ruleSet 來定義的 <ruleSet> 元素是「主動式相互關聯技術」規則語言的根元素。所有其他元素都內含在此 <ruleSet> 元素中。

詳細資訊

「主動式相互關聯技術」語言綱目 (act:ruleSet) 與「主動式相互關聯技術」基本規則集綱目 (br:ruleSet) 所定義的 <ruleSet> 元素是重複的。然而，建立規則集時，必須在 <ruleSet> 元素上指定下列名稱空間：act:ruleSet。

屬性

<ruleSet> 具有下列屬性：

表 58. <ruleSet> 元素的屬性

名稱	說明	資料類型	必要的嗎？
name	識別規則集。此 ID 必須是唯一的。它不能包含句點。	xsd:NMTOKEN	是

包含範圍

因爲 <ruleSet> 是規則語言的根元素，所以不會內含在任何元素中。

包含

<ruleSet> 包含下列元素。

元素必須按照所顯示的次序來編碼。如果元素是選用的，則無需進行編碼，但所有編碼的元素都必須遵循正確的次序。

表 59. <ruleSet> 元素中包含的元素

元素	必要的或選用的？
<comment>	選用的。出現次數 0 或 1 是可接受的。
<import>	選用的。出現次數 0 或更多是可接受的。
<variable>	選用的。出現次數 0 或更多是可接受的。
<ruleBlock>	選用的。出現次數 0 或更多是可接受的。

runUntilDeactivated 元素

<runUntilDeactivated> 元素會指定時間範圍繼續開啓，直到停用規則爲止。因此，當規則開始處理程序時，此規則的時間範圍就會啓動，直到規則停用或從規則集中移除，或是「主動式相互關聯技術」引擎關閉爲止。

詳細資訊

包含 <runUntilDeactivated> 元素之規則的特定行爲會根據規則類型而定。表 60說明每個規則類型的規則行爲，在這些規則類型中，<timeWindow> 元素有效且包含 <runUntilDeactivated> 元素。

表 60. 編碼 <runUntilDeactivated> 時的規則行爲

規則類型	編碼 <runUntilDeactivated> 時的規則行為
收集	收集規則會接受符合其事件選擇準則的第一個事件，並繼續接受及處理事件，直到停用規則才會執行 <onTimeWindowComplete> 元素中定義的動作，然後立即執行 <onDeactivation> 元素中定義的動作。
計算	計算規則會接受符合其事件選擇準則的第一個事件，並繼續接受及處理事件，直到停用規則才會執行 <onTimeWindowComplete> 元素中定義的動作，然後立即執行 <onDeactivation> 元素中定義的動作。
重複	重複規則會接受符合其事件選擇準則的第一個事件，並繼續接受及處理事件，直到停用規則才會執行 <onTimeWindowComplete> 元素中定義的動作，然後立即執行 <onDeactivation> 元素中定義的動作。
序列	序列規則會接受符合其事件選擇準則的第一個事件，並繼續接受及處理事件，直到發生下列其中一種情況爲止： • 偵測到序列型樣。發生此情況時，會執行 <onDetection> 元素中定義的動作，且規則會回到其起始狀態。此規則會重新開始處理事件，並且會多次重複此處理程序，直到停用規則爲止。 • 當規則正在處理事件時停用它。發生此情況時，會執行 <onTimeOut> 元素中定義的動作，然後立即執行 <onDeactivation> 元素中定義的動作。

表 60. 編碼 <runUntilDeactivated> 時的規則行爲 (繼續)

規則類型	編碼 <runUntilDeactivated> 時的規則行為
臨界值	臨界值規則會接受符合其事件選擇準則的第一個事件，並繼續接受及處理事件，直到發生下列其中一種情況爲止： • 偵測到臨界值型樣。發生此情況時，會執行 <onDetection> 元素中定義的動作，且規則會回到其起始狀態。此規則會重新開始處理事件，並且會多次重複此處理程序，直到停用規則爲止。 • 當規則正在處理事件時停用它。發生此情況時，會執行 <onTimeOut> 元素中定義的動作，然後立即執行 <onDeactivation> 元素中定義的動作。
計時器	計時器規則變成作用中之後，不會執行任何動作，直到停用時才會執行 <onTimeWindowComplete> 元素中定義的動作，然後立即執行 <onDeactivation> 元素中定義的動作。忽略 <timerRule> 元素的 repeat 屬性。

屬性

<runUntilDeactivated> 沒有屬性。

包含範圍

<runUntilDeactivated> 內含在下列元素中：

- <timeWindow>

包含

<runUntilDeactivated> 不包含任何元素。

sequenceRule 元素

<sequenceRule> 元素會根據序列型樣來定義規則。序列規則是唯一容許多個事件選擇器的規則。此外，它至少必須有兩個事件選擇器。

屬性

<sequenceRule> 具有下列屬性：

表 61. <sequenceRule> 元素的屬性

名稱	說明	資料類型	必要的嗎？
name	識別規則。此 ID 在包含此規則的規則區塊中必須是唯一的。它不能包含句點。	xsd:NMTOKEN	是
processOnlyForwardedEvents	決定規則會接收所有事件，或是只接收從其他規則轉遞來的事件。預設值爲 `false`，這表示規則會接收所有事件，包括那些從其他規則轉遞來的事件。	xsd:boolean	否

表 61. <sequenceRule> 元素的屬性 (繼續)

名稱	說明	資料類型	必要的嗎？
arrivalOrder	定義事件是否必須依照規則中編碼 <eventSelector> 元素的次序到達。有效值爲： • inOrder (預設值) • randomOrder	xsd:string	否

如果 arrivalOrder 屬性的值是 randomOrder，則 <eventSelector> 元素的編碼次序就非常重要。編碼時，應該將事件選擇準則最具體的 <eventSelector> 元素，置於事件選擇準則較爲簡略之 <eventSelector> 元素的前面。否則，便無法在需要時偵測到序列。

例如，假設下列情況：

- 已定義三個 <eventSelector> 元素。
- 第一個 <eventSelector> 元素檢查事件 eventA。
- 第二個 <eventSelector> 元素檢查任何事件。
- 第三個 <eventSelector> 元素檢查事件 eventB。
- 在指定的時間範圍內，會將下列事件呈現給系統：eventA、eventB、eventC。

規則行爲如下所示，且無法在需要時偵測到序列：

1. 第一個 <eventSelector> 元素接受第一個事件 eventA。
2. 第二個 <eventSelector> 元素接受第二個事件 eventB。
3. 忽略第三個事件 eventC。

假設下列情況，且已正確編碼 <eventSelector> 元素，也就是事件選擇準則最具體的元素位於事件選擇準則較爲簡略之元素的前面：

- 第一個 <eventSelector> 元素檢查事件 eventA。
- 第二個 <eventSelector> 元素檢查事件 eventB。
- 第三個 <eventSelector> 元素檢查任何事件。

規則行爲如下所示，且偵測到序列：

1. 第一個 <eventSelector> 元素接受第一個事件 eventA。
2. 第二個 <eventSelector> 元素接受第二個事件 eventB。
3. 第三個 <eventSelector> 元素接受第三個事件 eventC。

包含範圍

<sequenceRule> 內含在下列元素中：

- <ruleBlock>

包含

<sequenceRule> 包含下列元素。

元素必須按照所顯示的次序來編碼。如果元素是選用的，則無需進行編碼，但所有編碼的元素都必須遵循正確的次序。

表 62. <sequenceRule> 元素中包含的元素

元素	必要的或選用的？
<comment>	選用的。出現次數 0 或 1 是可接受的。
<variable>	選用的。出現次數 0 或更多是可接受的。
<activationInterval>	選用的。出現次數 0 或 1 是可接受的。
<lifeCycleActions>	選用的。出現次數 0 或 1 是可接受的。
<eventSelector>	序列規則要求此元素出現 2 次。可出現更多次。
<groupingKey>	選用的。出現次數 0 或 1 是可接受的。
<timeWindow>	必要的。只允許出現 1 次。
<onDetection>	選用的。出現次數 0 或 1 是可接受的。
<onTimeOut>	選用的。出現次數 0 或 1 是可接受的。

相關概念

第 12 頁的『序列型樣』
序列規則是由序列型樣來定義。此規則會偵測在時間間隔內是否有某個事件序列到達。序列可以是依序的，也可以是隨機的。序列規則是有狀態的規則。

start 元素

<start> 元素會定義是在某個日期的某個時間啓用規則，還是在「主動式相互關聯技術」引擎載入規則時啓用規則。

詳細資訊

如果 <start> 元素完全未編碼，預設開始時間便與 <whenLoaded> 元素所定義的時間相同。

屬性

<start> 沒有屬性。

包含範圍

<start> 內含在下列元素中：

- <activationTime>

包含

<start> 包含下列元素：

表 63. <start> 元素中包含的元素

元素	必要的或選用的？
<dateTime>	需要其中一個元素，且只容許所選的元素出現 1 次。
<whenLoaded>	
<inactiveWhenLoaded>	

stop 元素

<stop> 元素會定義停用規則的時間要在某個日期的某個時間、還是在某個期間之後，或是永不在某個時間。

詳細資訊

如果 <stop> 元素完全未編碼，預設停止時間便與 <never> 元素所定義的時間相同。

屬性

<stop> 元素沒有屬性。

包含範圍

<stop> 元素內含在下列元素中：

- <activationTime>

包含

<stop> 元素包含下列元素：

表 64. <stop> 元素中包含的元素

元素	必要的或選用的？
<dateTime>	需要其中一個元素，且只容許所選的元素出現 1 次。
<never>	
<after>	

stopAfter 元素

<stopAfter> 元素會指定規則實例 (如 <groupingKey> 元素所定義) 變成作用中之後保持作用中的持續期間。在此持續期間之後，便會停用規則實例。

屬性

<stopAfter> 元素具有下列屬性：

表 65. <stopAfter> 元素的屬性

名稱	說明	資料類型	必要的嗎？
duration	指定持續的時間量。此屬性的資料類型是根據 unit 屬性的值而定。	• 如果 unit 屬性的值爲 ISO-8601，則資料類型爲 xsd:duration。 • 如果 unit 屬性的值爲 milliseconds，則資料類型爲 xsd:positiveInteger。	是
unit	指定要使用的時間單位。此屬性的有效值爲： • ISO-8601 • milliseconds	xsd:string	是

對持續期間使用 ISO 8601 標準

將 ISO-8601 編碼爲 unit 屬性的值，表示會根據 ISO 8601 標準來編碼 duration 屬性的值，以將持續期間指定爲一個字串。標準 XML 綱目資料類型規格會使用 ISO 8601 來提供稱爲 duration 的資料類型。如需此資料類型的詳細說明，請參閱 http://www.w3.org/TR/xmlschema-2/#duration。

標準 XML 綱目中之 duration 資料類型的格式爲下列字串：

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

- P 是永遠位於字串開頭的字元。
- *n*Y 代表年數。一年等於 365 天。因此，1Y 與 365D 的編碼相同。
- *n*M 代表月數。一個月等於 30 天。因此，1M 與 30D 的編碼相同。
- *n*D 代表天數。
- T 是分隔日單位 (年、月及日) 與時間單位 (時、分及秒) 的分隔符號。時間單位永遠都在 T 的後面。
- *n*H 代表小時數。
- *n*M 代表分鐘數。
- *n*S 代表秒數。

下列爲格式範例：

- P5DT12H 爲 5.5 天。
- PT59M59S 爲 59 分鐘 59 秒。
- P1M 爲 1 個月。

包含範圍

<stopAfter> 內含在 <activateOnEvent> 元素中，但前提是必須在 <activationByGroupingKey> 元素內編碼 <activateOnEvent>。

包含

<stopAfter> 不包含任何元素。

thresholdRule 元素

<thresholdRule> 元素會根據臨界值型樣來定義規則。

屬性

<thresholdRule> 具有下列屬性：

表 66. <thresholdRule> 元素的屬性

名稱	說明	資料類型	必要的嗎？
name	識別規則。此 ID 在包含此規則的規則區塊中必須是唯一的。它不能包含句點。	xsd:NMTOKEN	是

表 66. <thresholdRule> 元素的屬性 (繼續)

名稱	說明	資料類型	必要的嗎？
processOnlyForwardedEvents	決定規則會接收所有事件，或是只接收從其他規則轉遞來的事件。預設值爲 `false`，這表示規則會接收所有事件，包括那些從其他規則轉遞來的事件。	xsd:boolean	否

包含範圍

<thresholdRule> 內含在下列元素中：

- <ruleBlock>

包含

<thresholdRule> 包含下列元素。

元素必須按照所顯示的次序來編碼。如果元素是選用的，則無需進行編碼，但所有編碼的元素都必須遵循正確的次序。

表 67. <thresholdRule> 元素中包含的元素

元素	必要的或選用的？
<comment>	選用的。出現次數 0 或 1 是可接受的。
<variable>	選用的。出現次數 0 或更多是可接受的。
<activationInterval>	選用的。出現次數 0 或 1 是可接受的。
<lifeCycleActions>	選用的。出現次數 0 或 1 是可接受的。
<eventSelector>	選用的。出現次數 0 或 1 是可接受的。
<groupingKey>	選用的。出現次數 0 或 1 是可接受的。
<booleanThreshold>	需要其中一個元素，且只容許所選的元素出現 1 次。
<computedThreshold>	
<eventCountThreshold>	
<timeWindow>	必要的。只允許出現 1 次。
<onDetection>	選用的。出現次數 0 或 1 是可接受的。
<onTimeOut>	選用的。出現次數 0 或 1 是可接受的。

相關概念

第 14 頁的『臨界值型樣』
臨界值規則是由臨界值型樣來定義。此規則會收集某個時間間隔內的選定事件群組，並判定在收到每個事件之後，是否符合臨界條件。它是有狀態的規則。

timeInterval 元素

<timeInterval> 元素會指定時間範圍的期間。

屬性

<timeInterval> 具有下列屬性：

表 68. <timeInterval> 元素的屬性

名稱	說明	資料類型	必要的嗎？
duration	指定持續的時間量。此屬性的資料類型是根據 unit 屬性的值而定。	• 如果 unit 屬性的值爲 ISO-8601，則資料類型爲 xsd:duration。 • 如果 unit 屬性的值爲 milliseconds，則資料類型爲 xsd:positiveInteger。	是
unit	指定要使用的時間單位。此屬性的有效值爲： • ISO-8601 • milliseconds	xsd:string	是

對持續期間使用 ISO 8601 標準

將 ISO-8601 編碼爲 unit 屬性的值，表示會根據 ISO 8601 標準來編碼 duration 屬性的值，以將持續期間指定爲一個字串。標準 XML 綱目資料類型規格會使用 ISO 8601 來提供稱爲 duration 的資料類型。如需此資料類型的詳細說明，請參閱 http://www.w3.org/TR/xmlschema-2/#duration。

標準 XML 綱目中之 duration 資料類型的格式爲下列字串：

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

- P 是永遠位於字串開頭的字元。
- *n*Y 代表年數。一年等於 365 天。因此，1Y 與 365D 的編碼相同。
- *n*M 代表月數。一個月等於 30 天。因此，1M 與 30D 的編碼相同。
- *n*D 代表天數。
- T 是分隔日單位 (年、月及日) 與時間單位 (時、分及秒) 的分隔符號。時間單位永遠都在 T 的後面。
- *n*H 代表小時數。
- *n*M 代表分鐘數。
- *n*S 代表秒數。

下列爲格式範例：

- P5DT12H 爲 5.5 天。
- PT59M59S 爲 59 分鐘 59 秒。
- P1M 爲 1 個月。

包含範圍

<timeInterval> 內含在下列元素中：

- <timeWindow>

包含

<timeInterval> 不包含任何元素。

timerRule 元素

<timerRule> 元素會根據計時器型樣定義規則。

屬性

<timerRule> 具有下列屬性：

表 69. <timerRule> 元素的屬性

名稱	說明	資料類型	必要的嗎？
name	識別規則。此 ID 在包含此規則的規則區塊中必須是唯一的。它不能包含句點。	xsd:NMTOKEN	是
processOnlyForwardedEvents	因爲計時器規則不處理事件，所以會忽略此屬性。	xsd:boolean	否
repeat	定義計時器規則是否會一直重複執行，直到停用它爲止。有效值爲： • `true` (預設值) • `false` 如果值設爲 `false`，則規則在整個時間間隔內只執行一次，並且會在各個時間範圍結束時，執行規則回應動作，然後停止。 如果計時器規則的 <timeWindow> 元素包含 <runUntilDeactivated> 元素，則會忽略 repeat 屬性。	xsd:boolean	否

包含範圍

<timerRule> 內含在下列元素中：

- <ruleBlock>

包含

<timerRule> 包含下列元素。

元素必須按照所顯示的次序來編碼。如果元素是選用的，則無需進行編碼，但所有編碼的元素都必須遵循正確的次序。

表 70. <timerRule> 元素中包含的元素

元素	必要的或選用的？
<comment>	選用的。出現次數 0 或 1 是可接受的。
<variable>	選用的。出現次數 0 或更多是可接受的。
<activationInterval>	選用的。出現次數 0 或 1 是可接受的。
<lifeCycleActions>	選用的。出現次數 0 或 1 是可接受的。
<timeWindow>	必要的。只允許出現 1 次。
<onTimeWindowComplete>	選用的。出現次數 0 或 1 是可接受的。

相關概念

第 16 頁的『計時器型樣』
計時器規則是由計時器型樣來定義。此規則會定期起始動作。它是有狀態的規則。雖然計時器規則不處理事件，但可以使用事件來啓用或停用它。

timeWindow 元素

<timeWindow> 元素包含的元素會定義規則處理的時間間隔。

詳細資訊

例如，重複規則的時間範圍會定義規則必須在多長的時間內檢查接受的第一個事件之重複事件。如果時間範圍是 30 秒，重複規則就會處理 30 秒內發生而與它接受之第一個事件重複的所有事件。

屬性

<timeWindow> 沒有屬性。

包含範圍

<timeWindow> 內含在下列元素中：

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

包含

<timeWindow> 包含下列元素：

表 71. <timeWindow> 元素中包含的元素

元素	必要的或選用的？
<timeInterval>	需要其中一個元素，且只容許所選的元素出現 1 次。
<runUntilDeactivated>	

variable 元素

<variable> 元素會定義變數，並包含可被表示式參照的格式資訊。您可以在規則集、規則區塊或規則層次定義變數。

詳細資訊

規則集變數

適用於整個規則集，並且可被規則集中的任何表示式參照。

規則區塊變數

僅在規則區塊內 (及任何內含的規則區塊內) 適用，並且可被規則區塊內的任何表示式參照。

規則變數

僅適用於規則中的表示式。

在規則階層的不同層次，變數可以有相同的名稱。當存取變數時，會使用變數的最本端定義。例如，如果在規則集層次、規則區塊層次及規則層次定義相同名稱的變數，則規則內的表示式會使用規則層次的變數定義。

當在規則集或規則區塊層次定義變數時，多個規則會在不同時間取得並設定這些變數。因此，爲了確保正確維護變數值，請瞭解您在規則集的變數之間，撰寫交談作業程式碼的方式。

如果在規則集或規則區塊層次定義變數，則在符合規則型樣之後，不會重新起始設定它。

在下列任何一種情況下，請使用取得並設定規則集及規則區塊變數的鎖定，以防止錯誤地設定變數值：

- 如果計時器規則在 <onTimeOut> 動作期間取得並設定變數
- 如果內含「主動式相互關聯技術」引擎的應用程式是多緒應用程式

如果使用分組鍵定義規則，則 <variable> 元素定義的規則變數在生命週期動作中是無效的，在內含於 <activationInterval> 元素之 <activateOnEvent> 或 <deactivateOnEvent> 元素內的 <filteringPredicate> 元素中也是無效的。這是因爲在此情況下，規則變數僅適用於規則實例，而規則實例在這些表示式執行時不存在。

屬性

<variable> 具有下列屬性：

表 72. <variable> 元素的屬性

名稱	說明	資料類型	必要的嗎？
name	識別特定的變數。變數可由其名稱來參照。	xsd:NMTOKEN	是
dataType	識別變數包含的資訊類型。這必須是完整的資料類型，例如 java.lang.String。	xsd:NMTOKEN	是

變數的名稱限制

變數名稱具有特定的限制。因此，<variable> 元素上的 name 屬性值具有下列限制：

- 它只能包含下列字元：
  - 大寫 ASCII 拉丁文字母 A-Z。Unicode 表示法爲 \u0041-\u005a。
  - 小寫 ASCII 拉丁文字母 a-z。Unicode 表示法爲 \u0061-\u007a。
  - ASCII 底線 (_)。Unicode 表示法爲 \u005f。
  - 錢幣符號 ($)。Unicode 表示法爲 \u0024。
  - ASCII 數字 0 – 9。Unicode 表示法爲 \u0030-\u0039。
- 不能爲空值。
- 不能爲空字串。
- 不能包含任何空格。
- 不能包含句點。
- 在任何格式 (大寫、小寫或大小寫混合格式) 下都不能以 act_ 開頭。

包含範圍

<variable> 內含在下列元素中：

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

包含

<variable> 包含下列元素。

元素必須按照所顯示的次序來編碼。如果元素是選用的，則無需進行編碼，但所有編碼的元素都必須遵循正確的次序。

表 73. <variable> 元素中包含的元素

元素	必要的或選用的？
<comment>	選用的。出現次數 0 或 1 是可接受的。
<varInitializer>	必要的。只允許出現 1 次。

相關概念

第 21 頁的『變數』
在規則語言中，會使用特定的變數，儲存在不同次數出現的事件或規則之中的事件相關資訊。這個與事件相關的資訊可從規則內的表示式來存取。部分類型的變數是

由規則撰寫者來定義，而其他類型的變數則由「主動式相互關聯技術」提供。部分類型可以直接從表示式存取，而其他類型則只能透過「主動式相互關聯技術」所提供的方法來存取。

varInitializer 元素

<varInitializer> 元素包含的表示式會針對在相關 <variable> 元素中定義的變數提供起始值。

詳細資訊

因爲變數可以是任何類型，所以表示式代碼可以傳回陣列物件，或由「主動式相互關聯技術」引擎來儲存的任何其他複式、實作特定的物件。

如需可在表示式中使用之變數的相關資訊，請參閱第 21 頁的『變數』。某些變數的使用取決於表示式的環境定義。

屬性

<varInitializer> 具有下列屬性：

表 74. <varInitializer> 元素的屬性

名稱	說明	資料類型	必要的嗎？
expressionLanguage	識別用來撰寫表示式的程式設計語言。因爲 Java 程式設計語言是唯一受支援的表示式語言 ，所以此屬性唯一的有效值是 `java`。	xsd:NMTOKEN	是

包含範圍

<varInitializer> 內含在下列元素中：

- <variable>

包含

<varInitializer> 不包含任何元素。

相關概念

第 17 頁的『表示式』
表示式是包含自訂邏輯的程式碼，您可以將這些自訂邏輯新增至規則。表示式還可以存取「主動式相互關聯技術」引擎的外部程式碼。在規則語言中， 表示式僅在特定環境定義或規則語言元素內才有效。

whenLoaded 元素

<whenLoaded> 元素會指定規則要在「主動式相互關聯技術」引擎載入它時啓動。

屬性

<whenLoaded> 沒有屬性。

包含範圍

<whenLoaded> 內含在下列元素中：

- <start>

包含

<whenLoaded> 不包含任何元素。

第 6 章 名詞解釋

此名詞解釋包含「主動式相互關聯技術」中之重要概念的術語及定義。

ACT 請參閱主動式相互關聯技術 (Active Correlation Technology)。

動作 (action)
作爲規則回應的一部分或當載入、卸載、啓用或停用規則時執行的表示式。

主動式相互關聯技術 (Active Correlation Technology)
透過規則提供事件相互關聯功能的 IBM 技術。

「主動式相互關聯技術」編譯器 (Active Correlation Technology compiler)
爲「主動式相互關聯技術」元件，會剖析規則集及內含在規則集中的任何程式碼，以產生「主動式相互關聯技術」引擎所需的內部資料結構。

「主動式相互關聯技術」引擎 (Active Correlation Technology engine)
爲「主動式相互關聯技術」元件，會根據「主動式相互關聯技術」編譯器的輸出來處理事件。

「主動式相互關聯技術」規則建置器 (Active Correlation Technology rule builder)
使用「主動式相互關聯技術」規則語言來撰寫相關性規則的 GUI。

「主動式相互關聯技術」規則語言 (Active Correlation Technology rule language)
用來撰寫讓事件相互關聯之規則的 XML 型語言。然後可以將這些規則部署到「主動式相互關聯技術」執行時期環境。

「主動式相互關聯技術」執行時期環境 (Active Correlation Technology runtime environment)
內嵌「主動式相互關聯技術」引擎 (具有或不具有編譯器) 的應用程式。

收集型樣 (collection pattern)
爲規則型樣，會定義規則以在時間間隔內收集選定的事件群組。由收集型樣定義的規則是有狀態的規則。

計算型樣 (computation pattern)
爲規則型樣，會定義規則以在時間間隔內接收到每個事件時，將計算 (透過表示式) 套用至已收集的事件。由計算型樣定義的規則是有狀態的規則。

網域 (domain)
規則群組會根據規則的功能套用至其中的種類。例如，網域可以代表特定的地理區域、IT 管理紀律 (例如安全偵測或網路事件相關性)，或商業組織 (例如特定的公司或公司的部門)。

重複型樣 (duplicate pattern)
爲規則型樣，會定義規則以計算在指定的時間間隔內接受的第二個及後續事件數，但會略過這些事件的規則集處理程序。由重複型樣定義的規則是有狀態的規則。

事件提供者 (event provider)
產生由「主動式相互關聯技術」處理之事件的任何軟體。

事件選擇器 (event selector)
選取事件的準則。這些準則決定規則會接受哪些事件來進行處理。事件選擇器包括事件類型及過濾述語。

表示式 (expression)
包含可以新增至規則之自訂邏輯的程式碼。規則撰寫者可以依不同的目的使用表示式，例如起始設定變數、定義事件選擇準則，或指定規則回應動作及生命週期動作。

表示式語言 (expression language)
用來撰寫表示式的程式設計語言。

外部事件 (external event)
「主動式相互關聯技術」引擎從其外部來源接收的事件。

外部物件 (external object)
應用程式建立用來與表示式通訊的物件。

過濾述語 (filtering predicate)
此表示式會定義條件，規則會在此條件下接受事件來進行處理。過濾述語是事件選擇器的一部分。過濾述語會傳回布林值。

過濾型樣 (filter pattern)
爲規則型樣，會定義規則以在它接受事件時執行特定動作。由過濾型樣定義的規則僅在單一事件上執行，因此是無狀態的規則。

分組鍵 (grouping key)
是針對具有分享共用性質的每個事件群組，指引規則以建立個別規則實例 (或其複本) 的方法。

匯入 (import)
讓表示式可以存取外部程式碼的程式設計語言專用方式。

內部事件 (internal event)
由在「主動式相互關聯技術」引擎中執行之規則建立的事件。此事件可能會轉遞至其他規則。

生命週期動作 (life cycle action)
載入、卸載、啓用或停用規則時執行的表示式。

節點 (node)
爲規則階層內的物件，您可以從規則集內個別獨立地新增、移除或取代此規則階層。具體而言，下列物件爲節點：

- 規則
- 規則區塊
- 規則區塊變數
- 規則集變數

因爲物件不能在規則層次下個別獨立地操作，所以規則變數不是節點。

述語 (predicate)
請參閱過濾述語 (filtering predicate)。

回應 (response)
請參閱規則回應 (rule response)。

規則 (rule)

用來辨識事件之間的關係及執行適當規則回應的相關性單位。規則是七個規則型樣的其中一個實作方式，可根據其功能組織到屬於規則集之一部分的規則區塊中。如果事件符合事件選擇準則，規則便會接受該事件以進行處理。

規則區塊 (rule block)

依據功能將規則分組至規則集中之網域的組織單位。規則區塊不僅可以包含規則，還包含其他規則區塊。

規則實例 (rule instance)

在分組鍵的環境定義中，是規則的副本。

規則型樣 (rule pattern)

事件相關性狀況的呈現 (例如臨界條件或重複事件偵測)。「主動式相互關聯技術」規則語言包括下列規則型樣：收集、計算、重複、過濾、序列、臨界值及計時器。當發生規則定義的狀況時，就會比對規則型樣。比對型樣時，規則會採取適當的規則回應動作來結束其處理程序。當規則處於作用中時，可能會多次比對規則型樣。

規則回應 (rule response)

當「主動式相互關聯技術」引擎辨識已符合規則條件時所執行的表示式。規則回應由一或多個動作組成。

規則回應動作 (rule response action)

請參閱動作 (action)。

規則集 (rule set)

「主動式相互關聯技術」規則語言的規則執行單位。規則集包含由「主動式相互關聯技術」引擎執行且組織到規則區塊中的規則。引擎只會在給定的時間，在一個規則集上執行。

序列型樣 (sequence pattern)

爲規則型樣，會定義規則以在時間間隔內偵測特定事件序列是否存在。序列可以是依序的，也可以是隨機的。由序列型樣定義的規則是有狀態的規則。

片段 (snippet)

原始碼的摘錄。

有狀態的規則 (stateful rule)

保留狀態資訊的規則，它是規則實例之性質的相關資訊，目的是在一段時間內執行事件收集。由下列任何規則型樣定義的規則都是有狀態的規則：收集、計算、重複、序列、臨界值或計時器。

無狀態的規則 (stateless rule)

不保留狀態資訊的規則，因此一次只能在一個事件上執行。由過濾型樣定義的規則是無狀態的規則。

臨界值型樣 (threshold pattern)

爲規則型樣，會定義規則以在時間間隔內收集已選取事件的群組，以及在接收每個事件之後，決定是否符合臨界條件。由臨界值型樣定義的規則是有狀態的規則。

計時器型樣 (timer pattern)

爲規則型樣，會定義規則以定期起始動作。由計時器型樣定義的規則是有狀態的規則。雖然計時器規則不處理事件，但可以使用事件來啓用或停用它。

附錄. 注意事項

本資訊是針對 IBM 在美國所提供之產品與服務開發出來的，而在其他國家中，IBM 不見得有提供本書中所提的各項產品、服務、或功能。要知道您所在區域是否可用到這些產品與服務時，請向當地的 IBM 業務代表查詢。本書在提及 IBM 產品、程式或服務時，不表示或默示只能使用 IBM 的產品、程式或服務。只要未侵害 IBM 的智慧財產權，任何功能相當的產品、程式或服務都可以取代 IBM 的產品、程式或服務。不過，其他非 IBM 產品、程式、或服務在運作上的評價與驗證，其責任屬於使用者。

在這本書或文件中可能包含著 IBM 所擁有之專利或專利申請案。本文件使用者並不享有前述專利之任何授權。您可以用書面方式來查詢授權，來函請寄到：

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785 U.S.A.

若要查詢有關二位元組 (DBCS) 資訊的特許權限事宜，請聯絡您國家的 IBM 智慧財產部門，或者用書面方式寄到：

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

下列段落若與該國之法律條款抵觸，即視爲不適用：

IBM 僅以「現狀」提供本書，而不提供任何明示或默示之保證 (包括但不限於可售性或符合特定效用的保證)。

若有些地區在某些交易上並不允許排除上述保證，則該排除無效。

本資訊中可能會有技術上或排版印刷上的訛誤。因此，IBM 會定期修訂；並將修訂後的內容納入新版中。同時，IBM 得隨時修改或變更本出版品中所提及的產品及程式。

本資訊中任何對非 IBM 網站的敘述僅供參考，IBM 對該網站並不提供保證。該網站上的資料，並非本 IBM 產品所用資料的一部分，如因使用該網站而造成損害，其責任由貴客戶自行負責。

IBM 得以各種適當的方式使用或散布由 貴客戶提供的任何資訊，而無需對您負責。

本程式之獲授權者若希望取得相關資料，以便使用下列資訊者可洽詢 IBM。其下列資訊指的是： (1) 獨立建立的程式與其他程式 (包括此程式) 之間更換資訊的方式 (2) 相互使用已交換之資訊方法 若有任何問題請聯絡：

IBM Corporation
2Z4A/101
11400 Burnet Road
Austin, TX 78758 U.S.A.

上述資料之取得在某些情況下附有條件，且必須付費方得使用。

IBM 基於雙方之「IBM 客戶合約」、「IBM 國際程式授權合約」或任何同等合約之條款，提供本文件中所提及的授權程式與其所有適用的授權資料。

商標

DB2、IBM、IBM 標誌、Tivoli、Tivoli 標誌、Tivoli Enterprise Console 及 WebSphere 是 International Business Machines Corporation 在美國及 (或) 其他國家的商標或註冊商標。

Java 以及所有與 Java 爲基礎的商標及標誌是 Sun Microsystems, Inc. 在美國及 (或) 其他國家的商標或註冊商標。

其他公司、產品及服務名稱，可能是其他公司的商標或服務標誌。

Printed in Taiwan