Novell Identity Manager Driver for JDBC*

2.1

www.novell.com

實作指南

2006 年 5 月 12 日

Novell®

法律聲明

Novell, Inc. 不對本文件的內容或使用做任何陳述或保證，且特別聲明不對任何特定用途的適銷性或適用性提供任何明示或隱喻的保證。此外，Novell, Inc. 保留隨時修改本出版品及其內容的權利，且在進行此類修正或更動時，不需另行通知任何人士或公司。

此外，Novell, Inc. 不對任何軟體作任何陳述或保證，且特別聲明不對任何特定用途的適銷性或適用性提供任何明示或隱喻的保證。此外，Novell, Inc. 保留隨時修改任何或全部 Novell 軟體的權利，且在進行此類更動時，不需通知任何人士或公司。

這份授權書中所提及的任何產品或技術資訊皆受到美國出口管制法 (U.S. Export Control) 及其他國家的交易法約束。您同意遵守所有出口管制法規，並取得出口、再出口或進口交付物品所需之任何必要的授權或類別。您同意不出口或再出口至目前美國出口排除清單上所列公司，或者至美國出口法所指定之禁運或恐怖份子的國家。您同意不將交付產品用在禁止的核子武器、飛彈或化學生物武器等用途上。如需更詳細的 Novell 軟體出口資訊，請參閱 www.novell.com/info/exports/。Novell 無須承擔您無法取得任何必要的出口核准之責任。

Novell, Inc. 擁有在此份文件中所描述產品內含技術的智慧財產權。尤其 (但不限於) 這些智慧財產權可能包含一或多個列於 http://www.novell.com/company/legal/patents/ 的美國專利，以及一或多個在美國和其他國家的額外專利或申請中的專利。

Novell, Inc.
404 Wyman Street, Suite 500
Waltham, MA 02451
U.S.A.
www.novell.com

線上文件：若要存取本產品及其他 Novell 產品的線上文件，或取得更新，請參閱 www.novell.com/documentation。

Novell 商標

如需 Novell 商標之清單，請參閱商標 (http://www.novell.com/company/legal/trademarks/tmlist.html)。

協力廠商資料

所有的協力廠商商標均爲其個別擁有廠商的財產。

目錄

關於本指南

Identity Manager Driver for Java* Database Connectivity (JDBC*) 提供將 Identity Vault 與關聯式資料庫之間的資料同步化之一般解決方案。

本指南提供驅動程式技術的綜覽，以及組態設定指示。

使用對象

本指南是針對使用 Identity Manager Driver for JDBC 之 Novell® eDirectory 及 Identity Manager 管理員而撰寫的。

意見反應

我們想知道您對於本手冊及其他 Novell Identity Manager 文件的意見與建議。請使用線上文件中每頁底下的「使用者意見」功能，或造訪 www.novell.com/documentation/feedback.html，然後寫下您的意見。

文件更新

如需本文件的最新版本，請參閱 Identity Manager 文件網站 (http://www.novell.com/documentation/lg/dirxmldrivers/index.html)。

其他文件

如需使用 Identiy Manager 和其他驅動程式的相關文件，請參閱 Identity Manager 文件網站 (http://www.novell.com/documentation/lg/dirxmldrivers)。

文件慣例

本文件中使用大於符號 (>) 分隔步驟中的各個動作，以及前後參照路徑中的數個項目。

商標符號 (®、™ 等) 代表 Novell® 的商標。星號 (*) 代表協力廠商的商標。

Identity Manager Driver for JDBC 簡介

1

Identity Manager Driver for Java DataBase Connectivity (JDBC) 提供將 Identity Manager 與 JDBC 可存取之關聯式資料庫之間的資料同步化的一般解決方案。

此驅動程式的關鍵價值在於它的一般性 (泛用性)。大部份驅動程式與單一應用程式連接，而此驅動程式與之不同，它可以與大部份關聯式資料庫和資料庫裝載的應用程式連接。

1.1 驅動程式新功能

Identity Manager 3 具有下列驅動程式新功能：

- 無觸發發行。請參閱「無觸發與觸發的發行」，第 16 頁。
- 批次處理。請參閱「批次大小」，第 74 頁。
- 未來事件處理。請參閱「啓用未來事件處理？」，第 69 頁。
- 每日發行。請參閱「每日發行時間」，第 73 頁。
- 更全面的資料庫支援。請參閱「受支援的資料庫」，第 123 頁。
- 增強對資料庫時間類型的支援。請參閱「時間語法」，第 42 頁。
- 更容易使用。請參閱「智慧型組態」，第 37 頁。
- 綱要篩選。請參閱「包含過濾器運算式」，第 47 頁和「排除過濾器運算式」，第 47 頁。
- 更多樣的檢視窗支援。請參閱「直接同步化」，第 82 頁。
- 增強對協力廠商驅動程式加密機制的支援。請參閱「連接啓始化陳述式」，第 49 頁。
- 密碼修改和檢查支援。
- 更精良的驅動程式組態 / 資料庫 SQL* 程序檔。

如需 Identiy Manager 新功能的相關資訊，請參閱《 *Identity Manager 3.0* 安裝指南》中的「Identity Manager 3 的新功能」。

1.2 術語的變更

以下是與舊版不同的詞彙：

表格 *1-1* 術語的變更

舊詞彙	新詞彙
DirXML®	Identity Manager
DirXML 伺服器	Metadirectory 伺服器
DirXML 引擎	Metadirectory 引擎
eDirectory™	Identity Vault (指 eDirectory 屬性或類別時除外)

1.3 詞彙和概念

- 「JDBC」，第 8 頁
- 「Identity Manager Driver for JDBC」，第 8 頁
- 「協力廠商 JDBC 驅動程式」，第 9 頁
- 「Identity Vault」，第 9 頁
- 「目錄綱要」，第 9 頁
- 「應用程式綱要」，第 9 頁
- 「資料庫綱要」，第 9 頁
- 「同步化綱要」，第 10 頁
- 「邏輯資料庫類別」，第 10 頁
- 「XDS」，第 10 頁

1.3.1 JDBC

Java DataBase Connectivity (JDBC) 是 Sun* Microsystems* 開發的跨平台資料庫介面標準。

大部份企業資料庫廠商都會提供其獨特的 JDBC 介面實作。有三種版本的 JDBC 介面可用：

- JDBC 1 (Java 1.0)
- JDBC 2 (Java 1.2 或 1.3)
- JDBC 3 (Java 1.4 或 1.5)

Identity Manager Driver for JDBC 主要使用 JDBC 1 介面。當協力廠商 JDBC 驅動程式支援它時，它使用 JDBC 2 或 JDBC 3 方法的小型子集。

1.3.2 Identity Manager Driver for JDBC

Identity Manager Driver for JDBC 使用 JDBC 介面，將 Identity Vault 與關聯式資料庫之間的資料和身份同步化。

驅動程式由四個 jar 檔案組成：

- JDBCShim.jar
- JDBCUtil.jar
- JDBCConfig.jar

- CommonDriverShim.jar

除了這些檔案，您還需要協力廠商 JDBC 驅動程式，來與每個個別資料庫通訊。

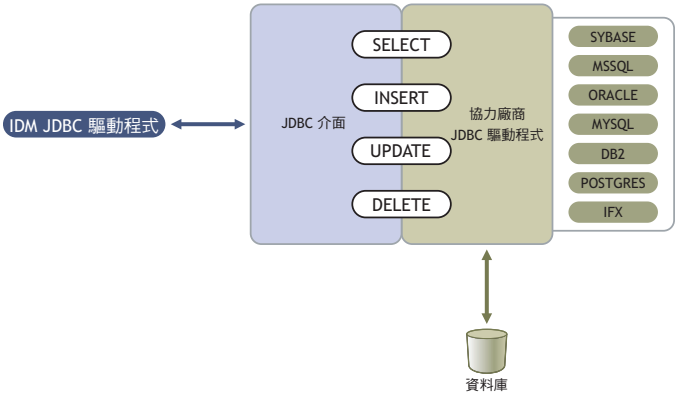
1.3.3 協力廠商 JDBC 驅動程式

協力廠商 JDBC 驅動程式是 Identity Manager Driver for JDBC 用來與特定資料庫通訊的其中一個 JDBC 介面實作。

例如，classes12.zip 是其中一個 Oracle* JDBC 驅動程式。不同的協力廠商 JDBC 驅動程式會實作 JDBC 介面規格的不同部份，並會以相對一致的方式實作介面。

下圖指出 Driver for JDBC 與協力廠商 JDBC 驅動程式之間的關係。

特性 ***1-1*** *IDM JDBC* 驅動程式與協力廠商 *JDBC* 驅動程式



1.3.4 Identity Vault

Identity Vault 是 Identity Manager 所使用的資料儲存。

1.3.5 目錄綱要

目錄綱要是目錄中的物件類別和屬性集。

例如，eDirectory™ 的「使用者」類別和「名」屬性是 eDirectory 綱要的一部份。

1.3.6 應用程式綱要

應用程式綱要是應用程式中的類別和屬性集。

因爲資料庫不瞭解類別或屬性，所以 Driver for JDBC 會將 eDirectory 類別映射至表格或檢視窗，並將 eDirectory 屬性映射至欄。

1.3.7 資料庫綱要

資料庫綱要基本上與擁有權同義。資料庫綱要由資料庫使用者所擁有的資料庫物件 (例如：表格、檢視窗、觸發、預存程序和函數) 組成。

有了 Driver for JDBC，綱要對設定資料庫的範圍就很有用 (減少執行時期驅動程式可見的資料庫數目)。

擁有權通常會藉由使用合格點標記來表示 (例如，indirect.usr 中，indirect 是擁有表格 usr 的資料庫使用者名稱)。indirect 所擁有的所有資料庫物件組成間接資料庫綱要。

1.3.8 同步化綱要

同步化綱要是執行時期驅動程式可見的資料庫綱要。

1.3.9 邏輯資料庫類別

邏輯資料庫類別是用來代表資料庫中 eDirectory 類別的表格或檢視窗集。

1.3.10 XDS

XDS 格式是 Identity Manager 可使用之可能 XML 格式的已定義 Novell® 子集。

XDS 是來自 Identity Vault 的資料啓始格式。藉由修改預設規則和變更樣式表，您可以設定 Driver for JDBC 的組態，以使用任何 XML 格式。

1.4 資料庫概念

1.4.1 結構化查詢語言

結構化查詢語言 (Structured Query Language，SQL) 是用於查詢和處理關聯式資料庫中資料的語言。

1.4.2 資料操作語言

資料操作語言 (Data Manipulation Language，DML) 陳述式是操作資料庫資料的高度標準化 SQL 陳述式。

無論您使用何種資料庫，資料操作語言 (DML) 陳述式基本上都相同。Driver for JDBC 以資料操作語言 (DML) 爲基礎。它將以 XDS XML 表示的 Identity Manager 事件映射至標準化資料操作語言 (DML) 陳述式。

下列範例顯示數個資料操作語言 (DML) 陳述式：

```
SELECT * FROM usr; INSERT INTO usr(lname) VALUES('Doe'); UPDATE usr SET
fname = 'John' WHERE idu = 1;
```

1.4.3 資料定義語言

資料定義語言 (Data Definition Language，DDL) 陳述式操作資料庫物件，例如表格、索引和使用者帳戶。

資料定義語言 (DDL) 陳述式是專屬陳述式，且在資料庫之間有很大的不同。雖然 Driver for JDBC 是以資料操作語言 (DML) 爲基礎，但您仍可以在 XDS 事件中內嵌資料定義語言 (DDL) 陳述式。如需其他資訊，請參閱「在 XDS 事件中內嵌 SQL 陳述式」，第 95 頁。

下列範例顯示數個資料定義語言 (DDL) 陳述式：

```
CREATE TABLE usr ( idu   INTEGER, fname VARCHAR2(64), lname
VARCHAR2(64) );

CREATE USER idm IDENTIFIED BY novell;
```

附註：本指南中使用的範例適用於 Oracle 資料庫。

1.4.4 檢視窗

檢視窗是邏輯表格。

使用 SELECT 陳述式進行查詢時，檢視窗由執行定義檢視窗時提供的 SQL 查詢組成。檢視窗是一個很有用的抽象機制，可以將多個任意結構的表格以單一表格或邏輯資料庫類別的方式呈現。

```
CREATE VIEW view_usr ( pk_idu, fname, lname ) AS SELECT idu, fname,
lname from usr;
```

1.4.5 身份欄 / 序列

身份欄和序列用於產生唯一的主索引鍵值。Identity Manager 可以與其他項目中的這些值相關聯。

身份欄是用於唯一識別表格中列的自行增加欄。將列插入表格時，會自動填入身份欄值。

序列物件是可用於唯一識別表格中列的計數器。與身份欄不同，序列物件不會結合至單一表格。然而，如果單一表格使用序列物件，則可以使用它來獲得相同結果。

下面爲序列物件的範例：

```
CREATE SEQUENCE seq_idu START WITH 1 INCREMENT BY 1 NOMINVALUE
NOMAXVALUE ORDER;
```

1.4.6 異動

異動是由一或多個陳述式組成的極小型資料庫操作。

完成異動時，會認可異動的所有陳述式。異動岔斷或異動中的其中一個陳述式發生錯誤時，異動會復原。異動復原時，會將資料庫保留在異動開始之前的狀態。

異動可以是手動 (使用者定義的) 或自動。手動異動可以由一或多個陳述式組成，且必須明確地進行認可。自動異動由單一陳述式組成，且在執行每個陳述式後以隱含方式進行認可。

手動 **(** 使用者定義的 **)** 異動

手動異動通常包含多個陳述式。在手動異動中，通常無法將資料定義語言 (DDL) 陳述式和資料操作語言 (DML) 陳述式組合在一起。

下列範例說明手動異動：

```
SET AUTOCOMMIT OFF INSERT INTO usr(lname) VALUES('Doe'); UPDATE usr
SET fname = 'John' WHERE idu = 1; COMMIT; -- explicit commit
```

自動異動

自動異動僅由一個陳述式組成。因爲在每個陳述式後以隱含方式認可變更，所以它們通常稱爲自動認可陳述式。自動認可陳述式獨立於任何其他陳述式之外。

下列範例說明自動異動：

```
SET AUTOCOMMIT ON INSERT INTO emp(lname) VALUES('Doe'); -- implicit
commit
```

1.4.7 預存程序或函數

預存程序或函數是儲存於資料庫中的程式邏輯。幾乎從任何網路位置都可以叫用預存程序或函數。

「訂閱者」通道可以使用預存程序或函數，從插入表格的列取回主索引鍵值，以建立關聯。您也可以從內嵌的 SQL 陳述式或觸發中叫用預存程序或函數。

預存程序與函數之間的差異視資料庫的不同而變化。通常，二者都可以傳回輸出，但是方式不同。預存程序通常會透過參數傳回值。函數通常透過純量傳回值或結果集傳回值。

下列範例說明傳回序列物件下一個值的預存程序定義：

```
CREATE SEQUENCE seq_idu START WITH 1 INCREMENT BY 1 NOMINVALUE
NOMAXVALUE ORDER;


CREATE PROCEDURE sp_idu(io_idu IN OUT INTEGER) IS BEGIN IF (io_idu IS
NULL) THEN SELECT seq_idu.nextval INTO io_idu FROM DUAL; END IF; END
sp_idu;
```

1.4.8 觸發

資料庫觸發是與表格相關的程式邏輯，會在某些特定條件下執行。滿足執行準則時，會引發觸發。

觸發在資料庫中建立副作用時很有用。在 Driver for JDBC 的網路位置中，觸發有助於擷取事件發行。下列爲 usr 表格上之資料庫觸發的範例。

```
CREATE TABLE usr ( idu   INTEGER, fname VARCHAR2(64), lname
VARCHAR2(64) );


-- t = trigger; i = insert CREATE TRIGGER t_usr_i AFTER INSERT ON usr
FOR EACH ROW

BEGIN UPDATE usr SET fname = 'John'; END;
```

針對具有觸發針對表格執行陳述式時，如果陳述式滿足觸發中指定的條件，則觸發會引發。例如，使用上面的表格，假設執行下列 insert 陳述式：

```
INSERT INTO usr(lname) VALUES('Doe')
```

執行 insert 陳述式後會引發觸發 t_emp_i，且還會執行下列 update 陳述式：

```
UPDATE usr SET fname = 'John'
```

觸發通常可以在觸發陳述式的前或後引發。一般而言，資料庫觸發時所執行的陳述式是包含在與觸發陳述式相同的異動中。在上面的範例中，INSERT 和 UPDATE 陳述式都會一起認可或復原。

1.4.9 取代觸發

取代觸發是在某些特定條件下才執行，並與檢視窗相關的程式邏輯。

取代觸發有助於讓檢視窗可寫入或可訂閱。它們經常用於定義檢視窗中 INSERT、UPDATE 和 DELETE 的含義。下列爲 usr 表格上之取代觸發的範例。

```
CREATE TABLE usr ( idu   INTEGER, fname VARCHAR2(64), lname
VARCHAR2(64) );



CREATE VIEW view_usr ( pk_idu, fname, lname ) AS SELECT idu, fname,
lname from usr; -- t = trigger; i = insert CREATE TRIGGER t_view_usr_i
INSTEAD OF INSERT ON usr BEGIN INSERT INTO usr(idu, fname, lname)
VALUES(:NEW.pk_idu, :NEW.fname, :NEW.lname); END;
```

針對具有取代觸發的檢視窗執行陳述式時，如果陳述式滿足觸發中指定的條件，則會執行取代觸發。與觸發不同，取代觸發一律在觸發陳述式之前執行。同時，與一般觸發不同的是，取代觸發的執行是用來取代觸發陳述式，而非同時執行。

例如，使用上面的檢視窗，假設執行下列 insert 陳述式而非原始的 insert 陳述式：

```
INSERT INTO view_usr(pk_idu, fname, lname) VALUES(1, 'John', 'Doe')
```

取代觸發 t_view_usr_i 會引發並執行下列陳述式，而非執行原始陳述式：

```
INSERT INTO usr(idu, fname, lname) VALUES(:NEW.pk_idu, :NEW.fname,
:NEW.lname);
```

在此範例中，陳述式恰好相等。

1.5 資料同步化模型

驅動程式支援兩種資料同步化模型：直接和間接。如果從所同步化之資料的最終目的地方面考慮，則會更加準確地理解這兩個詞彙。

模型	關聯	描述
直接	通常與檢視窗相關聯	檢視窗提供最便於與現有客戶表格進行整合的抽象機制。
間接	通常與表格相關聯	客戶表格可能與驅動程式所需的結構不相符。因此，通常需要建立與驅動程式所需之結構相符的中介階段表格。雖然結構可能相符，但是這種情況極少出現。

下列各節描述直接和間接同步化如何在「訂閱者」和「發行者」通道上運作。

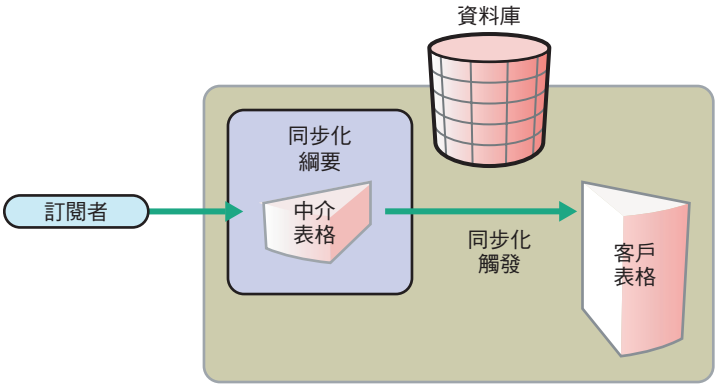
1.5.1 間接同步化

間接同步化使用中介階段表格，來同步化 Identity Vault 與資料庫之間的資料。

下列圖表描述間接同步化在「訂閱者」和「發行者」通道上運作的方式。在下列案例中，您可以具有一或多個客戶表格和中介階段表格。

訂閱者通道

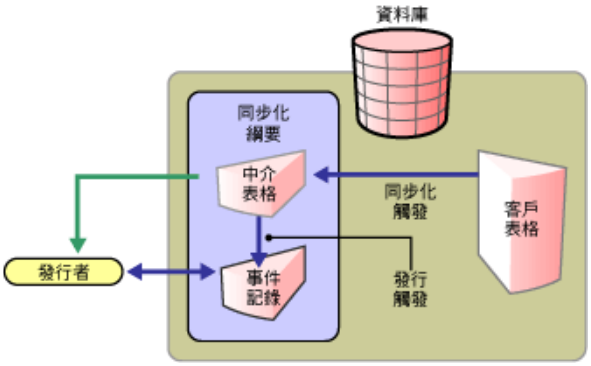
特性 ***1-2*** 訂閱者通道上的間接同步化



「訂閱者」通道會更新同步化綱要中的中介階段表格。然後，同步化觸發會更新資料庫中其他位置的客戶表格。

發行者通道

特性 ***1-3*** 發行者通道上的間接同步化



更新客戶表格時，同步化觸發會更新中介階段表格。然後，「發行」觸發會將一或多個列插入事件記錄表格中。「發行者」通道會讀取已插入的列並更新 Identity Vault。

依據從事件記錄表格讀取的列內容，「發行者」通道可能需要在更新 Identity Vault 之前，從中介表格取回其他資訊。更新 Identity Vault 之後，「發行者」通道會刪除列，或者將其標示爲已處理。

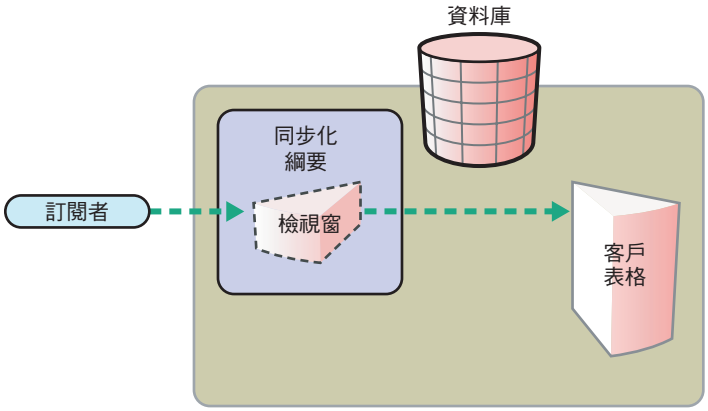
1.5.2 直接同步化

直接同步化通常會使用檢視窗，來同步化 Identity Manager 與資料庫之間的資料。如果表格與 Driver for JDBC 所需的結構一致，則您可以使用它們。

下列圖表描述直接同步化在「訂閱者」和「發行者」通道上運作的方式。在下列案例中，您可以具有一或多個客戶檢視窗或表格。

訂閱者通道

特性 ***1-4*** 訂閱者通道上的直接同步化

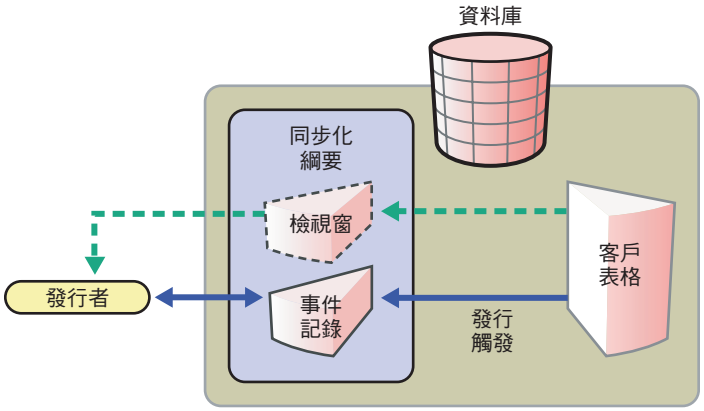


「訂閱者」通道會透過同步化綱要中的檢視窗更新現有的客戶表格。

附註：只有在客戶表格與 Driver for JDBC 所需的結構相符時，無檢視窗的直接同步化才可用。如需其他資訊，請參閱「間接同步化」，第 77 頁。

發行者通道

特性 ***1-5*** 發行者通道上的直接同步化



更新客戶表格時，發行觸發會將列插入事件記錄表格中。「發行者」通道會讀取已插入的列並更新 Identity Vault。

依據從事件記錄表格讀取的列內容，「發行者」通道可能需要在更新 Identity Vault 之前，從檢視窗取回其他資訊。更新 Identity Vault 之後，「發行者」通道會刪除列，或者將其標示爲已處理。

1.6 無觸發與觸發的發行

記錄發行事件不再需要觸發。在無法使用觸發來擷取精確事件時，「發行者」通道可以藉由檢查資料庫資料，來衍生資料庫變更。

當支援合約禁止對資料庫應用程式表格使用觸發，或進行快速建立原型時，無觸發發行特別有用。

無觸發發行的效率低於觸發的發行。利用觸發的發行，變更的內容是已知的。利用無觸發發行，必須先計算變更才能處理事件。

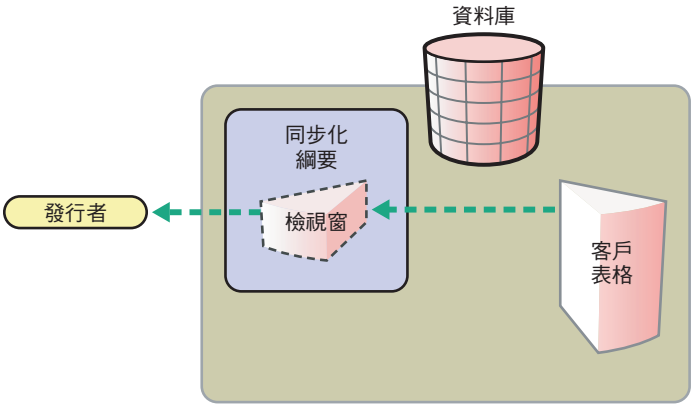
與觸發的發行不同，無觸發發行不會保留事件順序。它只會保證在輪詢週期結束時，資料庫與 Identity Vault 中物件的同步化。

與觸發的發行不同，無觸發發行不提供歷程資料，例如，舊值。它會提供物件現行狀態 (而非先前狀態) 的資訊。

無觸發發行會減少資料庫端的相依性，因此它更加簡便。撰寫資料庫觸發會很複雜，且需要具有資料庫特定 SQL 語法的大量知識。

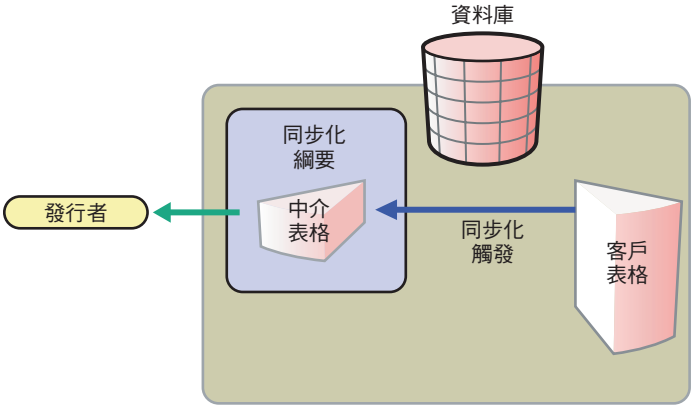
下圖說明直接無觸發發行：

特性 ***1-6*** 直接無觸發同步化



下圖說明間接無觸發發行：

特性 ***1-7*** 間接無觸發同步化



安裝驅動程式之前

2

2.1 驅動程式先決條件

Identity Manager Driver for JDBC 需要下列項目：

- ❑ 安裝在伺服器上的 Novell® iManager 2.5 或更新版本
- ❑ 安裝在伺服器上的 Novell Identity Manager 3
- ❑ Java 虛擬機器 (JVM*) 1.4 或更新版本
- ❑ 受支援的協力廠商 JDBC 驅動程式

2.2 受支援的平台、資料庫和驅動程式

驅動程式可在所有啓用 Identity Manager 的平台上執行，平台包含 Windows* NT*/2000、NetWare® 、Solaris*、Linux* 和 AIX*。

如需支援之資料庫的相關資訊，請參閱 「資料庫互通」，第 123 頁。

如需受支援之協力廠商 JDBC 驅動程式的相關資訊，請參閱 「協力廠商 JDBC 驅動程式互通性」，第 107 頁。

2.3 已知問題

- Identity Vault 的「時間」和「時戳」語法不足以表示其資料庫對應部份的範圍和規模。因爲與資料庫時間相關的類型通常有更廣的範圍和更大的規模 (通常爲毫微秒)，所以此爲發行問題。但相反情況則不成立。如需相關資訊，請參閱 「時間語法」，第 42 頁。
- Driver for JDBC 無法剖析專屬資料庫時戳格式。部份資料庫 (例如，Sybase* 和 DB2*) 具有 java.sql.Timestamp (http://java.sun.com/j2se/1.5.0/docs/api/java/sql/Timestamp.html) 類別無法剖析的專屬時戳格式。在從這些資料庫同步化時戳欄時，在預設狀態下，Driver for JDBC 假設事件記錄表格中放置的時戳值具有 ODBC 規範格式 (即，yyyy-mm-dd hh:mm:ss.ffffffff)。若要使 Driver for JDBC 可以處理專屬資料庫時戳格式，建議方法爲實作自定 DBTimestampTranslator 類別。此介面記錄在與 Driver for JDBC 隨附的「Javadoc 工具」中。使用此方法可避免在將資料庫中的時戳插入事件記錄表格，或在樣式表中重新格式化它們之前，出現重新格式化時戳的問題。Driver for JDBC 隨附原始 DB2 時戳格式和 Sybase 樣式 109 時戳格式的預設實作。
- 可能會無限期封鎖針對資料庫伺服器執行的陳述式。

  通常，封鎖是因獨佔方式鎖定資料庫資源而引起的。因爲鎖定機制和鎖定 SQL 會依資料庫而不同，所以此問題的一般解決方案是實作自定 DBLockStatementGenerator 類別。如需相關資訊，請參閱 「鎖定陳述式產生器類別」，第 54 頁。Driver for JDBC 隨附有 Oracle 的預設實作。

引起封鎖的因素有很多。若要降低封鎖的可能性，建議您不要將 異動隔離層級 參數的層級設爲高於 read committed。

JDBC 介面定義的方法 java.sql.Statement.setQueryTimeout(int):void (http://java.sun.com/j2se/1.5.0/docs/api/java/sql/Statement.html) 可讓陳述式在指定秒數後逾時。可惜，在協力廠商 JDBC 驅動程式之間實作此方法不是無法實作就是有錯誤。因此，此方法不適於充當一般目的的解決方案。

2.4 限制

- Driver for JDBC 不支援使用分隔 (引號) 資料庫識別碼 (例如，" 含有空格的名稱 ")。
- 不支援 JDBC 2 資料類型，但大型物件 (Large Object，LOB) 資料類型除外，例如 CLOB 和 BLOB。
- 不支援 JDBC 3 資料類型。
- PostgreSQL 不支援 <check-object-password> 事件。您可以手動將項目插入 pg_hba.conf 檔案來控制驗證。

安裝或升級 Driver for JDBC

3

如需解除安裝驅動程式的相關資訊，請參閱第 9 章 「解除安裝 IDM Driver for JDBC」, 第 137 頁

重要：我們建議您將驅動程式組態和資料庫程序檔當作一體來進行安裝或解除安裝。爲了防止出現無意的不相符狀況，資料庫程序檔和驅動程式組態的標題包含版本編號、目標資料庫名稱和資料庫版本。

3.1 升級至 Identity Manager 3

Identity Manager Driver for JDBC 2.1 不會在 Identity Manager 3.0 之前版本的 Identity Manager 上執行。若要使用 Driver for JDBC 2.1，您必須升級至 Identity Manager 3。

Identity Manager Driver for JDBC 2.0 可以在 Identity Manager 2 上執行。

在 Identity Manager 安裝期間，您可以在安裝 Metadirectory 引擎的同時，安裝 Driver for JDBC (與其他 Identity Manager 驅動程式一起)。請參閱《*Identity Manager 3.0 安裝指南*》。您可以從 DirXML 1.1a 或 Identity Manager 2 升級至 Identity Manager 3。

3.2 放置 Jar 檔案

下表指出在 Identity Manager 或「遠端載入器」伺服器放置 JDBC 驅動程式 jar 檔案的路徑 (假設 Identity Manager 和「遠端載入器」均使用預設安裝路徑)。

3.2.1 Identity Manager 檔案路徑

下表指出依平台在「身份管理」伺服器上放置 JDBC 驅動程式 jar 檔案的位置。

表格 ***3-1*** *jar* 檔案的位置：*Identity Manager* 伺服器

平台	目錄路徑
NetWare®	sys:\system\lib
Solaris、Linux 或 AIX	/usr/lib/dirxml/classes (eDirectory 8.8 之前的版本) /opt/novell/eDirectory/lib/dirxml/classes (eDirectory 8.8)
Windows NT/2000	novell\NDS\lib

3.2.2 遠端載入器檔案路徑

下表指出依平台在「遠端載入器」伺服器上放置 JDBC 驅動程式 jar 檔案的位置。

表格 *3-2* *jar* 檔案的位置：遠端載入器

平台	目錄路徑
Solaris、Linux 或 AIX	/usr/lib/dirxml/classes (eDirectory 8.8 之前的版本) /opt/novell/eDirectory/lib/dirxml/classes (eDirectory 8.8)
Windows NT/2000	novell\RemoteLoader\lib

3.3 安裝 Driver for JDBC

3.3.1 安裝驅動程式

您可以在安裝 Metadirectory 引擎的同時，安裝 Driver for JDBC (與其他 Identity Manager 驅動程式一起)。請參閱《 *Identity Manager 3.0* 安裝指南》。

您也可以在安裝 Metadirectory 引擎之後，單獨安裝驅動程式。

安裝至 **Windows**

1. 執行 Identity Manager 3 下載影像檔或 CD 中的安裝程式 (\nt\install.exe)。

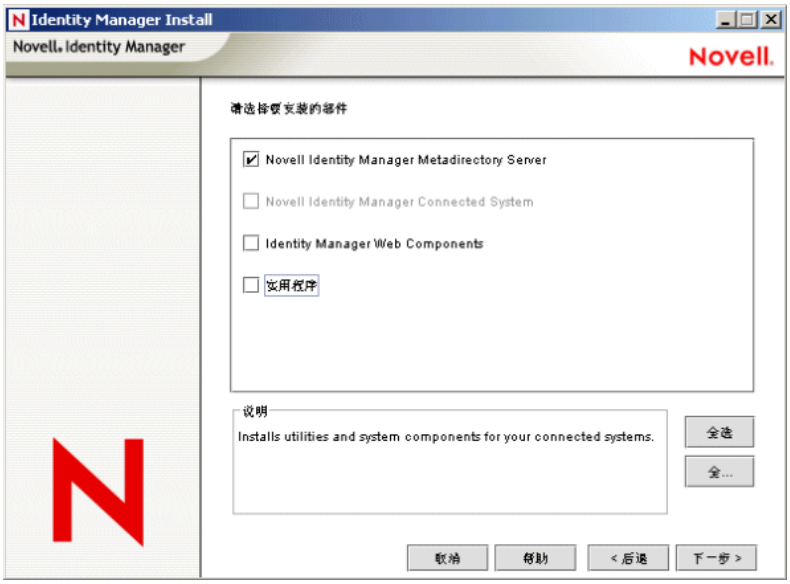
   您可由 Novell 下載 (http://download.novell.com/index.jsp) 取得下載。
2. 在「歡迎」對話方塊中，按「下一步」，然後接受授權合約。
3. 在第一個「Identity Manager 概觀」對話方塊中，檢視資訊，然後按「下一步」。

   對話方塊會提供下列資訊：
   - Metadirectory 伺服器
   - 已連接伺服器系統
4. 在第二個「Identity Manager 概觀」對話方塊中，檢視資訊，然後按「下一步」。

   對話方塊會提供下列資訊：
   - Web 型態的管理伺服器

- Identity Manager 公用程式

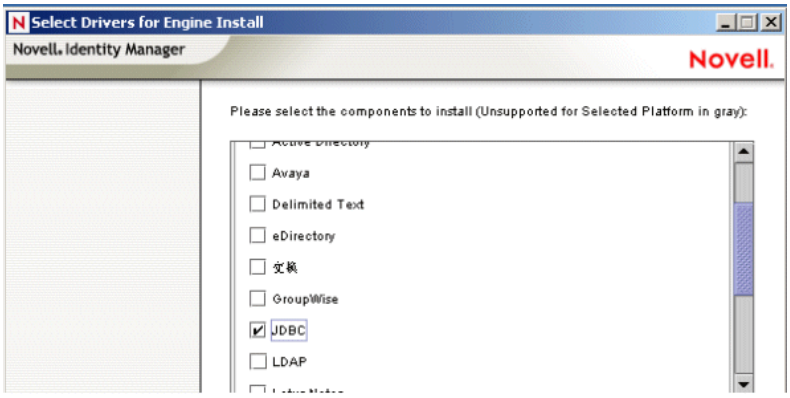
5 如果您在本地安裝，則只選取「*Metadirectory* 伺服器」，然後按「下一步」。



如果您在遠端安裝 (遠端載入器)，則選取「已連接系統」並參閱《*Novell Identity Manager 3.0* 管理指南》中的「設定遠端載入器」和「設定已連接系統」。

如果您安裝「遠端載入器」，則規則 (和規則所參考的二進位碼) 會在本地執行，但驅動程式 Shim 二進位碼會在遠端執行。如果您安裝「Metadirectory 伺服器」，則所有二進位碼和規則都會在本地執行。

6 在「選取要安裝的引擎驅動程式」對話方塊中，只選取「*JDBC*」，然後按「下一步」。



7 在「Identity Manager 升級警告」對話方塊中，按一下「確定」。

8 在「摘要」對話方塊中，檢視所選取的選項，然後按一下「完成」。

9 在「安裝完成」對話方塊中，按一下「關閉」。

程式安裝完成後，請依照「輸入範例組態檔案」，第 28 頁的說明設定驅動程式。

安裝至 **NetWare**

1 在 NetWare® 伺服器上，插入 Identity Manager CD，然後裝上 CD 作爲卷冊。

如果沒有 CD，請下載 Identity_Manager_3_NW_Win.iso，並製作成一片 CD。您可由 Novell 下載 (http://download.novell.com/index.jsp) 取得下載。

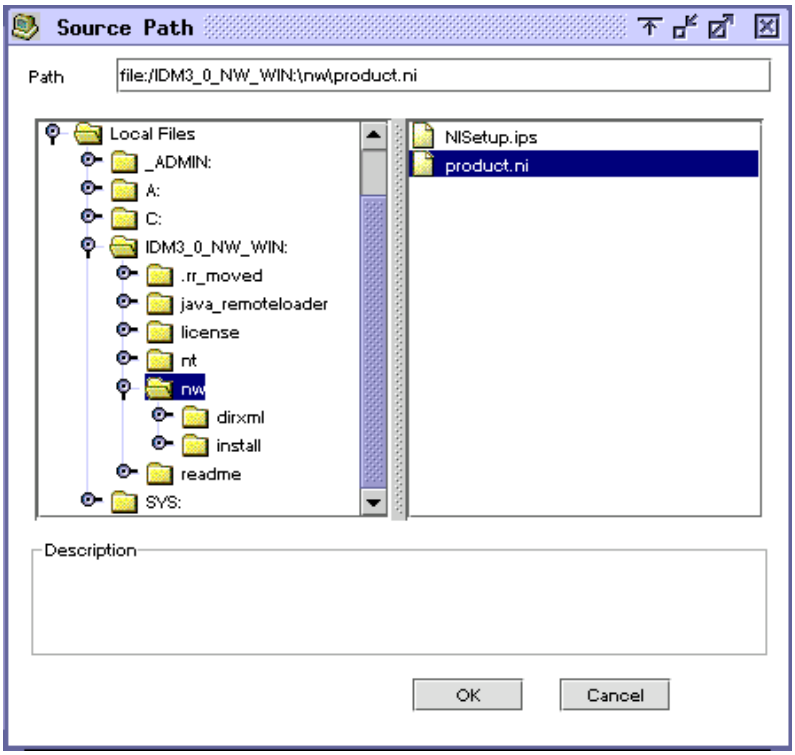
若要裝上 CD，請輸入 m cdrom。

2 (視情況) 若無法載入圖形化公用程式，請輸入 startx 來載入。

3 在圖形化公用程式中，按一下 Novell 圖示，然後按一下「安裝」。

4 在「已安裝產品」對話方塊中，按一下「新增」。

5 在「來源路徑」對話方塊中，瀏覽並選取「product.ni」檔案。



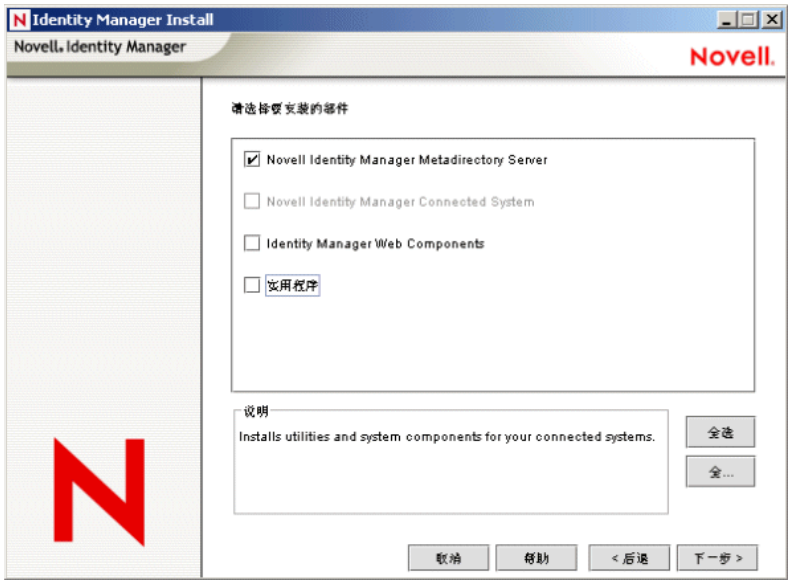
5a 瀏覽並展開您先前已裝上的 CD 卷冊 (IDM_3_0_NW_WIN)。

5b 展開「nw」目錄，選取「product.ni」，然後按兩下「確定」。

6 在「歡迎使用 Novell Identity Manager 3.0 安裝程式」對話方塊中，按「下一步」，然後接受授權合約。

7 檢視兩個「綜覽」對話方塊，然後按「下一步」。

8 在「安裝 Identity Manager」對話方塊中，只選取「*Metadirectory* 伺服器」。

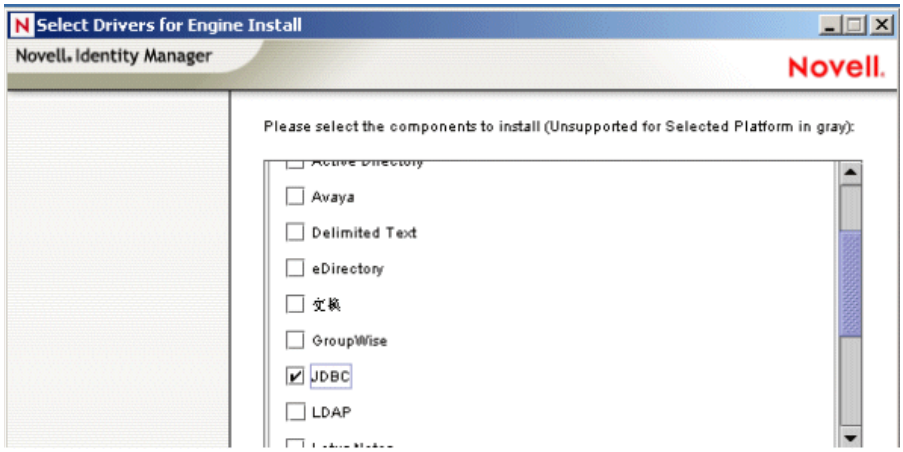


取消選取下列選項：

- Identity Manager Web 元件
- 公用程式

9 按「下一步」。

10 在「選取要安裝的引擎驅動程式」對話方塊中，只選取「*JDBC*」。



取消選取下列選項：

- Metadirectory 引擎
- 除了「分隔文字」之外的所有驅動程式

11 按「下一步」。

12 在「Identity Manager 升級警告」對話方塊中，按一下「確定」。

對話方塊建議您在 90 天內啓用驅動程式的授權。

13 在「摘要」頁面中，檢視所選取的選項，然後按一下「完成」。

14 按一下「關閉」。

安裝之後，請進行下列幾項操作：

- 輸入範例組態檔案。請參閱 「輸入範例組態檔案」，第 28 頁。
- 設定「遠端載入器」(選擇性)。請參閱 「設定遠端載入器」，第 30 頁。
- 設定資料庫物件的組態。請參閱 「安裝並設定資料庫物件」，第 30 頁。

安裝至 Linux 或 Solaris

在預設狀態下，當您在安裝 Metadirectory 引擎時，Identity Manager Driver for JDBC 就已經安裝完成。若當時驅動程式未安裝完成，本節可以協助您進行安裝。

執行安裝程式的過程中，輸入「previous」即可回到上一個步驟 (畫面)。

1 在終端機會期中，以根部身份登入。

2 請插入 Identity Manager CD，然後將其裝上。

如果沒有 CD，請下載 Identity_Manager_3_Linux.iso，並製作成一片 CD。您可由 Novell 下載 (http://download.novell.com/index.jsp) 取得下載。

一般而言，CD 會自動裝上。下表列出手動裝上 CD 的範例：

平台	輸入的內容
AIX	mount /mnt/cdrom，然後按 Enter
Red Hat*	mount /mnt/cdrom，然後按 Enter
Solaris	mount /cdrom，然後按 Enter
SUSE®	mount /media/cdrom，然後按 Enter

3 變更至設定目錄。

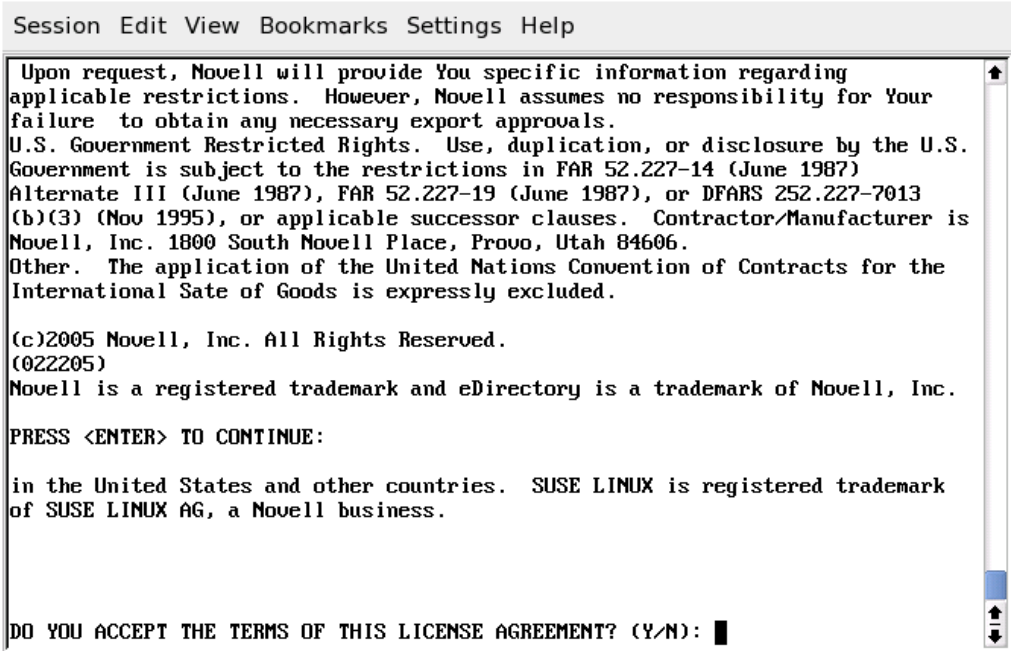
平台	路徑
AIX	/mnt/cdrom/setup/
Red Hat	/mnt/cdrom//setup/
Solaris	/cdrom//idm_3/setup/
SUSE	/media/cdrom//setup/

4 輸入 ./dirxml_linux.bin 執行安裝程式。

5 在「簡介」區段中，按 Enter。

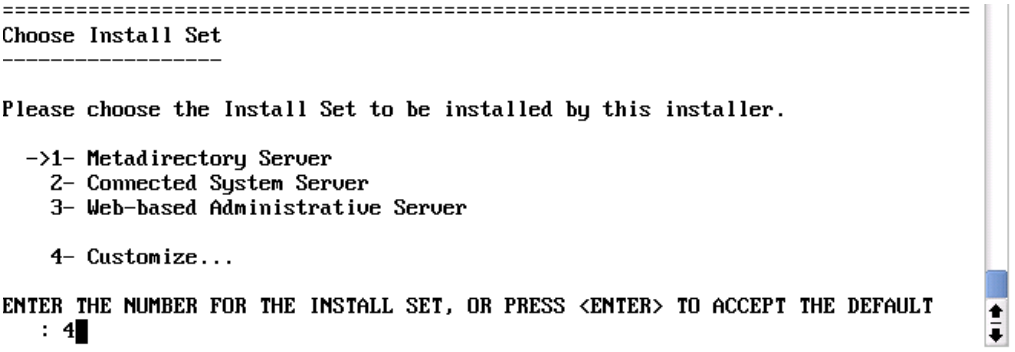
6 接受授權合約。

按著 Enter，直到顯示「您是否接受授權合約條款」後，輸入 y，然後按 Enter。



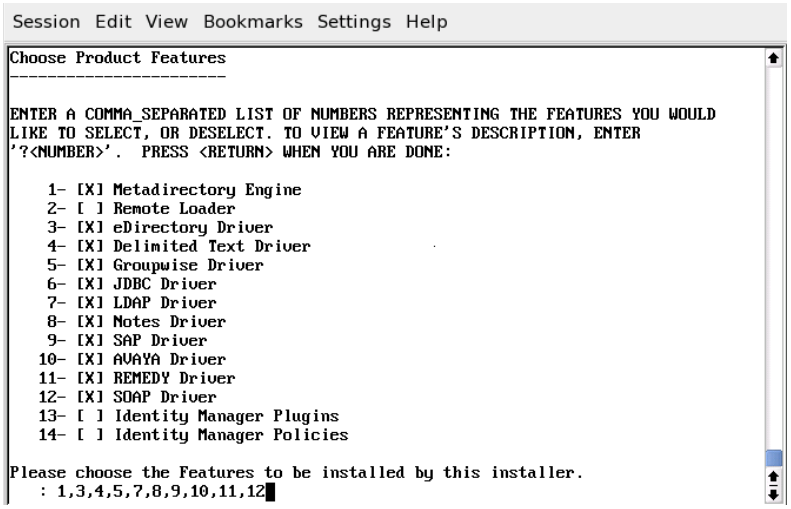
7 在「選擇安裝集」區段中，選取「自定」選項。

輸入 4，然後按 Enter。

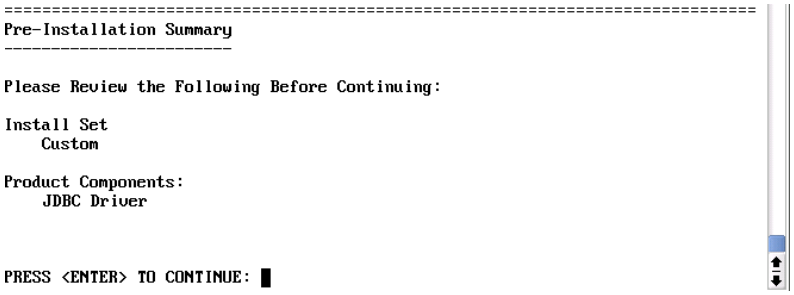


8 在「選擇產品功能」區段中，請取消選取除了 JDBC 以外的所有功能，然後按 Enter。

若要取消選取某項功能，請輸入該功能之編號。在您取消選取的其他功能之間輸入逗號。



9 在「預先安裝摘要」區段中，檢視選項。



若要返回上一個區段，請輸入「previous」，然後按 Enter。

若要繼續，請按 Enter。

10 安裝完成後，按 Enter 結束安裝。

程式安裝完成後，設定驅動程式組態。請參閱第 4 章 「設定 Identity Manager Driver for JDBC 組態」, 第 37 頁。

3.3.2 輸入範例組態檔案

若要設定 Identity Manager Driver for JDBC，請輸入驅動程式組態檔案，並設定資料庫組態。資料庫組態包含執行 SQL 程序檔。建議您執行資料庫 SQL 程序檔，並在啓動驅動程式之前測試它們。

隨附的組態僅是一個範例。建議您在嘗試自定組態之前，先將隨附的組態安裝至測試環境。

輸入驅動程式組態檔案：**iManager**

JDBCv2.xml 組態檔案會建立範例驅動程式正常運作所需的 Identity Manager 物件，並設定其組態。組態檔案還包含您可以自定的範例規則。

1 在 iManager 中，選取「*Identity Manager* 公用程式」>「新驅動程式」。

2 選取驅動程式集，然後按 「下一步」。

如果您將此驅動程式置於新的驅動程式集中，則請指定驅動程式集名稱、網路位置和相關聯的伺服器。

3 選取 「從伺服器 *(.XML* 檔案 *)* 輸入驅動程式組態」。

當您設定 iManager 時，驅動程式組態檔就會安裝在 Web 伺服器上。

4 從下拉式清單中，選取 「*JDBCv2.xml*」選項，然後按 「下一步」。

5 系統提示您輸入驅動程式名稱時，指定驅動程式的名稱 (例如，JDBC 2)，然後按 「下一步」。

6 選取目標資料庫，選取驅動程式是本地還是遠端，然後按 「下一步」。

7 選取同步化模型，選取協力廠商 JDBC 實作，然後按 「下一步」。

8 選取資料流程 (例如，雙向)，指定資料庫主機 IP 位址，輸入連接埠號碼，然後按 「下一步」。

9 指定 「使用者」容器 DN、「群組」容器 DN 和發行模式，然後按 「下一步」。

10 (選用性) 按一下 「定義安全性等值」。

10a 按一下 「新增」，然後選取具有 「管理員」權限 (或您想要驅動程式具有的任何其他權限) 的物件。

10b 按一下 「套用」，然後按一下 「確定」。

11 (選用性) 若要將物件從複製中排除，請按一下 「排除管理角色」。

11a 按一下 「新增」，然後選取您想要排除的使用者 (例如管理員使用者)。

11b 按一下 「套用」，然後按一下 「確定」。

12 若要檢視輸入摘要，請按 「下一步」。

13 驗證組態是否正確，然後按一下 「完成概觀」。

安裝過程會建立必要的 Identity Manager 驅動程式物件。如果您沒有在輸入時定義安全性等值或排除管理使用者，則可以藉由修改驅動程式物件的內容完成這些任務。

組態檔案慣例

- 資料庫使用者名稱是使用者的姓氏加上對應的數值主索引鍵值。例如，John Doe 的使用者名稱可能是 Doe1。
- 啓始密碼是使用者的姓氏。例如，John Doe 的密碼可能是 Doe。Sybase 密碼至少必須爲 6 個字元。少於 6 個字元時，姓氏會使用字元 "p" 填補。例如，John Doe 的密碼可能是 Doeppp。填補的字元可以在 「訂閱者指令轉換」規則中調整。

輸入驅動程式組態檔案：**Designer**

您可以使用 Designer for Identity Manager 來輸入 JDBC 的基本驅動程式組態檔案。此基本檔案會建立驅動程式正常運作所需的物件和規則並設定其組態。

以下程序說明了其中一種輸入範例組態檔案的方式：

1 在 Designer 中開啓專案。

2 在模擬器中，在 「驅動程式集」物件上按一下滑鼠右鍵，然後選取 「新增已連接應用程式」。

3 從下拉式清單中選取 「*JBDC.xml*」，然後按一下 「執行」。

4 在「執行提示驗證」視窗中，按一下「是」。

5 利用填入欄位來設定驅動程式。

請指定您環境的特定資訊。如需這些設定的相關資訊，請參閱 「組態參數」，第 39 頁。

6 指定參數後，請按一下「確定」來輸入驅動程式。

7 自定及測試驅動程式。

8 部署驅動程式至 Identity Vault。

請參閱《*Designer for Identity Manager 3*：管理指南》中的「部署專案至 Identity Vault」。

3.3.3 設定遠端載入器

您可以選擇性使用「遠端載入器」。除非您想要 JDBC 驅動程式在已連接的系統中執行，否則不需要它。

1 如果尚未安裝「遠端載入器」，請安裝。

請參閱《*Novell Identity Manager 3.0* 管理指南》中的「設定已連接系統」。

2 將適當的協力廠商 JDBC 驅動程式 jar 檔案複製到「遠端載入器」伺服器上。

2a 如需協力廠商 JDBC 驅動程式檔名及其取得位置的相關資訊，請參閱 「受支援的協力廠商 JDBC 驅動程式」，第 109 頁。

2b 如需檔案安裝路徑的相關資訊，請參閱 「放置 Jar 檔案」，第 21 頁。

3 設定遠端驅動程式的組態。

在「遠端驅動程式組態」參數中，將「驅動程式」參數設爲

```
com.novell.nds.dirxml.driver.jdbc.JDBCDriverShim.
```

4 設定其他遠端載入器參數的組態 請參閱《*Novell Identity Manager 3.0* 管理指南》中的「設定已連接系統」。

3.3.4 安裝並設定資料庫物件

安裝資料庫物件 (例如，表格、觸發和索引)，並設定它們的組態，以與範例驅動程式組態同步化。如果您沒有設定資料庫物件的組態，則範例組態檔案不會運作。

SQL 程序檔慣例

SQL 程序檔位於安裝目錄 \jdbc\sql\ 縮寫的資料庫名稱目錄中。

所有 SQL 程序檔都使用相同的慣例，而無論資料庫爲何。

DB2 識別碼的大小上限是 18 個字元。這個最小公分母長度定義了所有 SQL 程序檔中資料庫識別碼長度的上限。因爲此長度的限制，所以會使用縮寫。下表概述識別碼縮寫和它們的意義：

表格 ***3-3*** 識別碼縮寫和意義

縮寫	解釋
proc_[1]	預存程序 / 函數
idx_	索引
trg_	觸發
_i	插入時觸發
_u	更新時觸發
_d	刪除時觸發
chk_	檢查條件約束
pk_	檢視主索引鍵條件約束
fk_	檢視外部索引鍵條件約束
mv_	檢視多值欄
sv_	檢視單一值欄 (隱含預設值)

1 最常用的縮寫是 sp_。此字首會保留給 Microsoft* SQL Server 上的系統預存程序。同時，在評估任何修飾詞 (例如，資料庫或擁有者) 之前，此字首會強制先在主要資料庫中查閱程序。爲了大幅提升程序查閱效率，已刻意避免使用此字首。

下表指出索引、觸發、預存程序、函數和條件約束的識別碼命名慣例：

表格 ***3-4*** 識別碼命名慣例

資料庫物件	命名慣例	範例
預存程序 / 函數	proc_ *程序或函數名稱*	proc_idu
索引	idx_ *未經資格修飾表格名稱* _ *序列號碼*	idx_indirectlog_1
觸發	tgr_ *未經資格修飾表格名稱* _ *觸發陳述式類型* _ *序列號碼*	tgr_usr_i_1
主索引鍵條件約束	pk_ *未經資格修飾表格名稱* _ *欄位名稱*	pk_usr_idu
外部索引鍵條件約束	fk_ *未經資格修飾表格名稱* _ *欄位名稱*	fk_usr_idu
檢查條件約束	chk_ *未經資格修飾表格名稱* _ *欄位名稱*	chk_usr_idu

其他慣例：

- 所有的資料庫識別碼都是小寫的。

  這是資料庫間最常用的大小寫慣例。
- 字串欄位長度是 64 個字元。

  此長度的欄位可以容得下大部份 eDirectory™ 屬性值。若要增加儲存效率，建議您修改欄位長度。

- 基於效能考量，主索引鍵欄會儘可能使用原始純量數值類型 (例如 BIGINT 而不是 NUMERIC)。
- 事件記錄表格中的 record_id 欄具有每個資料庫允許的最大數值精確度以避免溢位。
- 身份欄和序列物件不會快取值。部份資料庫會在復原發生時丟棄快取值。此動作可能會導致身份欄或序列值不連續。

安裝 **IBM DB2 Universal Database (UDB)**

重要：針對 IBM* DB2，您必須在執行提供的 SQL 程序檔之前，手動建立作業系統使用者帳戶。

因爲不同的作業系統建立使用者帳戶的程序不同，所以下面的步驟 1 會因作業系統而異。這些指示適用於 Windows NT 作業環境。如果重新執行 SQL 程序檔，則僅需重複步驟 2 至 5。

DB2 的目錄網路位置是安裝目錄 \jdbc\sql\db2_udb\install

1. 爲使用者 idm、indirect 和 direct 建立使用者帳戶。

   使用 novell 做爲 「網域使用者管理員」中的密碼。

   對於此帳戶，請記得不選 「使用者必須在下一次登入時變更密碼」。

   建議您也選取 「密碼永不過期」。

   附註：剩下的指示與作業系統無關。

2. 將檔案路徑調整爲 1_install.sql 安裝程序檔中的 idm_db2.jar。idm_db2.jar 的檔案路徑應該反映用戶端機器上此檔案的位置。
3. 從指令行處理器 (Command Line Processor，CLP) 執行 1_install.sql 程序檔。

   例如：db2 -f 2_install_8.sql

   重要：程序檔不會在第 7 版以上的 「指令中心」介面上執行。程序檔會使用 僅 ' 行接續字元。「指令中心」的更新版本無法辨識此字元。

4. 針對第 8 版或更新的版本，執行 2_install_8.sql 程序檔。

   例如：db2 -f 2_install_8.sql

安裝 **Informix Dynamic Server (IDS)**

重要：針對 Informix* Dynamic Server，您必須在執行提供的 SQL 程序檔之前，手動建立作業系統使用者帳戶。

因爲不同的作業系統建立使用者帳戶的程序不同，所以下面的步驟 1 會因作業系統而異。這些指示適用於 Windows NT 作業環境。如果重新執行 SQL 程序檔，則應該僅重複步驟 2 至 4。

Informix SQL 程序檔的目錄網路位置是 安裝目錄 \jdbc\sql\informix_ids\install。

1. 在 Windows NT 中，爲使用者 idm 建立使用者帳戶。

   使用 novell 做爲 「網域使用者管理員」中的密碼。

對於此帳戶，請記得不選「使用者必須在下一次登入時變更密碼」。

建議您也選取「密碼永不過期」。

附註：剩下的指示與作業系統無關。

2 啓動用戶端，例如「SQL 編輯器」。

3 以 informix 使用者或具有資料庫管理員 (Database Administrator，DBA) 權限的其他使用者身份登入伺服器。

在預設狀態下，informix 使用者的密碼是 informix。

附註：如果您以 informix 之外的使用者身份執行程序檔，請在執行之前變更程序檔中 informix 的所有參考。

4 依據您要建立之資料庫類型的不同，從 ansi (執行屬性、ANSI 相容)、log (執行屬性、非 ANSI 相容) 或 no_log (非執行屬性、非 ANSI 相容) 子目錄，開啓並執行 1_install.sql。

安裝 **Microsoft SQL Server**

Microsoft SQL Server 程序檔的目錄網路位置是 安裝目錄 \jdbc\sql\mssql\install。

1 啓動用戶端，例如「查詢分析器」。

2 以 sa 使用者身份登入資料庫伺服器。

在預設狀態下，sa 使用者沒有密碼。

3 執行安裝程序檔。

若爲第 7 版，執行 1_install_7.sql。

若爲第 8 版 (2000)，執行 1_install_2k.sql。

附註：「查詢分析器」中的執行快速鍵是 F5。

安裝 **MySQL**

MySQL* SQL 程序檔的目錄網路位置是 安裝目錄 \jdbc\sql\mysql\install。

1 從 MySQL 用戶端 (例如 mysql)，以根部使用者或具有管理權限之其他使用者的身份登入。

例如，從指令行，執行

mysql -u root -p

在預設狀態下，root 使用者沒有密碼。

2 依據要使用之表格類型的不同，執行安裝程序檔 1_install_innodb.sql 或 1_install_myisam.sql。

例如：mysql> \. c:\1_install_innodb.sql

提示：請勿使用分號終止此陳述式。

安裝 **Oracle**

Oracle SQL 程序檔的目錄網路位置是 安裝目錄 \jdbc\sql\oracle\install。

1. 從 Oracle 用戶端 (例如 SQL Plus)，以 SYSTEM 使用者身份登入。

   在預設狀態下，SYSTEM 的密碼是 MANAGER。

   附註：如果以 SYSTEM 之外的使用者身份使用密碼 MANAGER 執行程序檔，則在執行之前變更程序檔中 SYSTEM 的所有參考。

   ---

2. 執行安裝程序檔 1_install.sql。

   例如：SQL> @c:\1_install.sql

安裝 **PostgreSQL**

PostgreSQL 程序檔的目錄網路位置是 安裝目錄 \jdbc\sql\postgres\install。執行 Postgres 指令的目錄網路位置是 postgres-install-dir/pgsql/bin。

1. 建立資料庫 idm。

   例如，從 UNIX* 指令行，執行指令 createdb：./createdb idm

2. 將 plpgsql 程序語言安裝至資料庫 idm。

   例如，從 UNIX 指令行，執行指令 createlang：./createlang plpgsql idm

3. 從 Postgres 用戶端 (例如 psql)，以使用者 postgres 身份登入 idm 資料庫。

   例如，從 UNIX 指令行，執行指令 psql：./psql -d idm postgres

   在預設狀態下，Postgres 使用者沒有密碼。

4. 從 psql 內部，執行程序檔 1_install.sql。

   例如：idm=# \i 1_install.sql

5. 更新 pg_hba.conf 檔案。

   例如，針對 idm 資料庫使用者新增項目。視需要調整 IP-ADDRESS 和 IP-MASK：

   ```
   # TYPE  DATABASE    USER   IP-ADDRESS        IP-MASK
   METHOD# allow driver user idm to connect to database idm host
   idm         idm    255.255.255.255   255.255.255.0     password
   ```

6. 重新啟動 Postgres 伺服器，以使對 pg_hba.conf 檔案進行的變更生效。

安裝 **Sybase Adaptive Server Enterprise (ASE)**

重要：請確定您已在資料庫伺服器上安裝 JDBC 中繼資料支援。通常 12.5 版之前的版本才會有此問題。

Sybase SQL 程序檔的目錄網路位置是 安裝目錄 \jdbc\sql\sybase_ase\install。

1. 從 Sybase 用戶端 (例如 isql)，以 sa 使用者身份登入，並執行 1_install.sql 安裝程序檔。

   例如，從指令行，執行：isql -U sa -P -i 1_install.sql

   在預設狀態下，sa 帳戶使用者沒有密碼。

3.3.5 測試

每個資料庫的測試程序檔都位於下列目錄中：

表格 **3-5** 資料庫程序檔的位置

資料庫	測試 SQL 程序檔位置
IBM DB2 Universal Database	*安裝目錄* \jdbc\sql\db2_udb\test
Informix Dynamic Server	*安裝目錄* \jdbc\sql\informix_ids\log\test *安裝目錄* \jdbc\sql\informix_ids\no_log\test Informix ANSI 測試程序檔位於 log\test 子目錄中。
Microsoft SQL Server	*安裝目錄* \jdbc\sql\mssql\test
MySQL	*安裝目錄* \jdbc\sql\mysql\test
Oracle	*安裝目錄* \jdbc\sql\oracle\test
PostgreSQL	*安裝目錄* \jdbc\sql\postgres\test
Sybase Adaptive Server Enterprise	*安裝目錄* \jdbc\sql\sybase_ase\test

建議您在啓動範例驅動程式之前嘗試測試程序檔。

3.3.6 疑難排解

- 除非您明確認可變更，否則「發行者」通道可能無法辨識發行事件。如需受支援之資料庫認可關鍵字的相關資訊，請參閱「認可關鍵字」，第 127 頁。
- 測試程序檔應該由驅動程式之 idm 資料庫使用者帳戶之外的使用者執行。如果以 idm 使用者身份執行它們，則除非允許發行迴路，否則驅動程式的「發行者」通道會忽略事件。如需允許或不允許發行迴路的相關資訊，請參閱「允許迴路？」，第 71 頁。

3.4 升級 Driver for JDBC

- 「反向不相容」，第 36 頁

Identity Manager Driver for JDBC 2.1 不能在 Identity Manager 3.0 之前的版本上執行。Identity Manager Driver for JDBC 2.0 可以在 Identity Manager 2.0 上執行。

若要從 Identity Manager Driver for JDBC 1.5 或更新版本升級至 2.1 版，請安裝 Driver for JDBC。此任務僅取代二進位碼。

表格 **3-6** 升級至 *Identity Manager Driver for JDBC 2.0*

如果正在執行此版本	升級至此版本	升級至此版本之前
Driver for JDBC 1.5 之前的版本	Driver for JDBC 1.51	Driver for JDBC 2.0
Driver for JDBC 1.5 或更新版本	無	IDriver for JDBC 2.0

表格 **3-7** 升級至 *Identity Manager Driver for JDBC 2.1*

如果正在執行此版本	升級至此版本	升級至此版本之前
Driver for JDBC 1.5 之前的版本	Driver for JDBC 1.51	Driver for JDBC 2.1
Driver for JDBC 1.5 或更新版本	無	IDriver for JDBC 2.1

若爲 Identity Manager Driver for JDBC 1.5 之前的版本，您必須先升級至 1.5 版。請參閱《*DirXML Driver 1.5 for JDBC* 實作指南 (http://www.novell.com/documentation/lg/dirxmldrivers/index.html)》。請務必使用 「2.1 關聯公用程式」。它會取代所有先前版本。

3.4.1 反向不相容

- 現在驅動程式最少需要兩個資料庫連接才能進行雙向同步化。如需其他資訊，請參閱 「使用最少的連接？」，第 48 頁。
- 現在驅動程式會傳回邏輯資料庫類別名稱 (父表格或檢視窗名稱) 的綱要修飾詞 (可用時)。除非類別名稱在 「綱要映射」規則中重新映射，否則此變更不會影響現有的組態。如果重新映射類別名稱，則現有規則中所有類別名稱的參考都需要是符合綱要的。
- 略微改變使用檢視窗的現有組態。將參數 「啓用中繼識別碼支援」設爲布林值 False。請參閱 「啓用中繼識別碼支援？」，第 55 頁。
- 略微改變參考 com.novell.nds.dirxml.driver.jdbc.util.MappingPolicy 類別的現有組態。此類別中的方法不再編輯來源文件。然而，它們會傳回必須複製到目的地文件中的節點集。範例驅動程式組態檔案 JDBCv2.xml 包含如何執行此動作的範例。
- 略微改變針對 DB2/AS400 或不實作或支援欄位置之其他舊資料庫部署的現有組態。新增並設定 「欄名稱排序方式」參數。若要按照字串定序順序排序欄名稱，請參閱 「欄名稱排序方式」，第 58 頁。預設行爲已變更爲按照十六進位值排序欄名稱。

3.5 啓用驅動程式

請在安裝後 90 天內啓用驅動程式。否則，驅動程式將停止執行。

如需啓用的相關資訊，請參閱 《*Identity Manager 3.0* 安裝指南》中的 「啓用 Novell Identity Manager 產品」。

4 設定 Identity Manager Driver for JDBC 組態

4.1 智慧型組態

Identity Manager Driver for JDBC 可以辨識一組支援的協力廠商 JDBC 驅動程式和資料庫。同時，驅動程式可以動態地和自動地設定大部份驅動程式相容性參數的組態。這些功能會減緩一般使用者瞭解並明確地設定此類參數的需要。

這些功能會透過下列四種類型的 XML 描述元檔案實作，這些檔案會描述 Driver for JDBC 協力廠商 JDBC 驅動程式或資料庫。

- 協力廠商 JDBC 驅動程式
- 協力廠商 JDBC 驅動程式輸入
- 資料庫
- 資料庫輸入

描述元檔案的保留檔名

與驅動程式一起提供的描述元檔名以底線字元 (_) 開頭。會保留此類檔名，以確保與驅動程式一起提供的描述元檔案不與自定描述元檔案衝突。顯然，自定描述元檔名不能以底線字元開頭。

輸入描述元檔案

輸入描述元檔案允許多個非輸入描述元檔案共享內容。此功能會減小非輸入描述元檔案的大小、將內容重複的需要降至最低，並增強可維護性。輸入檔案無法在主要類型中輸入。也就是說，JDBC 驅動程式描述元無法輸入資料庫輸入，且資料庫描述元無法輸入 JDBC 驅動程式輸入。

此外，自定非輸入描述元無法輸入保留的描述元輸入。例如，如果名爲 custom.xml 的自定協力廠商 JDBC 驅動程式描述元檔案嘗試輸入名爲 _reserved.xml 的保留協力廠商 JDBC 驅動程式描述元，則會發出錯誤。這些限制達到下列效果：

- 確保保留與和自定輸入檔案之間不存在相依性
- 允許現有的保留描述元檔案在驅動程式更新版本中延伸

描述元檔案位置

描述元檔案必須位於 jar 檔案中，該檔名以字首 "jdbc" (區分大小寫) 開頭，且位於執行時期 classpath 中。

下表指出將描述元放置在描述元 jar 檔案中的位置：

表格 ***4-1*** 放置描述元的位置

描述元類型	目錄路徑
協力廠商 JDBC 驅動程式	com/novell/nds/dirxml/driver/jdbc/db/descriptor/driver
協力廠商 JDBC 驅動程式輸入	com/novell/nds/dirxml/driver/jdbc/db/descriptor/driver/import
資料庫	com/novell/nds/dirxml/driver/jdbc/db/descriptor/db
資料庫輸入	com/novell/nds/dirxml/driver/jdbc/db/descriptor/db/import

保留的描述元檔案位於 JDBCConfig.jar 檔案中。爲了確保更新 Driver for JDBC 時未覆寫這些保留的檔案，請將自定描述元置於不同的 jar 檔案中。

優先順序

透過管理主控台 (例如 iManager) 明確指定之參數的優先順序，一律高於透過描述元檔案指定的參數。僅當未透過管理主控台設定參數時，描述元檔案參數才生效。

在不可輸入的描述元檔案中指定之參數和其他資訊的優先順序一律較在描述元輸入檔案中指定的要高。如果在描述元檔案中有重複的參數或其他資訊，則參數或資訊的第一個例項會優先於後續例項。

在輸入檔案之間，優先順序由輸入順序決定。輸入清單中較早宣告的輸入檔案會優先於隨後的那些輸入檔案。

自定描述元最佳作法

- 請勿讓自定描述元檔名以底線 (_) 字元開頭。
- 請將自定描述元檔案置於 JDBCConfig.jar 之外的 jar 檔案，並將檔名以字首 "jdbc" (不區分大小寫) 爲開頭。
- 請勿使用自定描述元來輸入保留的輸入檔案 (檔名以底線字元開頭)。

描述元檔案 DTD

下列附錄包含所有描述元檔案類型的文件類型定義 (Document Type Definition，DTD)。這些文件類型定義 (DTD) 可協助您建構自定描述元檔案。

表格 ***4-2*** 描述元 *DTD* 的位置

描述元類型	附錄
協力廠商 JDBC 驅動程式	附錄 F 「協力廠商 JDBC 驅動程式描述元 DTD」，第 157 頁

描述元類型	附錄
協力廠商 JDBC 驅動程式輸入	附錄 G 「協力廠商 JDBC 驅動程式描述元輸入 DTD」，第 159 頁
資料庫	附錄 H 「資料庫描述元 DTD」，第 161 頁
資料庫輸入	附錄 I 「資料庫描述元輸入 DTD」，第 163 頁

4.2 組態參數

- 「檢視驅動程式參數」，第 39 頁
- 「不建議使用的參數」，第 39 頁
- 「驗證參數」，第 39 頁

4.2.1 檢視驅動程式參數

1. 在 iManager 中，按一下「*Identity Manager*」>「*Identity Manager* 概觀」。
2. 找到包含驅動程式的驅動程式集，然後按一下驅動程式的圖示。
3. 從「Identity Manager 驅動程式概觀」中，按一下驅動程式物件。

   iManager 會顯示驅動程式的組態參數。

4.2.2 不建議使用的參數

自 1.6 版開始，不建議使用下列參數：

表格 ***4-3*** 不建議使用的參數

標籤名稱	調整
connection-tester-class	驅動程式現在會根據 XML 描述元檔案中的資訊，在執行時期以動態的方式建立連接測試器類別。此參數仍然可以運作，以便確保反向相容性。不過，不鼓勵繼續使用它。
connection-test-stmt	驅動程式現在會根據 XML 描述元檔案中的資訊，在執行時期以動態的方式建立連接測試器類別。此參數仍然可以運作，以便確保反向相容性。不過，不鼓勵繼續使用它。
reconnect-interval	兩個通道上的重新連接間隔現在都固定爲 30 秒。

4.2.3 驗證參數

輸入驅動程式之後，請提供目標資料庫的驗證資訊。

驗證 **ID**

「驗證 ID」是驅動程式之資料庫使用者 / 登入帳戶的名稱。每個資料庫的安裝 SQL 程序檔都會提供此帳戶對受支援的資料庫驗證時，所需之資料庫權限的相關資訊。這些程序檔位於 *install-dir*\tools\sql*abbreviated-database-name*\install 安裝目錄 \tools\sql\ 縮寫的資料庫名稱 \install 目錄。

透過記號 {$username}，可以在「連接內容」參數值中參考此值。請參閱「連接內容」，第 49 頁。

範例組態的預設值是 idm。

驗證網路位置

驗證網路位置是目標資料庫的 JDBC URL。

URL 格式和內容都是專屬的。它們在不同協力廠商 JDBC 驅動程式之間也有所不同。不過，它們在內容上有部份相似之處。每個 URL (不管格式爲何) 通常都包含 IP 位址或 DNS 名稱、連接埠號碼和資料庫識別碼。如需驅動程式的正確語法和內容要求，請參閱協力廠商驅動程式文件。

如需受支援的協力廠商驅動程式的 JDBC URL 語法清單，請參閱「JDBC URL 語法」，第 109 頁。

重要：使用無觸發發行時，變更此值中非 URL 內容的任何項目都會強制重新同步化所有物件。

應用程式密碼

應用程式密碼是驅動程式之資料庫使用者 / 登入帳戶的密碼。範例驅動程式組態的預設值是 novell。

透過記號 {$password}，可以在「連接內容」參數值中參考此值。請參閱「連接內容」，第 49 頁。

4.3 驅動程式參數

下表概述的是所有驅動程式層級參數及其內容：

表格 ***4-4*** 驅動程式參數和內容

顯示名稱	標籤名稱	範例值	預設值	必要
協力廠商 JDBC 驅動程式類別名稱	jdbc-class	oracle.jdbc.driver.OracleDriver	(無)	是
時間語法	time-syntax	1 (整數)	1 (整數)	否
同步化過濾器	sync-filter	schema (包含於綱要成員資格)	(無)	否
綱要名稱	sync-schema	indirect	(無)	是 [1]
包含過濾器運算式	include-table-filter	IDM_.*	(無)	否
排除過濾器運算式	exclude-table-filter	BIN\$.{22}==\$0	(無)	否
表格 / 檢視窗名稱	sync-tables	usr	(無)	是 [1]
連接啓始化陳述式	connection-init	USE idm	(無)	否
使用最少的連接？	use-single-connection	0 (否)	(動態 [3])	否

顯示名稱	標籤名稱	範例值	預設值	必要
連接內容	use-single-connection	USER={$username}; PASSWORD={$password}	(動態 [3])	否
狀態目錄	state-dir	. (目前目錄)	. (目前目錄)	否
JDBC 驅動程式描述元檔名	jdbc-driver-descriptor	ora_client_thin.xml	(無)	否
資料庫描述元檔名	database-descriptor	ora_10g.xml	(無)	否
使用手動異動？	use-manual-transactions	1 (是)	(動態 [2])	否
異動隔離層級	transaction-isolation-level	read committed	(動態 [3])	否
重複使用陳述式？	reuse-statements	1 (重複使用)	(動態 [3])	否
傳回的結果集數目	handle-stmt-results	one	(動態 [3])	否
啓用陳述式層級鎖定？	enable-locking	1 (是)	0 (否)	否
鎖定陳述式產生器類別	lock-generator-class	com.novell.nds.dirxml.driver.jdbc.db.lock.OraLockGenerator	(動態 [3])	否
啓用參考屬性支援？	enable-refs	1 (是)	1 (是)	否
啓用中繼識別碼支援？	enable-meta-identifiers	1 (是)	1 (是)	否
強制使用者名稱大小寫	force-username-case	upper (大寫)	(無)	否
左外部結合運算子	left-outer-join-operator	(+)	(動態 [3])	否
取回最少中繼資料	minimal-metadata	0 (否)	(動態 [3])	否
函數傳回方法	function-return-method	result set	(動態 [3])	否
支援中繼資料取回中的綱要？	supports-schemas-in-metadata-retrieval	1 (是)	(動態 [3])	否
欄名稱排序方式	column-position-comparator	com.novell.nds.dirxml.driver.jdbc.util.config.comp.StringByteComparator (十六進位值)	(動態 [3])	否

[1] 如果「同步化過濾器」參數不存在，則其中一個互斥參數必須存在。請參閱「同步化過濾器」，第 44 頁。[2] 此預設值會在執行時期以動態的方式從描述元檔案和資料庫中繼資料衍生。[3] 此預設值會在執行時期以動態的方式從描述元檔案衍生。

驅動程式參數歸屬於下列子類別：

- 「未分類參數」，第 60 頁
- 「資料庫範圍設定參數」，第 44 頁
- 「連接性參數」，第 48 頁
- 「相容性參數」，第 50 頁

4.3.1 未分類的參數

協力廠商 **JDBC** 驅動程式類別名稱

此參數是協力廠商 JDBC 驅動程式的完整 Java 類別名稱。

下表列出此參數的內容：

表格 ***4-5*** 協力廠商 *JDBC* 驅動程式類別名稱：內容

內容	值
標籤名稱	jdbc-class
必要？	是
區分大小寫？	是
範例值	oracle.jdbc.driver.OracleDriver
預設值	(無)

如需受支援的協力廠商 JDBC 驅動程式類別名稱的清單，請參閱 「JDBC 驅動程式類別名稱」，第 110 頁。

時間語法

「時間語法」參數可指定驅動程式傳回之時間相關資料類型的格式。該格式可以是下列任何選項：

格式：帶正負號的整數。可將資料庫 「時間」、「日期」和 「時戳」值做爲帶正負號的 32 位元整數傳回，並將其映射至類型爲 「時間」或 「時戳」的 eDirectory™ 屬性。

eDirectory 的 「時間」和 「時戳」語法由不帶正負號的 32 位元整數組成，表示自 1970 年 1 月 1 日中午 12 點 (UTC) 開始經過的整秒數。此資料類型的範圍上限大約是 136 年。當解譯爲不帶正負號的整數時 (如最初預期一樣)，這些語法便能將範圍在 1970 年至 2106 年之間的日期和時間表示到秒。而當解譯爲帶正負號的整數時，這些語法便能將範圍在 1901 年至 2038 年之間的日期和時間表示到秒。

此選項爲預設值。它具有兩個問題：

- Identity Vault 的 「時間」和 「時戳」語法所表示的日期範圍無法大於資料庫 「日期」或 「時戳」語法所表示的日期範圍。
- Identity Vault 「時間」和 「時戳」語法精確到秒。資料庫 「時戳」語法經常精確到毫微秒 (十億分之一秒)。

第二和第三個選項可以克服這兩項限制。

格式：規範字串。 可將資料庫 「時間」、「日期」和 「時戳」值做爲規範字串傳回，並將其映射至類型爲 「數值字串」的屬性。

下表顯示的是抽象資料庫資料類型及其對應的規範字串表示：

表格 ***4-6*** 資料庫類型和規範字串表示

JDBC 資料類型	**規範字串格式**[1]
java.sql.Time	HHMMSS
java.sql.Date	CCYYMMDD
java.sql.Timestamp	CCYYMMDDHHMMSSNNNNNNNNN

[1] C = 世紀、Y = 年、M = 月、D = 天、H = 小時、M= 分鐘、S = 秒、N = 毫微秒

這些長度固定的格式在任何平台上的任何地區設定中都會按時間順序排列。即使毫微秒的精度會依資料庫而不同，「時戳」的長度也不會發生變化。

格式：**Java** 字串表示。可如透過方法 toString():java.lang.String 傳回一樣，以「Java 字串」表示傳回資料庫「時間」、「日期」和「時戳」值，並將其映射至類型爲「大小寫忽略」/「大小寫相符字串」的屬性。

下表顯示的是抽象資料庫資料類型及其對應的「Java 字串」表示：

表格 ***4-7*** 資料庫類型和 *Java* 字串格式

JDBC 資料類型	**Java 字串格式**[1]
java.sql.Time	hh:mm:ss
java.sql.Date	yyyy-mm-dd
java.sql.Timestamp	yyyy-mm-dd hh:mm:ss.fffffffff

[1] y= 年、m= 月、d= 天、h= 小時、m= 分鐘、s= 秒、f= 毫微秒

這些長度固定的格式在任何平台上的任何地區設定中都會按時間順序排列。毫微秒的精度會依資料庫而不同，因此「時戳」長度也會依資料庫而不同。

下表列出「時間語法」參數的內容：

表格 ***4-8*** 時間語法：內容

內容	**值**
標籤名稱	time-syntax
必要？	no
預設值	1 (整數)
正確的值	1 (整數) 2 (規範字串) 3 (java 字串)
與綱要相依？	True

狀態目錄

「狀態目錄」參數可指定驅動程式例項應儲存狀態資料的位置。狀態資料目前用於無觸發發行。請參閱 「無觸發發行參數」，第 72 頁。未來可能會使用狀態資料儲存其他狀態資訊。

每個驅動程式例項都具有兩個狀態檔案。狀態檔名是遵循 jdbc_ 驅動程式例項 *GUID*.db 和 jdbc_ 驅動程式例項 *GUID*.lg 的格式。例如，jdbc_bd2a3dd5-d571-4171-a195-28869577b87e.db 和 jdbc_bd2a3dd5-d571-4171-a195-28869577b87e.lg 都是狀態檔名。如果您需要手動識別並刪除驅動程式例項的狀態檔案，啓動時就會追蹤每個驅動程式例項的 GUID。每次啓動具有相同狀態目錄的驅動程式例項時，都會刪除目前狀態目錄中的 defunct 狀態檔案 (那些屬於刪除之驅動程式的檔案)。

下表列出此參數的內容：

表格 ***4-9*** 狀態目錄：內容

內容	值
標籤名稱	state-dir
必要？	no
區分大小寫？	platform-dependent
範例值	c:\novell\nds\DIBFiles
預設值	. (目前目錄)

4.3.2 資料庫範圍設定參數

- 「同步化過濾器」，第 44 頁
- 「綱要名稱」，第 46 頁
- 「包含過濾器運算式」，第 47 頁
- 「排除過濾器運算式」，第 47 頁
- 「表格 / 檢視窗名稱」，第 47 頁

同步化過濾器

「同步化過濾器」參數可決定哪些資料庫物件 (例如表格和檢視窗) 是同步化綱要的成員 (在執行時期驅動程式可看到的表格 / 檢視窗集)。新增此參數之後，驅動程式現在可以兩種模式執行：可識別綱要或不可識別綱要。

不可識別綱要模式。當「同步化過濾器」參數存在，並設爲 empty (排除所有表格 / 檢視窗) 時，驅動程式爲不可識別綱要。它在啓動時不會取回表格 / 檢視窗中繼資料，因此無需任何中繼資料方法。請參閱附錄 D 「java.sql.DatabaseMetaData 方法」，第 149 頁。

當爲不可識別綱要時，同步化綱要可以爲 empty。「綱要名稱」和「同步化表格 / 檢視窗」參數都會完全忽略。兩者皆不是必要的，可以存在或不存在，具有值或不具有值。請參閱「綱要名稱」，第 46 頁和「表格 / 檢視窗名稱」，第 47 頁。

在不可識別綱要模式中，驅動程式會扮演內嵌式 SQL 的傳遞代辦。在此狀態中，標準 XDS 事件 (例如，「新增」、「修改」和「刪除」) 會忽略。請參閱「在 XDS 事件中內嵌 SQL 陳述式」，第 95 頁。同時，觸發或無觸發的發行會不再有效。

可識別綱要模式。當「同步化過濾器」參數不存在，或設爲 empty (排除所有表格 / 檢視窗) 以外的值時，驅動程式爲可識別綱要。它會取回有限數量之表格 / 檢視窗上的表格 / 檢視窗中繼資料，以便於資料同步化。您可以快取單一資料庫使用者 (包含於綱要成員資格) 擁有之所有表格 / 檢視窗上的中繼資料，或快取表格 / 檢視窗名稱 (包含於表格 / 檢視窗名稱) 明確清單上的中繼資料。當爲可識別綱要時，驅動程式會在啓動時取回資料庫表格 / 檢視窗中繼資料。如需必要中繼資料方法的清單，請參閱附錄 D 「java.sql.DatabaseMetaData 方法」，第 149 頁。

當爲可識別綱要時，參數「綱要名稱」或「表格 / 檢視窗名稱」必須存在且具有值。因爲這兩個參數互斥，所以只有一個參數可以具有值。請參閱「綱要名稱」，第 46 頁和「表格 / 檢視窗名稱」，第 47 頁。

下表列出需要驅動程式爲可識別綱要的參數。當驅動程式爲不可識別綱要時，這些參數對驅動程式行爲便不會有任何影響。

表格 ***4-10*** 與綱要相依參數

參數
鎖定陳述式產生器類別
啓用參考屬性支援？
啓用中繼識別碼支援？
左外部結合運算子
取回最少中繼資料
支援中繼資料取回中的綱要？
欄名稱排序方式
停用陳述式層級鎖定
檢查更新計數？
插入時新增預設值？
產生 / 取回方法 (全域表格)
取回時機 (全域表格)
取回時機
停用發行者？
停用陳述式層級鎖定？
發行模式
啓用未來事件處理？
事件記錄表格名稱
刪除已處理的列？
允許迴路？
啓動選項
輪詢間隔 (以秒爲單位)

參數
每日發行時間
輪詢後陳述式
批次大小

下表列出此參數的內容：

表格 ***4-11*** 同步化過濾器：內容

內容	值
標籤名稱	sync-filter
必要？	no
區分大小寫？	no
範例值	indirect
正確的值	empty (排除所有表格 / 檢視窗) schema (包含於綱要成員資格) list (包含於表格 / 檢視窗名稱)
預設值：	(無)

綱要名稱

「綱要名稱」參數識別所同步化的資料庫綱要。資料庫綱要與所同步化之表格或檢視窗的擁有者名稱類似。例如，若要同步化歸屬於資料庫使用者 idm 的 usr 和 grp 兩個表格，您需要輸入 idm 做爲此參數的值。

在使用此參數而不是「表格 / 檢視窗名稱」時，驅動程式會隱含地使資料庫物件的名稱符合綱要。因此，除非參考預存程序、函數或表格名稱的參數位於此處指定之綱要以外的綱要中，否則它們無需符合綱要。特別是「方法和時機 (本地表格)」和「事件記錄表格名稱」會受到影響。請參閱「表格 / 檢視窗名稱」，第 47 頁、「方法和時機 (本地表格)」，第 63 頁和「事件記錄表格名稱」，第 70 頁。

下表列出此參數的內容：

表格 ***4-12*** 綱要名稱：內容

內容	值
標籤名稱	sync-schema
必要？	yes[1]
區分大小寫？	請參閱「未分隔的識別碼區分大小寫」，第 126 頁。
範例值	indirect
預設值：	(無)

[1] 在沒有「同步化過濾器」參數的情況下使用「綱要名稱」參數時，「表格 / 檢視窗名稱」參數必須保留爲 empty 或從組態中省略。請參閱 「同步化過濾器」，第 44 頁和 「表格 / 檢視窗名稱」，第 47 頁。

重要：使用無觸發發行時，變更「綱要名稱」參數的值會強制重新同步化所有物件。

包含過濾器運算式

僅當使用「綱要名稱」參數時，才可操作「包含過濾器運算式」參數。請參閱 「綱要名稱」，第 46 頁。

下表列出此參數的內容：

表格 ***4-13*** 包含過濾器運算式：內容

內容	值
標籤名稱	include-table-filter
必要？	no
區分大小寫？	yes
範例值	idm_*. (以 "idm_" 開頭的所有表格 / 檢視窗名稱)
預設值	(無)
正確的值	(任何正確的 Java 一般運算式)

排除過濾器運算式

僅當使用「綱要名稱」參數時，此參數才會運作。請參閱 「綱要名稱」，第 46 頁。

下表列出此參數的內容：

表格 ***4-14*** 排除過濾器運算式：內容

內容	值
標籤名稱	exclude-table-filter
必要？	no
區分大小寫？	yes
範例值	bin*. (以 "bin" 開頭的所有表格 / 檢視窗名稱)
預設值	(無)
正確的值	(任何正確的 Java 一般運算式)

表格 / 檢視窗名稱

「表格 / 檢視窗名稱」參數可讓您藉由列出要同步化之邏輯資料庫類別的名稱，來建立邏輯資料庫綱要。邏輯資料庫類別名稱是父表格和檢視窗的名稱。列出子表格名稱是錯誤的。

此參數對於同步化不支援綱要概念的資料庫 (例如 MySQL)，或當資料庫綱要包含大量表格或檢視窗 (但其中只有幾個相關) 時，特別有用。減少由驅動程式快取之表格 / 檢視窗定義的數目可以縮短啓動時間，以及減少執行時期記憶體使用。

使用此參數而非 「綱要名稱」時，您可能需要使參考預存程序、函數或表格名稱的其他參數符合綱要。特別是 「方法和時機 (本地表格)」和 「事件記錄表格名稱」參數會受到影響。請參閱 「綱要名稱」，第 46 頁、「方法和時機 (本地表格)」，第 63 頁和 「事件記錄表格名稱」，第 70 頁。

下表列出此參數的內容：

表格 ***4-15*** 表格 / 檢視窗名稱：內容

內容	值
標籤名稱	sync-tables
必要？	yes[1]
區分大小寫？	請參閱 「未分隔的識別碼區分大小寫」，第 126 頁。
分隔符	分號、空白字元、逗號
範例值	indirect.usr; indirect.grp
預設值	(無)

[1] 在沒有 「同步化過濾器」參數的情況下使用此參數時，「綱要名稱」參數必須保留爲 empty 或從組態中省略。請參閱 「同步化過濾器」，第 44 頁和 「綱要名稱」，第 46 頁。

4.3.3 連接性參數

使用最少的連接？

使用最少的連接？ 參數可指定驅動程式是否應使用兩個而非三個資料庫連接。

在預設狀態下，驅動程式會使用三個連接：一個用於訂閱，而另兩個用於發行。 「訂閱者」通道會使用其兩個連接中的其中一個來查詢事件，使用另一個來協助執行查詢回覆操作。

當此參數設爲布林值 True 時，所需的資料庫連接數目會減少爲兩個。其中一個連接會在 「訂閱者」和 「發行者」通道之間共用，並用於處理訂閱和發行查詢回覆事件。另一個連接則用於查詢發行事件。

在先前版本中，驅動程式可以藉由使用單一連接來支援雙向同步化。發行演算法已經過重新設計，可用於增進效能、使其能支援未來事件的處理，並以需要額外一個連接來換取克服先前版本的限制。

表格 ***4-16*** 使用最少的連接？：內容

內容	值
標籤名稱	use-single-connection
必要？	no
預設值	(動態 [1])
正確的值	1、yes、true (是) 0、no、false (否)
與綱要相依	False

[1] 此預設值會在執行時期以動態的方式自描述元檔案衍生。否則，預設值爲布林值 False。

附註：將此參數設爲布林值 True 會降低效能。

連接啓始化陳述式

「連接啓始化陳述式」參數可指定連接至目標資料庫之後應立即執行的 SQL 陳述式 (如果有的話)。連接啓始化陳述式對於變更資料庫網路位置和設定會期內容而言很管用。每當驅動程式 (不考慮通道) 連接或重新連接至目標資料庫時，都會執行這些陳述式。

下表列出此參數的內容：

表格 ***4-17*** 連接啓始化陳述式：內容

內容	值
標籤名稱	connection-init
必要？	no
區分大小寫？	請參閱 「未分隔的識別碼區分大小寫」，第 126 頁。
分隔符	分號
範例值	USE idm; SET CHAINED OFF
預設值	(無)
與綱要相依	False

連接內容

「連接內容」參數可指定驗證內容。此參數對於指定無法透過在 「驗證網路位置」參數中指定之 JDBC URL 設定的內容而言很管用。請參閱 「驗證網路位置」，第 40 頁。

此參數的主要目的是針對協力廠商 JDBC 驅動程式啓用加密傳輸。如需相關連接內容的清單，請參閱 「Sybase Adaptive Server Enterprise JConnect JDBC 驅動程式」，第 120 頁和 「Oracle Thin Client JDBC 驅動程式」，第 118 頁。

連接內容會指定爲鍵值配對。鍵會指定爲 "=" 字元左邊的值。值是 "=" 字元右邊的值。您可以指定多個鍵值配對，但是必須以 ";" 字元分隔每個配對。

當您使用「連接內容」參數時，驗證資訊便可透過在「驗證網路位置」參數或在此處指定的 JDBC URL 傳遞。請參閱 「驗證網路位置」，第 40 頁。

如果指定爲連接內容，值記號便可做爲在「驗證 ID」參數中指定的資料庫使用者名稱預留位置，以及在「應用程式密碼」參數中指定的密碼預留位置使用。請參閱 「驗證 ID」，第 39 頁和 「應用程式密碼」，第 40 頁。對於使用者名稱，記號是 {$username}。對於密碼，記號是 {$password}。

下表列出此參數的內容：

表格 ***4-18*** 連接內容：內容

內容	值
標籤名稱	connection-properties
必要？	no
區分大小寫？	third-party JDBC driver-dependent
分隔符	分號 (;)
範例值	USER={$username}; PASSWORD={$password}; SYBSOCKET_FACTORY=DEFAULT
預設值	(無)
與綱要相依	False

4.3.4 相容性參數

- 「JDBC 驅動程式描述元檔名」，第 51 頁
- 「資料庫描述元檔名」，第 51 頁
- 「使用手動異動？」，第 51 頁
- 「異動隔離層級」，第 52 頁
- 「重複使用陳述式？」，第 53 頁
- 「傳回的結果集數目」，第 54 頁
- 「啓用陳述式層級鎖定？」，第 54 頁
- 「鎖定陳述式產生器類別」，第 54 頁
- 「啓用參考屬性支援？」，第 55 頁
- 「啓用中繼識別碼支援？」，第 55 頁
- 「強制使用者名稱大小寫」，第 56 頁
- 「左外部結合運算子」，第 56 頁
- 「取回最少中繼資料」，第 57 頁
- 「函數傳回方法」，第 57 頁
- 「支援中繼資料取回中的綱要？」，第 58 頁
- 「欄名稱排序方式」，第 58 頁

JDBC 驅動程式描述元檔名

「JDBC 驅動程式描述元檔名」參數可指定要使用的協力廠商 JDBC 描述元檔案。描述元檔名不能以底線字元 (例如，_mysql_jdriver.xml) 爲字首，因爲此類檔名是保留的。將描述元檔案置於以區分大小寫之 "jdbc" 爲字首的 jar 檔案 (例如 JDBCCustomConfig.jar) 和 jar 檔案的 com/novell/nds/dirxml/driver/jdbc/db/descriptor/driver 目錄中。

下表列出此參數的內容：

表格 ***4-19*** *JDBC* 驅動程式描述元檔名：內容

內容	值
標籤名稱	jdbc-driver-descriptor
必要？	no
區分大小寫？	platform-dependent
範例值	my_custom_jdbc_driver_descriptor.xml
預設值	(無)
與綱要相依	False

資料庫描述元檔名

「資料庫描述元檔名」參數可指定要使用的資料庫描述元檔案。請勿在「描述元」檔名的字首中使用底線字元 (例如，_mysql.xml)。此類名稱是保留的。將「描述元」檔案置於以區分大小寫之 "jdbc" 爲字首的 jar 檔案 (例如 JDBCCustomConfig.jar)。同時，將「描述元」檔案置於 jar 檔案的 com/novell/nds/dirxml/driver/jdbc/db/descriptor/db 目錄。

下表列出此參數的內容：

表格 ***4-20*** 資料庫描述元檔名：內容

內容	值
標籤名稱	jdbc-driver-descriptor
必要？	no
區分大小寫？	platform-dependent
範例值	my_custom_database_descriptor.xml
預設值	(無)
與綱要相依	False

使用手動異動？

「使用手動異動？」參數可指定是否使用手動或使用者定義的異動。

此參數主要用於啓用與不支援異動之 MySQL MyISAM 表格類型之間的互通。

當設爲布林值 True 時，驅動程式使用手動異動。當設爲布林值 False 時，由驅動程式執行的每個陳述式都會以自發的方式執行 (自動)。

下表列出此參數的內容：

表格 ***4-21*** 使用手動異動？：內容

內容	值
標籤名稱	use-manual-transactions
必要？	no
區分大小寫？	否
預設值	(動態 [1])
正確的值	1、yes、true (是) 0、no、false (否)
與綱要相依	False

[1] 此預設值會在執行時期以動態的方式自描述元檔案和資料庫中繼資料衍生。

附註：若要確保資料完整性，請盡可能隨時將此參數設爲布林值 True。

異動隔離層級

「異動隔離層級」參數可針對驅動程式所使用的連接設定異動隔離層級。有下列 6 個值存在：

- unsupported
- none
- read uncommitted
- read committed
- repeatable read
- serializable

其中 5 個值對應於在 java.sql 介面連接 (http://java.sun.com/j2se/1.5.0/docs/api/java/sql/Connection.html) 中定義的公用常數。

因爲部份協力廠商驅動程式不支援將連接的異動隔離層級設爲 none，所以驅動程式還支援其他非標準化值 unsupported。PostgreSQL 線上文件 (http://www.postgresql.org/docs/current/static/transaction-iso.html) 具有一個關於每個隔離層級實際意義之較好的簡要描述。

重要：由於資料庫不同，受支援的隔離層級的清單也會不同。如需受支援資料庫的受支援異動隔離層級清單，請參閱 「受支援的異動隔離層級」，第 126 頁。

建議您使用 read committed 異動隔離層級，因爲它是防止驅動程式查看未確認變更 (改動讀取) 的最低隔離層級。

下表列出此參數的內容：

表格 ***4-22*** 異動隔離層級：內容

內容	值
標籤名稱	transaction-isolation-level
必要？	no
區分大小寫？	no
預設值	(動態 [1])
正確的值	unsupported none read uncommitted read committed repeatable read serializable
與綱要相依	False

[1] 此預設值會在執行時期以動態的方式自描述元檔案衍生。否則，預設值爲 read committed。

重複使用陳述式？

「重複使用陳述式？」參數可指定在給定的連接上是否可以一次使用一或多個 java.sql.Statement 項目。請參閱 java.sql.Statement (http://java.sun.com/j2se/1.5.0/docs/api/java/sql/Statement.html)。

此參數主要用於啓用與 Microsoft SQL Server 2000 Driver for JDBC 的互通。

當設爲布林值 True 時，驅動程式會配置一次「Java SQL 陳述式」並重複使用它。當設爲布林值 False 時，驅動程式會在每次使用陳述式物件時配置 / 取消配置它們，以確定在給定的連接上一次只使用一個陳述式。

下表列出此參數的內容：

表格 ***4-23*** 重複使用陳述式？：內容

內容	值
標籤名稱	reuse-statements
必要？	no
區分大小寫？	no
預設值	(動態 [1])
正確的值	1、yes、true (是) 0、no、false (否)
與綱要相依	False

[1] 此預設值會在執行時期以動態的方式自描述元檔案衍生。否則，預設值爲布林值 True。

附註：將此參數設爲布林值 False 會降低效能。

傳回的結果集數目

「傳回的結果集數目」參數可指定能夠從任意 SQL 陳述式傳回的 java.sql.Result 物件數量。請參閱 java.sql.ResultSet (http://java.sun.com/j2se/1.5.0/docs/api/java/sql/ResultSet.html)。

此參數主要用於在評估任意 SQL 陳述式的結果時，避免 「Oracle Thin Client JDBC 驅動程式」，第 118 頁中發生無限迴路情況。

下表列出此參數的內容：

表格 ***4-24*** 傳回的結果集數目：內容

內容	值
標籤名稱	handle-stmt-results
必要？	no
範例值	one
預設值	(動態 [1])
正確的值	none、no (無) single、one (一個) multiple、many、yes (多個)
與綱要相依	False

[1] 此預設值會在執行時期以動態的方式自描述元檔案衍生。否則，預設值爲 multiple、many 或 yes。

啓用陳述式層級鎖定？

「啓用陳述式層級鎖定？」參數可指定驅動程式在執行 SQL 陳述式之前，是否明確地鎖定資料庫資源。

下表列出此參數的內容：

表格 ***4-25*** 啓用陳述式層級鎖定？：內容

內容	值
標籤名稱	enable-locking
必要？	no
預設值	0 (否)
正確的值	1、yes、true (是) 0、no、false (否)
與綱要相依	True

鎖定陳述式產生器類別

「鎖定陳述式產生器類別」參數可指定爲了產生明確鎖定待處理 SQL 陳述式之資料庫資源所需要的 SQL 陳述式，所使用的 DBLockStatementGenerator 實作。DBLockStatementGenerator 介面上的資訊位於隨附於驅動程式的 Java 文件中。

下表列出此參數的內容：

表格 ***4-26*** 鎖定陳述式產生器類別：內容

內容	值
標籤名稱	lock-generator-class
必要？	no
範例值	com.novell.nds.dirxml.driver.jdbc.db.lock.OraLockGenerator
預設值	(動態 [1])
正確的值	1、yes、true (是) 0、no、false (否)
與綱要相依	True

[1] 此預設值會在執行時期以動態的方式自描述元檔案衍生。否則，預設值爲 com.novell.nds.dirxml.driver.jdbc.db.lock.DBLockGenerator。

啓用參考屬性支援？

「啓用參考屬性支援？」參數會切換驅動程式是否要辨識邏輯資料庫類別之間的外部索引鍵條件約束。這些參數用於表示內含項目。邏輯資料庫類別內，父表格和子表格之間的外部索引鍵條件約束不會受到影響。

當設爲布林值 True 時，外部索引鍵欄會解譯爲參考。當設爲布林值 False 時，外部索引鍵欄會解譯爲非參考。

此參數的主要目的是確保驅動程式 1.0 版的反向相容性。如需 1.0 版的相容性，請將此參數設爲布林值 False。

下表列出此參數的內容：

表格 ***4-27*** 啓用參考屬性支援？：內容

內容	值
標籤名稱	enable-refs
必要？	no
預設值	1 (是)
正確的值	1、yes、true (是) 0、no、false (否)
與綱要相依	True

啓用中繼識別碼支援？

「啓用中繼識別碼支援？」參數會切換驅動程式是否要將檢視欄名稱字首 (例如 "pk_" 和 "fk_") 嚴格解譯爲中繼資料。當解譯爲中繼資料時，此類字首不會視爲檢視欄名稱的一部份。

例如，當啓用中繼識別碼支援時，欄 "pk_idu" 具有有效的欄名 "idu"，以禁止在同一檢視窗中存在具有相同有效名稱的欄。當停用中繼識別碼支援時，欄 "pk_idu" 具有有效的欄名 "pk_idu,"，以允許存在其他名爲 "idu" 的欄。而且，當啓用中繼識別碼支援時，具有名爲 "pk_idu" 之主索引鍵的檢視窗會與具有名爲 "idu" 之主索引鍵欄的表格衝突。當停用中繼識別碼支援時，它們就不會衝突。

當設爲布林值 True 時，檢視欄字首會解譯爲中繼資料。當設爲布林值 False 時，檢視欄名稱字首會解譯爲正確欄名的一部份。

此參數的主要目的是確保驅動程式 1.5 版的反向相容性。如需 1.5 版的相容性，請將此參數設爲布林值 False。

下表列出此參數的內容：

表格 ***4-28*** 啓用中繼識別碼支援？：內容

內容	值
標籤名稱	enable-meta-identifiers
必要？	no
預設值	1 (是)
正確的值	1、yes、true (是) 0、no、false (否)
與綱要相依	True

強制使用者名稱大小寫

「強制使用者名稱大小寫」參數會變更用於驗證目標資料庫的驅動程式使用者名稱大小寫。

此參數的主要目的是當針對 ANSI 相容的資料庫使用該參數時，啓用與「Informix JDBC 驅動程式」的互通。請參閱 「Informix JDBC 驅動程式」，第 114 頁。

下表列出此參數的內容：

表格 ***4-29*** 強制使用者名稱大小寫：內容

內容	值
標籤名稱	force-username-case
必要？	no
預設值	(不強制)
正確的值	lower (針對小寫) mixed (針對大小寫混合) upper (針對大寫)
與綱要相依	False

左外部結合運算子

「左外部結合運算子」參數可指定用於無觸發發行查詢的左外部結合運算子，將來可能會有其他用途。

下表列出此參數的內容：

表格 ***4-30*** 左外部結合運算子：內容

內容	值
標籤名稱	left-outer-join-operator
必要？	no
預設值	(動態 [1])
正確的值	*= (+) LEFT OUTER JOIN
與綱要相依	True

[1] 此預設值會在執行時期以動態的方式自描述元檔案衍生。否則，預設值爲 LEFT OUTER JOIN。

取回最少中繼資料

當設爲布林值 True 時，驅動程式只會呼叫必要的中繼資料方法。當設爲布林值 False 時，驅動程式會呼叫必要的和選擇性的中繼資料方法。如需必要的和選擇性的中繼資料方法清單，請參閱附錄 D 「java.sql.DatabaseMetaData 方法」，第 149 頁。多值和參考屬性同步化需要選擇性中繼資料方法。

表格 ***4-31*** 取回最少中繼資料：內容

內容	值
標籤名稱	minimal-metadata
必要？	no
預設值	(動態 [1])
正確的值	1、yes、true (是) 0、no、false (否)
與綱要相依	True

[1] 此預設值會在執行時期以動態的方式自描述元檔案衍生。否則，預設值爲布林值 False。

附註：將此值設爲布林值 True 會改進啓動時間和協力廠商 JDBC 驅動程式相容性，但同時也會影響功能。

函數傳回方法

「函數傳回方法」參數可指定如何從資料庫函數中取回資料。

此參數的主要目的是啓用與 Informix JDBC 驅動程式的互通。請參閱「Informix JDBC 驅動程式」，第 114 頁。

當設爲 result set 時，函數結果便會透過結果集取回。當設爲 return value 時，函數結果會做爲單一的純量傳回值取回。

表格 ***4-32*** 函數傳回方法：內容

內容	值
標籤名稱	function-return-method
必要？	no
預設值	(動態 [1])
正確的值	result set return value (純量傳回值)
與綱要相依	False

[1] 此預設值會在執行時期以動態的方式自描述元檔案衍生。

支援中繼資料取回中的綱要？

「支援中繼資料取回中的綱要？」參數可指定當取回資料庫中繼資料時，是否應該使用綱要名稱。

此參數的主要目的是當針對 ANSI 相容的資料庫使用該參數時，啓用與「Informix JDBC 驅動程式」的互通。請參閱 「Informix JDBC 驅動程式」，第 114 頁。

當設爲布林值 True 時，會使用綱要名稱。當設爲布林值 False 時，則不會使用它們。

表格 ***4-33*** 支援中繼資料取回中的綱要？：內容

內容	值
標籤名稱	supports-schemas-in-metadata-retrieval
必要？	no
預設值	(動態 [1])
正確的值	1、yes、true (是) 0、no、false (否)
與綱要相依	False

[1] 此預設值會在執行時期以動態的方式自描述元檔案衍生。否則，預設值爲布林值 True。

欄名稱排序方式

「欄名稱排序方式」參數可指定如何決定不支援依欄名排序之舊資料庫的欄位置。

此參數的主要目的是啓用與舊資料庫 (例如 DB2/AS400) 的互通。

依十六進位值排序欄名可確保如果驅動程式例項重新定位至不同的伺服器時，它可以繼續運作而無需修改。依平台或地區設定字串定序順序來排序欄名會更直觀，但如果驅動程式重新定位至不同的伺服器，則可能需要變更組態。特別是記錄表格欄順序和複合欄名順序可能會變更。在後一種情況下，可能需要更新「綱要映射」規則和物件關聯值。在前一種情況下，則可能要重新命名記錄表格欄。

只要發生下列情況，就可能需要指定所有完整的 Java 類別名稱：

- Java 類別名稱實作 java.util.Comparator (http://java.sun.com/j2se/1.5.0/docs/api/java/util/Comparator.html) 介面。
- Java 類別名稱接受 java.lang.String (http://java.sun.com/j2se/1.5.0/docs/api/java/lang/String.html) 引數。
- 類別位於執行時期 classpath 中。

表格 ***4-34*** 欄名稱排序方式：內容

內容	值
標籤名稱	column-position-comparator
必要？	no
預設值	(動態 [1])
正確的值	com.novell.nds.dirxml.driver.jdbc.util.config.comp.StringByteComparator (十六進位值) com.novell.nds.dirxml.driver.jdbc.util.config.comp.StringComparator (字串定序順序) (接受 java.lang.String 引數的所有 java.util.Comparator)
與綱要相依	True

[1] 此預設值會在執行時期以動態的方式自描述元檔案衍生。否則，預設值爲 com.novell.nds.dirxml.driver.jdbc.util.config.comp.StringByteComparator。

重要：在您針對給定的組態設定此參數之後，請勿變更該參數。

4.4 訂閱參數

下表概述「訂閱者層級」參數及其內容：

表格 ***4-35*** 訂閱者層級參數和內容

顯示名稱	標籤名稱	範例值	預設值	必要
停用訂閱者？	disable	1 (是)	0 (否)	no
產生 / 取回方法 (全域表格)	key-gen-method	auto	none (訂閱事件)	
取回時機 (全域表格)	key-gen-timing	after (在列插入之後)	before (在列插入之前)	no
方法和時機 (本地表格)	key-gen	usr("?=indirect.proc_idu()", before)	(無)	no
停用陳述式層級鎖定？	disable-locking	1 (是)	0 (否)	no
檢查更新計數？	check-update-count	0 (否)	1 (是)	no

顯示名稱	標籤名稱	範例值	預設值	必要
插入時新增預設值？	add-default-values-on-view-insert	0 (否)	(動態 [1])	no

[1] 此預設值會在執行時期以動態的方式自描述元檔案衍生。

訂閱參數分爲兩個子類別：

- 「未分類參數」，第 60 頁
- 「主索引鍵參數」，第 62 頁

4.4.1 未分類參數

- 「停用訂閱者？」，第 60 頁
- 「停用陳述式層級鎖定？」，第 60 頁
- 「檢查更新計數？」，第 61 頁
- 「插入時新增預設值？」，第 61 頁

停用訂閱者？

「停用訂閱者？」參數可指定是否停用 「訂閱者」通道。

此參數設爲布林值 True 時，「訂閱者」通道會停用。當參數設爲布林值 False 時，「訂閱者」通道會啓用。

表格 ***4-36*** 停用訂閱者？：內容

內容	值
標籤名稱	disable
必要？	no
預設值	0 (否)
正確的值	1、yes、true (是) 0、no、false (否)
與綱要相依	False

停用陳述式層級鎖定？

「停用陳述式層級鎖定？」參數可指定在執行每個 SQL 陳述式之前，是否在此通道上明確地鎖定資料庫資源。只有在 啓用陳述式層級鎖定？ 設爲布林值 True 時，此參數才會爲使用中。

此參數設爲布林值 True 時，資料庫資源便會明確地鎖定。此參數設爲布林值 False 時，資料庫資源則不會明確地鎖定。

表格 ***4-37*** 停用陳述式層級鎖定？：內容

內容	值
標籤名稱	disable-locking
必要？	no
預設值	0 (否)
正確的值	1、yes、true (是) 0、no、false (否)
與綱要相依	True

檢查更新計數？

「檢查更新計數？」參數可指定 「訂閱者」通道是否會檢查當針對表格執行 INSERT、UPDATE 和 DELETE 陳述式時，是否實際更新表格。

當設爲布林值 True 時，會檢查更新計數。如果沒有更新任何項目，則會發生例外。當設爲布林值 False 時，會忽略更新計數。

當在觸發前邏輯中重新定義陳述式時，將其參數設爲布林值 False。

因爲觸發邏輯 (其可能會復原異動) 中的錯誤不會傳達至 「訂閱者」通道，所以當使用 Microsoft SQL Server 時，請使用預設值。

表格 ***4-38*** 檢查更新計數？：內容

內容	值
標籤名稱	check-update-count
必要？	no
預設值	1 (是)
正確的值	1、yes、true (是) 0、no、false (否)
與綱要相依	True

插入時新增預設值？

「插入時新增預設值？」參數可指定當針對檢視窗執行 INSERT 陳述式時，「訂閱者」通道是否提供預設值。

此參數的主要目的是啓用與 Microsoft SQL Server 2000 的互動。此資料庫需要限制爲 NOT NULL 的檢視欄在 INSERT 陳述式中具有非 NULL 值。

當此參數設爲布林值 True 時，預設值會提供給針對檢視窗所執行的 INSERT 陳述式，且尙不可以使用明確值。當此參數設爲布林值 False 時，不會提供預設值。

表格 ***4-39*** 插入時新增預設值？：內容

內容	值
標籤名稱	add-default-values-on-view-insert
必要？	no
預設值	(動態 [1])
正確的值	1、yes、true (是) 0、no、false (否)
與綱要相依	True

[1] 此預設值會在執行時期以動態的方式自描述元檔案衍生。

4.4.2 主索引鍵參數

- 「產生 / 取回方法 (全域表格)」，第 62 頁
- 「取回時機 (全域表格)」，第 63 頁
- 「方法和時機 (本地表格)」，第 63 頁

當處理映射至 INSERT 陳述式的 <add> 事件時，「訂閱者」通道會使用主索引鍵值來建立 Identity Manager 關聯。這些參數可指定「訂閱者」通道取得建構關聯值所需之主索引鍵值的方式和時間。取得主索引鍵值的方式是主索引鍵產生 / 取回方法。取回時機是指取回主索引鍵值的時間。

下表識別受支援的方法和時機：

表格 ***4-40*** 受支援的方法和時機

方法	時機：before (列插入)	時機：after (列插入)
None (訂閱事件)	X	0[1]
Driver (訂閱者所產生)	X	X
Auto (自動產生的 / 身份欄)	0[2]	X
(預存程序 / 函數)	X	X

[1] 「訂閱者」通道會自動將此時機置換爲 before。[2] 「訂閱者」通道會自動將此時機置換爲 after。

產生 / 取回方法 (全域表格)

「產生 / 取回方法 (全域表格)」參數可指定如何針對所有父表格和檢視窗來產生或取回主索引鍵值。「方法和時機」參數會在每個表格 / 檢視窗上置換此參數。請參閱 「方法和時機 (本地表格)」，第 63 頁。

當此參數設爲 none 時，會假設訂閱事件中已存在主索引鍵值。當此參數設爲 driver 時，會使用下列其中一種方法產生主索引鍵值：

- 如果取回時機設爲 before，則使用 SELECT (MAX()+1) 陳述式

- 如果取回時機設爲 after，則使用 SELECT (MAX() 陳述式

若爲字串欄類型，「訂閱者」通道會使用 System.CurrentTimeMillis() 的傳回值來產生值，其他資料類型則不受支援。

當此參數設爲 auto 時，會透過 java.sql.Statement.getGeneratedKeys():java.sql.ResultSet 方法取回 主索引鍵值。MySQL Connector/J JDBC 驅動程式是目前實作此方法唯一受支援的協力廠商 JDBC 驅動程式。請參閱 「MySQL Connector/J JDBC 驅動程式」，第 117 頁。

表格 ***4-41*** 產生 / 取回方法 *(* 全域表格 *)*：內容

內容	值
標籤名稱	key-gen-method
必要？	no
預設值	none (訂閱事件)
正確的值	none (訂閱事件) driver (訂閱者所產生) auto (自動產生的 / 身份欄)
與綱要相依	True

取回時機 **(** 全域表格 **)**

「取回時機 (全域表格)」參數可指定 「訂閱者」通道取回所有父表格和檢視窗之主索引鍵值的時間。「方法和時機 (本地表格)」參數會置換此參數。請參閱 「方法和時機 (本地表格)」，第 63 頁。

當此參數設爲 before 時，會在插入之前取回主索引鍵值。當此參數設爲 after 時，會在插入之後取回主索引鍵值。

表格 ***4-42*** 取回時機 *(* 全域表格 *)*：內容

內容	值
標籤名稱	key-gen-timing
必要？	no
預設值	before (在列插入之前)
正確的值	before (在列插入之前) after (在列插入之後)
與綱要相依	True

方法和時機 **(** 本地表格 **)**

「方法和時機 (本地表格)」參數可指定每個父表格 / 檢視窗的主索引鍵值產生 / 取回方法和取回時機。它實際上會將產生 / 取回方法和取回時機映射至表格或檢視窗名稱。此參數的語法會以多個引數 (例如，方法名稱 (引數 *1*, 引數 *2*)) 鏡像複製程序的程式設計語言方法呼叫。

當使用表格 / 檢視窗名稱參數時，您可能需要對此參數值中參考的所有表格、檢視窗、預存程序或函數進行明確的綱要限定。當使用綱要名稱參數時，會以該綱要名稱對此參數值中參考的所有表格、檢視窗、預存程序或函數進行隱含的綱要限定。如果此參數值中參考的表格、檢視窗、預存程序或函數位於非隱含綱要的不同綱要中，則必須對它們進行綱要限定。

BNF

此參數值的 BNF (Backus Naur Form (http://cui.unige.ch/db-research/Enseignement/analyseinfo/AboutBNF.html)) 表示如下所示：

```
<key-gen> ::= <table-or-view-name> "(" <generation-retrieval-method>,
<retrieval-timing> ")" {[<delimiter>] <key-gen>}

<generation-retrieval-method> ::= none | driver | auto | """
<procedure-signature> """ | """ <function-signature>  """

<table-or-view-name> ::= <legal-undelimited-database-table-or-view-
identifier>

<delimiter> ::= ";" | "," | <white-space>

<procedure-signature> ::= <schema-qualifier> "." <stored-routine-
name>"("<argument-list>")"

<function-signature> ::= "?=" <procedure-signature>

<schema-qualifier> ::= <legal-undelimited-database-username-
identifier>

<stored-routine-name> ::= <legal-undelimited-database-stored-routine -
identifier>

<argument-list> ::= <column-name>{"," <column-name>}

<column-name> ::= <column-from-table-or-view-name-previously-
specified>
```

產生或取回方法

產生或取回方法可指定如何產生或取回 (必要時) 主索引鍵值。可能的方法有 None、Driver、Auto 和預存程序 / 函數：

None 在預設狀態下，「訂閱者」通道會假設 Identity Vault 是主索引鍵值的授權來源，且必要值已存在於給定的 <add> 事件中。在此情況下，因爲主索引鍵值已存在，所以不需要再產生。主索引鍵值只需要從目前的 <add> 事件中取回即可。當 eDirectory 屬性 (例如 GUID) 明確地綱要映射至父表格或檢視窗的主索引鍵欄時，需要使用此方法。

假設存在名爲 usr 的表格和名爲 view_usr 的檢視窗，其中 Identity Vault 是主索引鍵值的授權來源，此參數的值會如下所示：

usr(none); view_usr(none)

使用此方法時，建議您將 GUID 而不是公用名稱 (Common Name，CN) 映射至父表格或檢視窗的主索引鍵欄。

Driver 此方法會假設資料庫是指定父表格或檢視窗之主索引鍵值的授權來源。

在建立原型時或在部署的啓始階段，經常需要在寫入預存程序或函數之前，讓「訂閱者」通道產生主索引鍵值。您還可以針對不支援預存程序或函數的資料庫使用此方法。然而，當您在生產環境中使用此方法時，<add> 事件產生的所有 SQL 陳述式都應該包含在可序列化的異動中。如需其他資訊，請參閱「異動隔離層級」，第 52 頁。

若不讓所有異動都成爲可序列化，您還可以使用內嵌式 SQL 屬性設定個別的異動隔離層級。如需其他資訊，請參閱「異動隔離層級」，第 101 頁。

針對所有數值欄類型，「訂閱者」通道會使用下列方法產生主索引鍵值：

- 針對 before 時機使用簡單的 SELECT(MAX+1) 陳述式
- 針對 after 時機使用 SELECT MAX() 陳述式

若爲字串欄類型，「訂閱者」通道會使用 System.CurrentTimeMillis() 的傳回值來產生值。不支援其他資料類型。

假設存在名爲 usr 的表格和名爲 view_usr 的檢視窗，其中資料庫是主索引鍵值的授權來源，此參數的值會如下所示：

usr(driver); view_usr(driver)

使用此方法時，建議您省略「綱要映射」規則和通道過濾器的主索引鍵欄。

Auto 此方法會假設資料庫是指定父表格或檢視窗之主索引鍵值的授權來源。

部份資料庫支援會針對已插入列自動產生主索引鍵值的身份欄。此方法會透過 JDBC 3 介面方法 java.sql.Statement.getGeneratedKeys():java.sql.ResultSet 取回自動產生的主索引鍵值。MySQL Connector/J JDBC 驅動程式是目前實作此方法唯一受支援的協力廠商 JDBC 驅動程式。請參閱「MySQL Connector/J JDBC 驅動程式」，第 117 頁。

假設存在名爲 usr 的表格和名爲 view_usr 的檢視窗，其中資料庫是主索引鍵值的授權來源，此參數的值會如下所示：

usr(auto); view_usr(auto)

使用此方法時，建議您省略「綱要映射」規則和通道過濾器的主索引鍵欄。

預存程序 / 函數：此方法會假設資料庫是指定父表格或檢視窗之主索引鍵值的授權來源。

假設

- 存在名爲 usr 的表格，且主索引鍵欄名爲 idu
- 檢視窗名爲 view_usr，且主索引鍵值名爲 pk_idu
- 存在資料庫函數 func_last_usr_idu 和預存程序 sp_last_view_usr_pk_idu，兩者都會傳回各自表格 / 檢視窗之最後產生的主索引鍵值

此參數的值會如下所示：

usr("?=func_last_usr_idu()"); view_usr("sp_last_view_usr_pk_idu(pk_idu)")

在先前的範例中，參數會傳送至預存程序。參數還可以傳送至函數，但通常不需這樣做。與函數不同，預存程序通常會透過參數傳回值。若爲預存程序，則必須將主索引鍵欄做爲 IN OUT 參數傳送。非索引鍵欄必須做爲 IN 參數傳送。

若爲預存程序和函數，則參數順序、編號和資料類型必須對應於程序或函數預期之參數的順序、編號和資料類型。

使用此方法時，建議您省略「綱要映射」規則和通道過濾器的主索引鍵欄。

取回時機

「取回時機」參數可指定取回主索引鍵值的時間。

<add> 事件會一直針對父表格或檢視窗產生至少一個 INSERT 陳述式。此參數的這部份可指定相對於啓始 INSERT 陳述式，取回主索引鍵值的時間。

Before 此爲預設設定。指定此設定之後，主索引鍵值便會在啓始 INSERT 陳述式之前取回。

重要：所有產生 / 取回方法 (auto 除外) 都支援此取回時機。none 方法需要取回時機。

After 指定此設定之後，主索引鍵值便會在啓始 INSERT 陳述式之後取回。

重要：所有產生 / 取回方法 (none 除外) 都支援此取回時機。auto 方法需要取回時機。

下列範例藉由新增取回時機資訊，擴展先前的項目：

usr(none, before); view_usr(none, before)

usr(driver, before); view_usr(driver, after)

usr(auto, after); view_usr(auto, after)

usr("?=func_last_usr_idu()", before); view_usr("sp_last_view_usr_pk_idu(pk_idu)", after)

下表列出此參數的內容：

表格 ***4-43*** 取回時機：內容

內容	值
標籤名稱	key-gen
必要？	no
區分大小寫？	請參閱 「未分隔的識別碼區分大小寫」，第 126 頁。
範例值	usr("?=proc_idu()", before)
預設值	(無)
正確的值	(符合 BNF 的任何字串)
與綱要相依	True

4.5 發行參數

下表概述發行者層級參數及其內容：

表格 *4-44* 發行者層級參數和內容

顯示名稱	標籤名稱	範例值	預設值	必要
停用發行者？	disable	1 (是)	0 (否)	否
停用陳述式層級鎖定？	disable-locking	1 (是)	0 (否)	否
發行模式	publication-mode	2 (無觸發)	1 (已觸發)	否
事件記錄表格名稱	log-table	indirect_process	(無)	是 [1]
刪除已處理的列？	delete-from-log	0 (否)	1 (是)	否
允許迴路？	allow-loopback	1 (是)	0 (否)	否
啓用未來事件處理？	handle-future-events	1 (是)	0 (否)	否
啓動選項	startup-option			否
輪詢間隔 (以秒爲單位)	polling-interval	60	10	否 [2]
每日發行時間	time-of-day	15:30:00	(無)	否 [2]
輪詢後陳述式	post-poll-stmt	DELETE FROM direct.direct_process	(無)	否
批次大小	batch-size	16	1	否
活動訊號間隔 (以分鐘爲單位)	pub-heartbeat-interval	10	0	否

[1] 已觸發發行模式的必要項目。[2] 這些參數互斥。

發行參數分爲四個主要子類別：

- 「未分類參數」，第 60 頁
- 「已觸發發行參數」，第 70 頁
- 「無觸發發行參數」，第 72 頁
- 「輪詢參數」，第 72 頁

4.5.1 未分類參數

- 「停用發行者？」，第 67 頁
- 「發行模式」，第 68 頁

停用發行者？

「停用發行者？」參數可指定是否停用 「發行者」通道。停用時，「發行者」通道不會查詢資料庫事件。與停用訂閱者？參數不同，您仍然可以在 「發行者」通道上發出資料庫查詢，以促進不同的發行演算法進行。

此參數設爲布林值 True 時，「發行者」通道會停用。此參數設爲布林值 False 時，「發行者」通道會啓用。

表格 ***4-45*** 停用發行者？：內容

內容	值
標籤名稱	disable
必要？	no
預設值	0 (否)
正確的值	1、yes、true (是) 0、no、false (否)
與綱要相依	True

停用陳述式層級鎖定？

「停用陳述式層級鎖定？」參數可指定在執行每個 SQL 陳述式之前，是否在此通道上明確地鎖定資料庫資源。此參數僅在啓用陳述式層級鎖定？參數設爲布林值 True 時才處於使用中。

此參數設爲布林值 True 時，資料庫資源會明確鎖定。此參數設爲布林值 False 時，資料庫資源則不會明確地鎖定。

表格 ***4-46*** 停用陳述式層級鎖定？：內容

內容	值
標籤名稱	disable-locking
必要？	no
預設值	0 (否)
正確的值	1、yes、true (是) 0、no、false (否)
與綱要相依	True

發行模式

「發行模式」參數可指定所使用的發行演算法。

設爲 1 (已觸發) 時，「發行者」通道會輪詢事件的事件記錄表格。設爲 2 (無觸發) 時，「發行者」通道會攫取同步化綱要中的所有表格 / 檢視窗的變更，並會合成事件。

下表列出此參數的內容：

表格 ***4-47*** 發行模式：內容

內容	值
標籤名稱	publication-mode
必要？	no
預設值	1 (已觸發)

內容	值
正確的值	1 (已觸發) 2 (無觸發)
與綱要相依	True

啓用未來事件處理？

針對已觸發發行，「啓用未來事件處理？」可指定事件記錄表格中的列是依插入順序 (record_id 欄) 還是依時間 (event_time 欄) 排序和處理的。

此參數設爲布林值 True 時，事件記錄表格中的列會依插入順序發行。此參數設爲布林值 False 時，事件記錄表格中的列會依時間順序發行。

針對無觸發的發行，「啓用未來事件處理」可指定是否每個事件都發行資料庫本地時間。此額外資訊可以用於強制重試未來日期的事件。若要使其正常運作，在每個使用此功能的邏輯資料庫類別中都必須包含指定事件處理時間的欄，並要將此欄做爲僅通知屬性置於「發行者」過濾器中。請參閱附錄 H 「資料庫描述元 DTD」，第 161 頁 。

資料庫本地時間會做爲每個 XDS 事件 (例如新增、修改、刪除) 的屬性發行。屬性名稱是 jdbc:database-local-time，其中 jdbc 名稱空間字首會與 urn:dirxml:jdbc 結合。此格式是 java.sql.Timestamp 的 Java 字串表示：yyyy-mm-dd hh:mm:ss.fffffffff。根據「時間語法」參數的值，指出事件處理時間的值可以做爲整數、標準字串或 Java 字串來發行。請參閱「時間語法」，第 42 頁。

無論發行語法爲何，此值都可以加以剖析，並與資料庫本地時間值相比較。下表將時間語法映射至適當的剖析方法。

表格 ***4-48*** 將時間語法映射至剖析方法

時間語法	剖析方法
整數	java.sql.Timestamp.valueOf(java.lang.String):java.sql.Timestamp (http://java.sun.com/j2se/1.5.0/docs/api/java/sql/Timestamp.html)
標準字串	com.novell.nds.dirxml.driver.jdbc.db.DSTime(java.lang.String, java.lang.String, java.lang.String, java.lang.String)
java 字串	java.sql.Timestamp.valueOf(java.lang.String):java.sql.Timestamp (http://java.sun.com/j2se/1.5.0/docs/api/java/sql/Timestamp.html)

兩種時間值都是以一般 Timstamp 物件表示時，可以使用下列方法比較它們：

- com.novell.nds.dirxml.driver.jdbc.db.TimestampUtil.before(java.sql.Timestamp, java.sql.Timestamp):boolean
- com.novell.nds.dirxml.driver.jdbc.db.TimestampUtil.after(java.sql.Timestamp, java.sql.Timestamp):boolean

附錄 H 「資料庫描述元 DTD」，第 161 頁 中提供範例規則。

此參數設爲布林值 True 時，每個事件都會發行本地資料庫時間。此參數設爲布林值 False 時，會省略此資訊。

下表列出此參數的內容：

表格 ***4-49*** 啓用未來事件處理？：內容

內容	值
標籤名稱	handle-future-events
必要？	no
預設值	0 (否)
正確的值	1、yes、true (是) 0、no、false (否)
與綱要相依	True

4.5.2 已觸發發行參數

Driver for JDBC 可以使用四種已觸發發行參數中的任何一種。

- 「事件記錄表格名稱」，第 70 頁
- 「刪除已處理的列？」，第 70 頁
- 「允許迴路？」，第 71 頁

事件記錄表格名稱

「事件記錄表格名稱」參數指定儲存發行事件的事件記錄表格名稱。

此處指定的表格必須符合 「事件記錄表格」，第 87 頁的定義。

使用 「表格 / 檢視窗名稱」，第 47 頁時，您很可能需要對此表格名稱進行明確的綱要限定。使用 「綱要名稱」，第 46 頁時，會使用該綱要名稱對此表格名稱進行隱含的綱要限定。如果此表格位於非隱含綱要的綱要中，則必須對其進行綱要限定。

下表列出此參數的內容：

表格 ***4-50*** 事件記錄表格名稱：內容

內容	值
標籤名稱	log-table
必要？	no[1]
區分大小寫？	請參閱 「未分隔的識別碼區分大小寫」，第 126 頁。
範例值	eventlog
預設值	(無)
與綱要相依	True

[1] 如果 「發行模式」，第 68 頁設爲 1 (已觸發發行)，則此參數是必要項目。

刪除已處理的列？

「刪除已處理的列？」參數指定是否從事件記錄表格刪除已處理的列。

此參數設爲布林值 True 時，會刪除已處理的列。此參數設爲布林值 False 時，會更新已處理列的 status 欄位值。

若要減輕已處理列留存在事件記錄表格中對於效能所造成的影響，建議您定期將這些列移入歷程表格。請進行下列其中一項操作：

- 透過 「輪詢後陳述式」，第 73 頁參數呼叫清理預存程序。
- 將刪除之前觸發置於事件記錄表格中，以攔截針對事件記錄表格執行的刪除事件，並在已刪除列從事件記錄表格中刪除之前，將其移至歷程表格中。

下表列出此參數的內容：

表格 ***4-51*** 刪除已處理的列？：內容

內容	值
標籤名稱	delete-from-log
必要？	no
預設值	0 (否)
正確的值	1、yes、true (是) 0、no、false (否)
與綱要相依	True

附註：除非定期從事件記錄表格移除已處理的列，否則將此參數設爲布林值 False 會降低發行效能。

允許迴路？

「允許迴路？」參數指定是否應該發行由驅動程式資料庫使用者帳戶導致的事件。

此參數設爲布林值 True 時，會發行迴路事件。此參數設爲布林值 False 時，會忽略迴路事件。

下表列出此參數的內容：

表格 ***4-52*** 允許迴路？：內容

內容	值
標籤名稱	allow-loopback
必要？	no
預設值	0 (否)
正確的值	1、yes、true (是) 0、no、false (否)
與綱要相依	True

附註：將此參數設爲布林值 True 可能會降低效能，因爲可能會發行無關的事件。

4.5.3 無觸發發行參數

Driver for JDBC 可以使用一個無觸發發行參數。然而，「狀態目錄」，第 44 頁參數也會影響無觸發發行。

啓動選項

「啓動選項」參數指定在啓動無觸發發行者時發生的事件。

表格 ***4-53*** 啓動選項：設定和結果

設定	結果
1	假設已變更和重新發行所有物件。
2	忽略過去和目前的變更。
3	發行所有過去和目前的變更。

下表列出此參數的內容：

表格 ***4-54*** 啓動選項：內容

內容	值
標籤名稱	startup-option
必要？	no
預設值	1 (處理所有變更)
正確的值	1 (重新同步化所有物件) 2 (僅處理未來變更) 3 (處理所有變更)
與綱要相依	True

重要：無觸發發行中的下列變更可以強制進行重新同步化：

- 使用無觸發發行時，變更驗證網路位置參數中非 URL 內容的任何項目都會強制重新同步化所有物件。
- 使用無觸發發行時，變更驗證網路位置參數的值會強制重新同步化所有物件。
- 變更「狀態目錄」參數值。

4.5.4 輪詢參數

- 「輪詢間隔 (以秒爲單位)」，第 73 頁
- 「每日發行時間」，第 73 頁
- 「輪詢後陳述式」，第 73 頁
- 「批次大小」，第 74 頁
- 「活動訊號間隔 (以分鐘爲單位)」，第 75 頁

輪詢間隔 (以秒爲單位)

「輪詢間隔 (以秒爲單位)」參數指定輪詢週期間無活動的秒數。

下表列出此參數的內容：

表格 *4-55* 輪詢間隔 *(以秒爲單位)*：內容

內容	值
標籤名稱	polling-interval
必要？	no
預設值	10 (秒)
正確的值	1-604800 (1 週)
與綱要相依	True

附註：建議您將此值設爲至少 10 秒。

每日發行時間

「每日發行時間」參數指定在每天發行開始的時間。時間視爲表示伺服器本地時間 (執行驅動程式之伺服器上的時間)。

下表列出此參數的內容：

表格 *4-56* 每日發行時間：內容

內容	值
標籤名稱	time-of-day
必要？	no
範例值	13:00:00 (1PM)
預設值	(無)
正確的值	hh:mm:ss (h = 小時、m = 分鐘、s = 秒)
與綱要相依	True

附註：此參數會置換「輪詢間隔 (以秒爲單位)」參數。請參閱「輪詢間隔 (以秒爲單位)」，第 73 頁。

輪詢後陳述式

「輪詢後陳述式」參數指定在每個使用中輪詢週期結束時執行的 SQL 陳述式。使用中輪詢週期是發生部份發行活動的週期。

此參數的主要目的是允許清理發行活動之後的事件記錄表格。

您很可能需要對這些陳述式中參考的所有資料庫物件 (例如，表格、預存程序和函數) 進行明確的綱要限定。

下表列出此參數的內容：

表格 ***4-57*** 輪詢後陳述式：內容

內容	值
標籤名稱	post-poll-stmt
必要？	no
區分大小寫？	請參閱 「未分隔的識別碼區分大小寫」，第 126 頁。
分隔符	分號 (;)
範例值	DELETE FROM direct.direct_process
預設值	(無)
正確的值	(任何正確的 SQL 陳述式集)
與綱要相依	True

批次大小

「批次大小」參數指定單一發行文件中傳送的事件數目。

基本上，批次越大，效能越佳。

- 較大的批次需要較少的網路間往返。
- 單一文件中的事件越多，所需的「發行者」通道與 Identity Manager 引擎間的往返越少 (假設沒有使用查詢回覆事件)。
- 較大的批次會使「發行者」通道與資料庫間的往返數減至最少 (假設協力廠商 JDBC 驅動程式和資料庫支援批次處理)。
- 批次越大，需要對本端檔案系統中狀態文件的認可越少。

  認可的成本也會很高。

此參數會定義上限。「發行者」通道可能會在某些條件下置換指定的值。選擇上限 128，可以將 Java 堆積溢位的可能性降至最低，且會減少驅動程式關閉時「發行者」線串的終止延遲。

下表列出此參數的內容：

表格 ***4-58*** 批次大小：內容

內容	值
標籤名稱	batch-size
必要？	no
預設值	1

內容	值
正確的值	1 到 128
與綱要相依	True

活動訊號間隔 (以分鐘爲單位)

「活動訊號間隔 (以分鐘爲單位)」參數指定 「發行者」通道在傳送活動訊號文件之前可以處於非使用中的分鐘數。實際上，經過的時間會超過指定的分鐘數。也就是說，此參數是定義下限。只有當 「發行者」通道處於非使用中的時間已達指定的分鐘數後，才會傳送活動訊號文件。實際上，傳送的任何發行文件都是活動訊號文件。

下表列出此參數的內容：

表格 ***4-59*** 活動訊號間隔 *(以分鐘爲單位)*：內容

內容	值
標籤名稱	pub-heartbeat-interval
必要？	no
預設值	0
正確的值	0 到 2,147,483,647 (java.lang.Integer.MAX_VALUE)
與綱要相依	False

4.6 追蹤層級

若要查看驅動程式的除錯輸出，請在包含驅動程式例項的驅動程式集上新增從 1 到 7 的 DirXML-DriverTraceLevel 屬性值。此屬性通常會與 DirXML-XSL TraceLevel 屬性混淆。如需驅動程式集追蹤層級的相關資訊，請參閱 《*Novell Identity Manager 3.0* 管理指南》。

驅動程式支援下列七個追蹤層級：

表格 ***4-60*** 受支援追蹤層級

層級	描述
1	最少的追蹤
2	資料庫內容
3	連接狀態、SQL 陳述式、事件記錄
4	詳細輸出
5	資料庫資源配置 / 解除配置、狀態檔案內容
6	JDBC 應用程式介面 (Application Programming Interfaces，API) (叫用的方法、傳遞的引數、傳回值等)
7	協力廠商驅動程式

層級 6 和 7 特別適用於協力廠商驅動程式的除錯。

4.7 設定協力廠商 JDBC 驅動程式的組態

下列指示會協助您設定協力廠商驅動程式的組態。如需特定的組態設定指示，請參閱您的協力廠商驅動程式文件。

- 使用最新版本的驅動程式。
- 可以設定協力廠商驅動程式行爲的組態。

  在很多情況下，可以藉由調整驅動程式的 JDBC URL 內容來解決不相容問題。
- 當您使用國際字元時，必須經常將資料庫所使用的字元編碼明確地指定給協力廠商驅動程式。

  指定方法爲將內容字串附加至驅動程式的 JDBC URL 結尾。

  內容通常由內容關鍵字和字元編碼值組成 (例如，jdbc:odbc:mssql;charSet=Big5)。不同協力廠商驅動程式中的內容關鍵字可能有所不同。

  Sun 定義了可能的字元編碼值。如需相關資訊，請參閱 Sun 的受支援編碼網站 (http://java.sun.com/j2se/1.5.0/docs/guide/intl/encoding.doc.html)。

下表列出最大驅動程式相容性的建議設定。當您在啓始組態設定期間使用不受支援的協力廠商驅動程式時，這些設定非常有用。

表格 ***4-61*** *協力廠商 JDBC 驅動程式的建議設定*

參數名稱	相容性值
同步化過濾器	empty
重複使用陳述式？	0 (否)
使用手動異動？	0 (否)
使用最少的連接？	yes
取回最少中繼資料？	1 (是)
傳回的結果集數目	one

進階組態

5

安裝範例驅動程式組態之後，自定它以供專用。

5.1 綱要映射

下表顯示高層級檢視窗，以說明驅動程式如何將 Novell® Identity Vault 物件映射至資料庫物件。

表格 *5-1* 映射 *Identity Vault* 物件至資料庫物件

Identity Vault 物件	資料庫物件
網路樹	綱要
類別	表格 / 檢視窗
屬性	欄
關聯	主索引鍵

5.1.1 邏輯資料庫類別

邏輯資料庫類別是表格集或檢視窗集，用於代表資料庫中的 eDirectory 類別。邏輯資料庫類別可以由單一檢視窗或一個父表格和零或多個子表格組成。

邏輯資料庫類別的名稱是父表格或檢視窗的名稱。

5.1.2 間接同步化

在間接同步化模型中，驅動程式映射下列項目：

表格 *5-2* 間接同步化中的映射

Identity Vault 物件	資料庫物件
類別	表格
屬性	欄

Identity Vault 物件	資料庫物件
1 個類別	1 個父表格 和 0 或多個子表格
單一值屬性	父表格欄
多值屬性	父表格欄 (存放分隔值) 或 子表格欄 (偏好選項)

映射 **eDirectory** 類別至邏輯資料庫類別

在下列範例中，邏輯資料庫類別 usr 由下列項目組成：

- 一個父表格 usr
- 兩個子表格：usr_phone 和 usr_faxno。

邏輯類別 usr 會映射至 eDirectory 類別「使用者」。

```
CREATE TABLE indirect.usr ( idu         INTEGER  NOT NULL, fname
VARCHAR2(64), lname       CHAR(64), pwdminlen   NUMBER(4), pwdexptime
DATE, disabled    NUMBER(1), username    VARCHAR2(64), loginame
VARCHAR2(64), photo       LONG RAW, manager     INTEGER, CONSTRAINT
pk_usr_idu    PRIMARY KEY (idu), CONSTRAINT fk_usr_manager FOREIGN KEY
(manager)         REFERENCES indirect.usr(idu) )


CREATE TABLE indirect.usr_phone ( idu      INTEGER       NOT NULL,
phoneno  VARCHAR2(64)  NOT NULL, CONSTRAINT fk_phone_idu FOREIGN KEY
(idu)         REFERENCES indirect.usr(idu) )


CREATE TABLE indirect.usr_fax ( idu    INTEGER       NOT NULL, faxno
VARCHAR2(64)  NOT NULL, CONSTRAINT fk_fax_idu FOREIGN KEY (idu)
        REFERENCES indirect.usr(idu) )


<rule name="Schema Mapping Rule"> <attr-name-map> <class-name> <nds-
name>User</nds-name> <app-name>indirect.usr</app-name> </class-name>
<attr-name class-name="User"> <nds-name>Given Name</nds-name> <app-
name>fname</app-name> </attr-name> <attr-name class-name="User"> <nds-
name>Surname</nds-name> <app-name>lname</app-name> </attr-name> <attr-
name class-name="User"> <nds-name>Password Expiration Time</nds-name>
<app-name>pwdexptime</app-name> </attr-name> <attr-name class-
name="User"> <nds-name>jpegPhoto</nds-name> <app-name>photo</app-name>
</attr-name> <attr-name class-name="User"> <nds-name>manager</nds-
name> <app-name>manager</app-name> </attr-name> <attr-name class-
name="User"> <nds-name>Password Minimum Length</nds-name> <app-
name>pwdminlen</app-name> </attr-name> <attr-name class-name="User">
```

```
<nds-name>Facsimile Telephone Number</nds-name> <app-
name>usr_fax.faxno</app-name> </attr-name> <attr-name class-
name="User"> <nds-name>Telephone Number</nds-name> <app-
name>usr_phone.phoneno</app-name> </attr-name> <attr-name class-
name="User"> <nds-name>Login Disabled</nds-name> <app-name>disabled</
app-name> </attr-name> </attr-name-map> </rule>
```

父表格

父表格是具有明確主索引鍵條件約束的表格，包含一或多個欄。在父表格中，需要明確的主索引鍵條件約束，以使驅動程式瞭解在關聯值中要併入的欄位。

```
CREATE TABLE indirect.usr ( idu  INTEGER  NOT NULL, -- ... CONSTRAINT
pk_usr_idu PRIMARY KEY (idu) )
```

下表包含表格 indirect.usr 的範例資料。

idu	fname	lname
1	John	Doe

此列的結果關聯爲

idu=1,table=usr,schema=indirect

附註：關聯值中的資料庫識別碼大小寫會在執行時期從資料庫中繼資料動態地決定。

父表格欄

父表格欄只能包含一個值。就這一點而論，它們非常適於映射單一值 eDirectory 屬性，例如將單一值 eDirectory 屬性 Password Minimum Length 映射至單一值父表格欄 pwdminlen。

系統會隱含地在父表格欄前面加上綱要名稱和父表格的名稱。因此，不必明確地在父表格欄前面加上表格名稱。例如，在綱要映射方面，indirect.usr.fname 等同於 fname。

```
<rule name="Schema Mapping Rule"> <attr-name-map> <class-name> <nds-
name>User</nds-name> <app-name>indirect.usr</app-name> </class-name>
<attr-name class-name="User"> <nds-name>Given Name</nds-name> <app-
name>fname</app-name> </attr-name> </attr-name-map> </rule>
```

大型二進位和字串資料類型通常應該映射至父表格欄。若要映射至子表格欄，則必須可以在 SQL 陳述式中比較資料類型。在 SQL 陳述式中通常無法比較大型的資料類型。

如果發生下列狀況，則可以將大型二進位和字串資料類型映射至子表格欄：

- 這些類型上的每個 <remove-value> 事件在規則中都轉換爲 <remove-all-values> 元素
- 每個 <remove-value> 事件後面都接著一個 <add-value> 元素

子表格

子表格是在其父表格主索引鍵上具有外部索引鍵條件約束的表格，該條件約束會將兩個表格連結在一起。包含子表格外部索引鍵的欄名稱可以與父表格主索引鍵中的欄名稱不同。

下列範例顯示父表格 usr 和子表格 usr_phone 與 usr_faxno 之間的關係：

```
CREATE TABLE indirect.usr (     idu  INTEGER  NOT NULL, -- ...
CONSTRAINT pk_usr_idu  PRIMARY KEY (idu) )


CREATE TABLE indirect.usr_phone ( idu      INTEGER       NOT NULL,
phoneno  VARCHAR2(64)  NOT NULL, CONSTRAINT fk_phone_idu FOREIGN KEY
(idu)          REFERENCES indirect.usr(idu) )


CREATE TABLE indirect.usr_fax ( idu    INTEGER       NOT NULL, faxno
VARCHAR2(64)  NOT NULL,   CONSTRAINT fk_fax_idu FOREIGN KEY (idu)
        REFERENCES indirect.usr(idu) )
```

附註：在子表格中，約束所有欄爲 NOT NULL。

子表格中的第一個受約束欄會識別父表格。在上述範例中，子表格 usr_phone 中受約束的欄是 idu。此欄的唯一用途是將表格 usr_phone 與 usr 相關聯。由於受約束的欄不包含任何有用的資訊，因此從發行觸發和「綱要映射」規則省略它們。

不受約束的欄是相關的欄。它代表單一、多值屬性。在上述範例中，不受約束的欄爲 phoneno 和 faxno。由於不受約束的欄可以存放多值，因此它們非常適於映射多值 eDirectory 屬性 (例如，將多值 eDirectory 屬性 Telephone Number 映射至 usrphone.phoneno)。

下表包含 indirect.usr_phone 的範例資料。

表格 ***5-3*** 範例資料

idu	phoneno
1	111-1111
1	222-2222

與父表格欄相似，子表格欄前面會隱含地加上綱要名稱。然而，與父表格欄不同的是，必須明確地在子表格欄名稱前加上子表格名稱 (例如，usr_phone.phoneno)。否則，驅動程式會隱含地將欄 phoneno (父表格欄) 解譯爲 usr.phoneno，而非子表格欄 usr_phone.phoneno。

```
<rule name="Schema Mapping Rule"> <attr-name-map> <class-name> <nds-
name>User</nds-name> <app-name>indirect.usr</app-name> </class-name>
<attr-name class-name="User"> <nds-name>Facsimile Telephone Number</
nds-name> <app-name>usr_fax.faxno</app-name> </attr-name> <attr-name
class-name="User"> <nds-name>Telephone Number</nds-name> <app-
name>usr_phone.phoneno</app-name> </attr-name> </attr-name-map> </
rule>
```

附註：將每個多值 eDirectory 屬性映射至不同的子表格。

參考屬性

您可以使用外部索引鍵條件約束來代表資料庫中的參考內含項目。參考屬性是邏輯資料庫類別中的欄，它們參考相同邏輯資料庫類別或其他邏輯資料庫類別中父表格的主索引鍵欄。

單一值參考屬性

您可以透過單一值父表格欄來關聯兩個父表格。此欄必須具有指向其他父表格主索引鍵的外部索引鍵條件約束。下列範例將單一父表格 usr 與其自身相關聯：

```
CREATE TABLE indirect.usr ( idu      INTEGER  NOT NULL, -- ... manager
INTEGER, CONSTRAINT pk_usr_idu     PRIMARY KEY (idu), CONSTRAINT
fk_usr_manager FOREIGN KEY (manager)          REFERENCES
indirect.usr(idu) )
```

附註：單一值參考欄應該可爲 Null。

```
<rule name="Schema Mapping Rule"> <attr-name-map> <class-name> <nds-
name>User</nds-name> <app-name>indirect.usr</app-name> </class-name>
<attr-name class-name="User"> <nds-name>manager</nds-name> <app-
name>manager</app-name> </attr-name> </attr-name-map> </rule>
```

上述範例的解譯爲，每個使用者只可以具有一個自身是使用者的管理員。

多值參考屬性

您可以透過一般子表格來關聯兩個父表格。此子表格必須具有由指向其他父表格主索引鍵之外部索引鍵所約束的欄。下列範例透過一般子表格 member 將兩個父表格 usr 和 grp 相關聯。

```
CREATE TABLE indirect.usr ( idu  INTEGER  NOT NULL, -- ... CONSTRAINT
pk_usr_idu PRIMARY KEY (idu) )


CREATE TABLE indirect.grp ( idg  INTEGER  NOT NULL, -- ... CONSTRAINT
pk_grp_idg PRIMARY KEY (idg) )


CREATE TABLE indirect.grp_member ( idg  INTEGER  NOT NULL, idu  INTEGER
NOT NULL, CONSTRAINT fk_member_idg FOREIGN KEY (idg)      REFERENCES
indirect.grp(idg),    CONSTRAINT fk_member_idu FOREIGN KEY (idu)
REFERENCES indirect.usr(idu) )
```

附註：將子表格中的所有欄約束爲 NOT NULL。

```
<rule name="Schema Mapping Rule"> <attr-name-map> <class-name> <nds-
name>Group</nds-name> <app-name>indirect.grp</app-name> </class-name>
```

```
<class-name> <nds-name>User</nds-name> <app-name>indirect.usr</app-
name> </class-name> <attr-name class-name="Group"> <nds-name>Member</
nds-name> <app-name>grp_member.idu</app-name> </attr-name> </attr-
name-map> </rule>
```

子表格中的第一個受約束的欄決定子表格 grp_member 所屬的邏輯資料庫類別。在上述範例中，grp_member 是邏輯資料庫類別 grp 的一部份。grp_member 是 grp 的正確子代。子表格中第二個受約束的欄是多值參考屬性。

在下列範例中，會將受約束欄的順序反向，以使 grp_member 成爲類別 usr 的一部份。爲了更加準確地反映關係，會將表格 grp_member 重新命名爲 usr_mbr_of。

```
CREATE TABLE indirect.usr ( idu  INTEGER  NOT NULL, -- ... CONSTRAINT
pk_usr_idu PRIMARY KEY (idu) )


CREATE TABLE indirect.grp ( idg  INTEGER  NOT NULL, -- ... CONSTRAINT
pk_grp_idg PRIMARY KEY (idg) )


CREATE TABLE indirect.usr_mbr_of ( idu  INTEGER  NOT NULL, idg  INTEGER
NOT NULL, CONSTRAINT fk_mbr_of_idu FOREIGN KEY (idu)
        REFERENCES indirect.usr(idu) ON DELETE CASCADE,    CONSTRAINT
fk_mbr_of_idg FOREIGN KEY (idg)         REFERENCES indirect.grp(idg)
ON DELETE CASCADE )


<rule name="Schema Mapping Rule"> <attr-name-map> <class-name> <nds-
name>Group</nds-name> <app-name>indirect.grp</app-name> </class-name>
<class-name> <nds-name>User</nds-name> <app-name>indirect.usr</app-
name> </class-name> <attr-name class-name="User"> <nds-name>Group
Membership</nds-name> <app-name>usr_mbr_of.idg</app-name> </attr-name>
</attr-name-map> </rule>
```

在不瞭解欄位置的資料庫中 (例如，DB2/AS400)，順序是藉由依字串或十六進位值排序欄名稱來決定。如需其他資訊，請參閱「欄名稱排序方式」，第 58 頁。

一般而言，只需要做爲一個或另一個類別來同步化雙向、多值、參考屬性，而非同時做爲兩個類別。如果您想要同時同步化兩種類別的參考屬性，請爲兩個類別各建構一個子表格。例如，如果您想要同步化 eDirectory 屬性「群組成員資格」和「成員」，您需要兩個子表格。

實際上，當您同步化「使用者」和「群組」類別時，我們建議您同步化類別「使用者」的「群組成員資格」屬性，而非類別「群組」的「成員」屬性。同步化使用者的群組成員資格通常比同步化群組的所有成員更有效率。

5.1.3 直接同步化

在直接同步化模型中，驅動程式映射下列項目：

表格 *5-4* 直接同步化中的映射

Identity Vault 物件	資料庫物件
類別	檢視窗
屬性	檢視欄
類別	檢視窗
單一值屬性	檢視欄
多值屬性	檢視欄

檢視窗的更新功能在資料庫之間會有所不同。大部份資料庫可讓檢視窗在包括單一基礎資料表時更新 (也就是說，它們不會結合多個表格)。如果檢視窗爲完全唯讀，則它們無法用於訂閱。部份資料庫允許在取代觸發的檢視窗上定義更新邏輯，這可讓檢視窗結合多個基礎資料表，並且仍可更新。

如需支援取代觸發的資料庫清單，請參閱 「資料庫功能」，第 124 頁。無論資料庫功能爲何，都可以使用內嵌式 SQL 模擬取代觸發邏輯。請參閱 「虛擬觸發」，第 99 頁。

檢視欄中繼識別碼

檢視窗是邏輯表格。檢視窗實際上並不存在於資料庫中，這與表格不同。這樣，檢視窗通常無法具有傳統主索引鍵 / 外部索引鍵條件約束。爲了模擬這些建構元，Driver for JDBC 會在檢視欄名稱中內嵌條件約束和其他中繼資料。這些條件約束和傳統約束的差異在於前者不會在資料庫層級強制執行。它們是應用程式層級的建構元。

例如，若要對驅動程式識別建構關聯值時要使用的欄位，請將主索引鍵條件約束置於父表格上。結果是檢視窗會使用 pk_ (不區分大小寫) 做爲一或多個欄名稱的字首。

下表列出可以在檢視欄名稱中內嵌的條件約束字首。

表格 *5-5* 條件約束字首

條件約束字首 (不區分大小寫)	解譯
pk_	主索引鍵
fk_	外部索引鍵
sv_	單一值
mv_	多值

下列範例檢視窗包含所有這些條件約束字首：

```
CREATE VIEW direct.view_usr ( pk_idu,             -- primary key
column; implicitly single-valued sv_fname,          -- single-valued
column mv_phoneno,        -- multi-valued column fk__idu__manager,  --
self-referential foreign key column; refers                          --
to primary key column idu in view_usr;                          --
implicitly single-valued fk_mv__idg__mbr_of -- extra-referential
foreign key column; refers                          --   to primary key
```

```
column idg in view_grp;                          --   multi-valued ) AS -
- ...


CREATE VIEW direct.view_grp ( pk_idg,           -- primary key column;
implicitly single-valued fk_mv__idu__mbr    -- extra-referential
foreign key column; refers --   to primary key column idu in view_usr;
--   multi-valued ) AS -- ...
```

BNF

檢視欄中繼識別碼的 BNF (Backus Naur Form (http://cui.unige.ch/db-research/Enseignement/analyseinfo/AboutBNF.html)) 表示：

```
<view-column-name> ::= [<meta-info>] <column-name>

<column-name> ::= <legal-unquoted-database-identifier> <meta-info> ::=
<referential> | <non-referential>

<non-referential> ::= [<single-value> | <multiple-value>]

<single-value> ::= "sv_"

<multiple-value> ::= "mv_"

<referential> ::= <primary-key> | <foreign-key>

<primary-key> ::= "pk_" [<single-value>] [<column-group-id>]
[<referenced-column-name>]


<column-group-id> ::= <non-negative-integer> "_"

<referenced-column-name> ::= "_" <column-name> "__"

<foreign-key> ::= "fk_" [<non-referential>] [<column-group-id>]
<referenced-column-name>
```

標準化格式

在預設狀態下，所有的檢視欄名稱都是單一值。因此，在檢視欄名稱中明確地指定 sv_ 字首是多餘的。例如，sv_fname 和 fname 是同一欄名稱的同等格式。

同時，主索引鍵欄名稱隱含地參考它們本身。因此，指定參考的欄名稱是多餘的。例如，pk_idu 等同於 pk__idu__idu。

Driver for JDBC 使用檢視窗中繼識別碼的兩種標準化格式：

- 資料庫原始格式

  資料庫原始格式是資料庫中宣告的欄名稱。此格式通常遠比綱要映射格式詳細，並且包含所有必要的中繼資訊。

- 綱要映射格式

  在驅動程式傳回應用程式綱要時，會傳回綱要映射格式。因爲資料庫原始格式中所包含的所多中繼資訊是以 XDS XML 而非識別碼代表，所以此格式比資料庫原始格式更加簡明。

  參考字首 pk_ 和 fk_ 是以綱要映射格式所保留的唯一中繼資訊。此限制會確保反向相容性。

下表提供每個格式的範例：

表格 ***5-6*** 檢視窗中繼識別碼的標準化格式範例

資料庫原始格式	綱要映射格式
pk_idu	pk_idu
sv_fname	fname
mv_phoneno	phoneno
fk_mv__idg__mbr_of	fk_mbr_of

同等格式

不含中繼資訊的檢視欄名稱稱爲其「有效的」名稱，這類似於目錄物件「有效的」權限。對於驅動程式而言，在決定檢視窗欄名稱等同時，在預設狀態下會與中繼資訊無關。例如，pk_idu 等同於 idu，而 fk_mv__idg__mbr_of 等同於 mbr_of。檢視窗中繼欄識別碼的任何不同格式都可以在執行時間傳遞至驅動程式。由於反向相容性原因，中繼資訊可以視爲有效檢視欄名稱的一部份。請參閱 「啓用中繼識別碼支援？」，第 55 頁。

主索引鍵欄

主索引鍵欄名稱在同步化綱要的所有檢視窗中必須是唯一的。

綱要映射

檢視窗和檢視欄的綱要映射慣例與父表格和父表格欄使用的綱要映射慣例是等同的。

5.1.4 同步化主索引鍵欄

當資料庫爲主索引鍵欄的授權來源時，通常會省略「發行者」和「訂閱者」過濾器、「綱要映射」規則和發行觸發的欄。

當 Identity Vault 爲主索引鍵欄的授權來源時，併入「訂閱者」過濾器和「綱要映射」規則中的欄，而省略「發行者」過濾器和發行觸發的欄。同時，建議將 GUID 而非公用名稱 (CN) 用做主索引鍵。公用名稱 (CN) 是多值屬性並且可以變更。GUID 具有單一值並且是靜態的。

5.1.5 同步化多個類別

同步化多個 eDirectory 類別時，針對不同的父表格或檢視窗同步化每個類別。每個邏輯資料庫類別都必須具有唯一的主索引鍵欄名稱。「發行者」通道使用此公用欄名來識別單一邏輯

資料庫類別相關之事件記錄表格中的所有列。例如，邏輯資料庫類別 usr 和 grp 具有唯一的主索引鍵欄名稱。

```
CREATE TABLE usr ( idu   INTEGER       NOT NULL, lname VARCHAR2(64)
NOT NULL, --... CONSTRAINT pk_usr_idu PRIMARY KEY(idu) );


CREATE TABLE grp ( idg  INTEGER  NOT NULL, --... CONSTRAINT pk_grp_idg
PRIMARY KEY(idg) );
```

5.1.6 將多值屬性映射至單一值資料庫欄位

在預設狀態下，驅動程式會假設所有映射至父表格欄或檢視欄的 eDirectory 屬性都具有單一值。由於驅動程式不瞭解 eDirectory 綱要，因此它無法瞭解 eDirectory 屬性具有單一值還是多值。相應地，會用相同方式處理多值和單一值屬性映射。

驅動程式針對單一值父表格或檢視欄實作最近使用過的 (Most Recently Touched，MRT) 演算法。最近使用過的 (MRT) 演算法可確保將最近新增或刪除的屬性值儲存在資料庫中。如果有疑問的屬性具有單一值，則該演算法已足夠。

如果屬性具有多值，則演算法會產生一些不必要的結果。在從多值屬性刪除值時，它所映射至的資料庫欄位會設爲 NULL，並且在新增其他值之前保持爲 NULL。對此不必要行爲的偏好解決方案是擴充 eDirectory 綱要，以便只有單一值屬性映射到父表格或檢視欄。

其他解決方案如下：

- 若爲間接同步化，將每個多值屬性映射至其自己的子表格。
- 若同時爲直接或間接同步化，在將多值插入表格或檢視欄之前，使用規則將它們分隔。
- 藉由使用 com.novell.nds.indirect.driver.jdbc.util.MappingPolicy 類別中提供的方法，實作樣式表中每個複製本規則的第一個或最後一個值。在每個複製本的第一個值 (FPR) 規則下，一律同步化 eDirectory 複製本上的第一個屬性值。在每個複製本的最後一個值 (LPR) 規則下，一律同步化複製本上的最後一個屬性值。藉由使用全域組態值，您可以設定範例驅動程式組態，以使用 FPR 或 LPR 映射規則。多值到單一值屬性映射規則都包含在「訂閱者指令轉換」規則容器中。範例驅動程式組態會將多值 eDirectory 屬性 Given Name 和 Surname 分別映射至單一值欄 fname 和 lname。

5.2 XDS 事件至 SQL 陳述式的映射

下表概述「訂閱者」通道如何將 XDS 事件映射到資料操作語言 (DML) SQL 陳述式，以進行間接同步化：

表格 ***5-7*** 映射 *XDS* 事件以間接同步化

XML 事件	SQL 同等項目
<add>	0 或多個 select 陳述式，取決於相符規則 所有單一值 <add-attr> 元素具有 1 個父表格 insert 陳述式 0 或 1 個預存程序 / 函數呼叫，在父表格 insert 陳述式之前或之後取回主索引鍵值 每個多值 <add-attr> 元素都具有的 1 個子表格 insert 陳述式

XML 事件	SQL 同等項目
<modify>	每個單一值 <add-value> 或 <remove-value> 元素都具有的 1 個父表格 update 陳述式 每個多值 <add-value> 元素都具有的 1 個子表格 insert 陳述式 每個 <remove-value> 元素都具有的 1 個子表格 delete 陳述式
<delete>	1 個父表格 delete 陳述式 每個子表格都具有的 1 個 delete 陳述式
<query>	1 個父表格 select 陳述式 每個子表格都具有的 1 個 select 陳述式
<move> <rename> <modify-password> <check-object-password>	0 個陳述式，除非結合內嵌式 SQL 陳述式

下表概述「訂閱者」通道如何將 XDS 事件映射至資料操作語言 (DML) SQL 陳述式，以進行直接同步化：

表格 ***5-8*** 映射 *XDS* 事件以直接同步化

XML 事件	SQL 同等項目
<add>	0 或多個 select 陳述式，取決於相符規則 所有單一值 <add-attr> 元素都具有的 1 個檢視窗 insert 陳述式 0 或 1 個預存程序 / 函數呼叫，在檢視窗 insert 陳述式之前或之後取回主索引鍵值 每個多值 <add-attr> 元素都具有的 1 個檢視窗 insert 陳述式
<modify>	每個單一值 <add-value> 或 <remove-value> 元素都具有的 1 個檢視窗 update 陳述式 每個多值 <add-value> 元素都具有的 1 個檢視窗 insert 陳述式 每個 <remove-value> 元素都具有的 1 個檢視窗 delete 陳述式
<delete>	1 個檢視窗 delete 陳述式
<query>	1 個檢視窗 select 陳述式
<move> <rename> <modify-password> <check-object-password>	0 個陳述式，除非結合內嵌式 SQL 陳述式

5.3 事件記錄表格

事件記錄表格會儲存發行事件。本節討論事件記錄表格的結構和功能。

您可以自定事件記錄表格及其欄的名稱，以避免與保留的資料庫關鍵字相衝突。不過，其欄的順序、數目和資料類型都是固定的。在不瞭解欄位置的資料庫中，順序會由「欄名稱排序方式」參數決定。請參閱「欄名稱排序方式」，第 58 頁。

此表格中的事件可以依插入順序 (record_id 欄) 或時間順序 (event_time 欄) 排序。依時間排序事件會延遲事件處理。若要依時間排序發行事件，請將「啓用未來事件處理」參數設爲布林值 True。請參閱「啓用未來事件處理？」，第 69 頁。

- 「事件記錄欄」，第 88 頁
- 「事件類型」，第 90 頁

5.3.1 事件記錄欄

本節會描述事件記錄表格中的欄。欄依位置排序。

1. record_id

   record_id 欄用於唯一識別事件記錄表格中的列並排序發行事件。此欄必須包含循序、遞增、唯一的正整數值。record_id 值之間的不連續不再會提前結束輪詢週期。

2. status

   status 欄會指出給定列的狀態。下表列出允許的值：

表格 ***5-9*** *Status* 欄的允許值

字元值	解譯
N	新
S	成功
W	警告
E	錯誤
F	嚴重錯誤

若要進行處理，所有插入至事件記錄表格中的列都必須具有 status 值 N。狀態字元的其他部份只會由「發行者」通道用來指定已處理列。所有其他字元都會保留供以後使用。

附註：狀態值區分大小寫。

3. event_type

   此欄中的值必須在 1 到 8 之間。所有其他數值會保留供以後使用。

   下表描述每個事件類型：

表格 ***5-10*** 事件類型

事件類型	解譯
1	插入欄位
2	更新欄位
3	更新欄位 (移除所有值)
4	刪除列
5	插入列 (查詢回覆)
6	更新列 (查詢回覆)
7	插入欄位 (查詢回覆)
8	更新欄位 (查詢回覆)

如需此欄位的其他資訊，請參閱「事件類型」，第 90 頁。

4. event_time

   此欄做爲 record_id 的取代排序欄使用。它包含事件的有效日期。它不能是 NULL。若要使此欄成爲排序欄，請將「啓用未來事件處理」參數設爲布林值 True。請參閱 「啓用未來事件處理？」，第 69 頁。

5. perpetrator

   此欄會識別啓動事件的資料庫使用者。NULL 值會解譯爲使用者，而不是驅動程式使用者。同樣，會發行具有 NULL 值或不等同於驅動程式資料庫使用者名稱的列。除非「允許迴路發行者」參數設爲布林值 True，否則不會發行值爲驅動程式資料庫使用者名稱的列。請參閱 「允許迴路？」，第 71 頁。

6. table_name

   發生事件之表格或檢視窗的名稱。

7. table_key

   此欄的格式值，在邏輯資料庫類別的所有觸發中都完全相同。此參數的 BNF 或 Backus Naur Form (http://cui.unige.ch/db-research/Enseignement/analyseinfo/AboutBNF.html) 定義如下：

   ```
   <table-key> ::= <unique-row-identifier> {"+"
                   <unique-row-identifier>}

   <unique-row-identifier> ::= <primary-key-column-name> "=" <value>
   ```

   例如，對於本章中所參考的 usr 表格，此欄的值可能是 idu=1。

   對於本章中所參考的 view_usr 檢視窗，此欄的值可能是 pk_empno=1。

   對於假設的複合主索引鍵 (包含多個欄的主索引鍵)，此欄的值可能是 *pkey1=value1+pkey2=value2*。

   附註：如果位於 table_key 欄位中的主索引鍵值包含任何特殊字元 {, ; ' + " = \ < >}，其中揍 ' 和 挺 ' 包含特殊字元集，則使用雙引號分隔該值。當包含在一對雙引號內時，您還需要將雙引號字元 " 逸出爲 \"，以及將常值逸出字元 \ 字元逸出爲 \\。

   對於包含特殊字元的假設主索引鍵，此欄的值可能是 pkey=", ; ' + \" = \\ < >" (請注意雙引號和逸出字元)。

   附註：填塞或格式化上的差異可能會導致事件處理發生問題。由於效能原因，從數值移除所有不需要的空白。例如，"idu=1" 就優先於 "idu= 1" (請注意 "idu= 1" 中的空格)。

   ---

8. column_name

   已變更的欄名。此欄僅用於每個欄位 (1-3, 7-8) 事件類型。不過，它必須一直呈現在事件記錄表格中。如果遺失，則「發行者」通道無法啓動。

9. old_value

   欄位的舊值。此欄僅用於每個欄位、非查詢回覆事件類型 (1-3)。不過，它必須一直呈現在事件記錄表格中。如果遺失，則「發行者」通道無法啓動。

10. new_value

    欄位的新值。此欄僅用於每個欄位、非查詢回覆事件類型 (1-3)。不過，它必須一直呈現在事件記錄表格中。如果遺失，則「發行者」通道無法啓動。

5.3.2 事件類型

下表描述每個事件類型：

表格 ***5-11*** 事件類型

事件類型	解譯
1	插入欄位
2	更新欄位
3	更新欄位 (移除所有值)
4	刪除列
5	插入列 (查詢回覆)
6	更新列 (查詢回覆)
7	插入欄位 (查詢回覆)
8	更新欄位 (查詢回覆)

事件類型主要分爲四個類別。部份類別會重疊。下表描述每個類別，並指出是成員的事件類型：

表格 ***5-12*** 事件類別和類型

事件類別	事件類型
每個欄位 (屬性)	1, 2, 3, 7, 8
每列 (物件)	4, 5, 6
非查詢回覆	1, 2, 3, 4
查詢回覆	5, 6, 7, 8
每個欄位，非查詢回覆	1, 2, 3
每個欄位，查詢回覆	7, 8
每列，非查詢回覆	4
每列，非查詢回覆	5, 6

一般而言，每個類別的事件類型組合都會使空間、時間、實作複雜度和效能等方面達到最好的平衡。

與每列事件類型相比，每個欄位事件類型較精確、需要較多空間，並且實作起來較複雜。與每個欄位事件類型相比，每列事件較不精確，需要較少空間，並且較易於實作。

與非查詢回覆事件類型相比，查詢回覆事件類型使用的空間較少，但是需要的處理時間較長。與查詢回覆事件類型相比，非查詢回覆事件類型使用的空間較多，但是需要的處理時間較短。

查詢回覆事件類型優於它們的非查詢回覆事件類型。如果相同的欄位或物件記錄查詢回覆事件，則會忽略非查詢回覆事件。例如，如果類型 2 (更新欄位、非查詢回覆) 和 8 (更新欄位、查詢回覆) 的事件記錄在同一欄位上，則會忽略類型 2 事件而記錄類型 8 事件。

而且，查詢回覆列事件類型優於查詢回覆欄位事件類型。例如，如果事件類型 8 (更新欄位、查詢回覆) 和事件類型 6 (更新列、查詢回覆) 的事件記錄在同一物件上，則會忽略類型 8 事件而記錄類型 6 事件。

如果資料庫物件不再存在，則「發行者」會忽略查詢回覆事件。它們取決於在處理時間仍然存在的資料庫物件。因此，已記錄查詢回覆新增和修改 (事件類型 5、6、7、8) 會在它們參考的資料庫物件刪除之後就無效。

下表顯示發行事件類型和「發行者」通道產生之 XDS XML 之間的基本關連性。

表格 ***5-13*** 發行事件類型的基本關連性

事件類型	結果 XDS
插入	<add>
更新	<modify>
刪除	<delete>

下列範例說明由「發行者」通道針對事件產生的 XML，這些事件記錄在每個可能的事件類型之 usr 表格上。

```
CREATE TABLE indirect.usr ( idu   INTEGER  NOT NULL, fname
VARCHAR2(64), photo LONGRAW, --... CONSTRAINT pk_usr_idu PRIMARY
KEY(idu) );
```

下表顯示插入新列之後，usr 的啓始內容：

表格 ***5-14*** *usr* 表格中的插入列

idu	fname	lname	photo
1	Jack	Frost	0xAAAA

下表顯示更新列之後，usr 的目前內容：

表格 ***5-15*** *usr* 表格中的更新列

idu	fname	lname	photo
1	John	Doe	0xBBBB

插入欄位

下表會顯示新列插入至表格 usr 之後，事件記錄表格的內容。欄 photo 的值已進行 Base64 編碼。0xAAAA 的 Base64 編碼同等項目爲 qqo=。

表格 ***5-16*** 事件記錄表格：類型 *1*

event_type	table	table_key	column_name	old_value	new_value
1	usr	idu=1	fname	NULL	Jack
1	usr	idu=1	lname	NULL	Frost
1	usr	idu=1	photo	NULL	qqo=

「發行者」通道會產生下列 XML：

```
<add class-name="usr"> <association>idu=1,table=usr,schema=indirect </
association> <add-attr attr-name="fname"> <value type="string">Jack</
value> </add-attr> <add-attr attr-name="lname"> <value
type="string">Frost</value> </add-attr> <add-attr attr-name="photo">
<value type="octet">qqo=</value> </add-attr> </add>
```

更新欄位

下表顯示更新表格 usr 中的列之後，事件記錄表格的內容。photo 欄的值已進行 Base64 編碼。0xBBBB 的 Base64 編碼同等項目爲 u7s=。

表格 ***5-17*** 事件記錄表格：類型 *2*

event_type	table	table_key	column_name	old_value	new_value
2	usr	idu=1	fname	Jack	John
2	usr	idu=1	lname	Frost	Doe
2	usr	idu=1	photo	qqo=	u7s=

「發行者」通道會產生下列 XML：

```
<modify class-name="usr"> <association>idu=1,table=usr,schema=indirect
</association> <modify-attr attr-name="fname"> <remove-value> <value
type="string">Jack</value> </remove-value> <add-value> <value
type="string">John</value> </add-value> </modify-attr> <modify-attr
attr-name="lname"> <remove-value> <value type="string">Frost</value>
</remove-value> <add-value> <value type="string">Doe</value> </add-
value> </modify-attr> <modify-attr attr-name="photo"> <remove-value>
<value type="octet">qqo=</value> </remove-value> <add-value> <value
type="octet">u7s=</value> </add-value> </modify-attr> </modify>
```

更新欄位 **(** 移除所有值 **)**

下表顯示更新表格 usr 中的列之後，事件記錄表格的內容。photo 欄的值已進行 Base64 編碼。

表格 ***5-18*** 事件記錄表格：類型 *3*

event_type	table	table_key	column_name	old_value	new_value
3	usr	idu=1	fname	Jack	John
3	usr	idu=1	lname	Frost	Doe
3	usr	idu=1	photo	qqo=	u7s=

「發行者」通道會產生下列 XML：

```
<modify class-name="usr"> <association>idu=1,table=usr,schema=indirect
</association> <modify-attr attr-name="fname"> <remove-all-values/>
<add-value> <value type="string">John</value> </add-value> </modify-
attr> <modify-attr attr-name="lname"> <remove-all-values/> <add-value>
<value type="string">Doe</value> </add-value> </modify-attr> <modify-
attr attr-name="photo"> <remove-all-values/> <add-value> <value
type="octet">u7s=</value> </add-value> </modify-attr> </modify>
```

刪除列

下表顯示刪除表格 usr 中的列之後，事件記錄表格的內容。

表格 ***5-19*** 事件記錄表格：類型 *4*

event_type	table	table_key	column_name	old_value	new_value
4	usr	idu=1	NULL	NULL	NULL

「發行者」通道會產生下列 XML：

```
<delete class-name="usr"> <association>idu=1,table=usr,schema=indirect
</association> </delete>
```

插入列 **(** 查詢回覆 **)**

下表顯示在表格 usr 中插入新列之後，事件記錄表格的內容。

表格 ***5-20*** 事件記錄表格：類型 *5*

event_type	table	table_key	column_name	old_value	new_value
5	usr	idu=1	NULL	NULL	NULL

「發行者」通道會產生下列 XML。值會反映表格 usr 的目前內容，而不是啓始內容。

```
<add class-name="usr"> <association>idu=1,table=usr,schema=indirect </
association> <add-attr attr-name="fname"> <value type="string">John</
```

```
value> </add-attr> <add-attr attr-name="lname"> <value
type="string">Doe</value> </add-attr> <add-attr attr-name="photo">
<value type="octet">u7s=</value> </add-attr> </add>
```

更新列 (查詢回覆)

下表顯示更新表格 usr 中的列之後，事件記錄表格的內容。

表格 ***5-21*** 事件記錄表格：類型 *6*

event_type	table	table_key	column_name	old_value	new_value
6	usr	idu=1	NULL	NULL	NULL

「發行者」通道會產生下列 XML。值會反映表格 usr 的目前內容，而不是啓始內容。

```
<modify class-name="usr"> <association>idu=1,table=usr,schema=indirect
</association> <modify-attr attr-name="fname"> <remove-all-values/>
<add-value> <value type="string">John</value> </add-value> </modify-
attr> <modify-attr attr-name="lname"> <remove-all-values/> <add-value>
<value type="string">Doe</value> </add-value> </modify-attr> <modify-
attr attr-name="photo"> <remove-all-values/> <add-value> <value
type="octet">u7s=</value> </add-value> </modify-attr> </modify>
```

插入欄位 (查詢回覆)

下表顯示表格 usr 中插入新列之後，事件記錄表格的內容。因爲不會使用舊值和新值，所以會省略它們。

表格 ***5-22*** 事件記錄表格：類型 *7*

event_type	table	table_key	column_name	old_value	new_value
7	usr	idu=1	fname	NULL	NULL
7	usr	idu=1	lname	NULL	NULL
7	usr	idu=1	photo	NULL	NULL

「發行者」通道會產生下列 XML。值會反映表格 usr 的目前內容，而不是啓始內容。

```
<add class-name="usr"> <association>idu=1,table=usr,schema=indirect </
association> <add-attr attr-name="fname"> <value type="string">John</
value> </add-attr> <add-attr attr-name="lname"> <value
type="string">Doe</value> </add-attr> <add-attr attr-name="photo">
<value type="octet">u7s=</value> </add-attr> </add>
```

更新欄位 **(** 查詢回覆 **)**

下表顯示更新表格 usr 中的列之後，事件記錄表格的內容。因爲不會使用舊值和新值，所以會省略它們。

表格 ***5-23*** 事件記錄表格：類型 *8*

event_type	table	table_key	column_name	old_value	new_value
8	usr	idu=1	fname	NULL	NULL
8	usr	idu=1	lname	NULL	NULL
8	usr	idu=1	photo	NULL	NULL

「發行者」通道會產生下列 XML。值會反映表格 usr 的目前內容，而不是啓始內容。

```
<modify class-name="usr"> <association>idu=1,table=usr,schema=indirect
</association> <modify-attr attr-name="fname"> <remove-all-values/>
<add-value> <value type="string">John</value> </add-value> </modify-
attr> <modify-attr attr-name="lname"> <remove-all-values/> <add-value>
<value type="string">Doe</value> </add-value> </modify-attr> <modify-
attr attr-name="photo"> <remove-all-values/> <add-value> <value
type="octet">u7s=</value> </add-value> </modify-attr> </modify>
```

5.4 在 XDS 事件中內嵌 SQL 陳述式

本節含有可協助您在 XDS 事件中內嵌 SQL 的資訊。

所有範例都會參考下列 indirect.usr 表格。

```
CREATE TABLE indirect.usr ( idu   INTEGER  NOT NULL, fname
VARCHAR2(64), lname VARCHAR2(64),

    CONSTRAINT pk_usr_idu PRIMARY KEY(idu) );
```

內嵌式 SQL 可讓您在 XDS 格式的 XML 文件中內嵌 SQL 陳述式。您可以與 XDS 事件結合使用或獨立使用內嵌式 SQL 陳述式。內嵌式 SQL 陳述式獨立使用時，內嵌式 SQL 處理不需要驅動程式瞭解目標資料庫中的表格 / 檢視窗。因此，驅動程式可以在不瞭解綱要的模式中執行。請參閱「同步化過濾器」，第 44 頁。獨立使用內嵌式 SQL 時，您必須手動建立關聯。驅動程式不會爲您建立它們。

與 XDS 事件結合使用時，內嵌式 SQL 可以做爲虛擬的資料庫觸發。您可以在表格上安裝資料庫觸發，且會在執行某些 SQL 陳述式時在資料庫中產生副作用，與此相同，內嵌式 SQL 也會爲回應某些 XDS 事件而在資料庫中產生副作用。

5.4.1 內嵌式 SQL 的一般用途

您可以藉由在 XDS 事件中內嵌 SQL，以完成下列操作：

- 建立資料庫使用者或角色。
- 管理使用者密碼

  您可以設定、檢查或修改使用者密碼。
- 管理資料庫使用者或角色權限。

範例組態檔案 JDBCv2.xml 會示範如何做爲 XDS 事件的副作用，建立資料庫使用者、管理使用者密碼和管理使用者權限。若要啓用資料庫使用者帳戶管理，請將全域組態變數 (Global Configuration Variable, GCV) user-ddl 設爲 true。內嵌式 SQL 範例包含在「訂閱者」通道上的 User DDL「指令轉換」樣式表中。

5.4.2 內嵌式 SQL 基礎

- 「元素」，第 96 頁
- 「名稱空間」，第 96 頁
- 「內嵌式 SQL 範例」，第 97 頁

元素

SQL 會透過 <jdbc:statement> 和 <jdbc:sql> 元素內嵌在 XDS 事件中。<jdbc:statement> 元素可以包含一或多個 <jdbc:sql> 元素。

名稱空間

當在 XML 文件的外部受到參考時，本節中使用的名稱空間字首 jdbc 會隱含地與名稱空間 urn:dirxml:jdbc 結合。

您必須使用名稱空間字首的內嵌式 SQL 元素和屬性。否則，驅動程式將無法辨識它們。在本節的所有範例中，使用的字首是 jdbc。實際上，只要字首與名稱空間值 urn:dirxml:jdbc 結合，它可以是您想要的任何字元。

下列 XML 範例說明如何使用並適當地建立內嵌式 SQL 元素的名稱空間字首。在下列範例中，名稱空間宣告和名稱空間字首以粗體顯示：

```
<input xmlns:jdbc="urn:dirxml:jdbc"> <add class-name="usr"> <add-attr
name="lname"> <value>Doe</value> </add-attr> </add> <jdbc:statement>
<jdbc:sql>UPDATE indirect.usr SET fname = 'John'</jdbc:sql> </
jdbc:statement> </input>
```

內嵌式 **SQL** 範例

下列 XML 範例說明如何使用 <jdbc:statement> 和 <jdbc:sql> 元素以及它們的解譯。在下列範例中，內嵌式 SQL 元素以粗體顯示：

```
<input xmlns:jdbc="urn:dirxml:jdbc"> <add class-name="usr"> <add-attr
name="lname"> <value>Doe</value> </add-attr> </add> <jdbc:statement>
<jdbc:sql>UPDATE indirect.usr SET fname = 'John'</jdbc:sql> </
jdbc:statement> </input>
```

因爲「訂閱者」通道會將 <add> 事件解析爲一或多個 INSERT 陳述式，所以上面顯示的 XML 會解析爲：

```
SET AUTOCOMMIT OFF INSERT INTO indirect.usr(lname)VALUES('Doe');
COMMIT; --explicit commit UPDATE indirect.usr SET fname = 'John';
COMMIT; --explicit commit
```

5.4.3 記號替換

「訂閱者」通道支援內嵌式 SQL 陳述式的記號替換，而不需要您從關聯剖析欄位值。在下列範例中，記號及其參考的值都以粗體顯示：

```
<input xmlns:jdbc="urn:dirxml:jdbc"> <modify class-name="usr">
<association>idu=1,table=usr,schema=indirect</association> <modify-
attr name="lname"> <add-value> <value>DoeRaeMe</value> </add-value> </
modify-attr> </modify> <jdbc:statement> <jdbc:sql>UPDATE indirect.usr
SET fname = 'John' WHERE idu = {$idu}</jdbc:sql> </jdbc:statement> </
input>
```

記號預留位置必須遵循 XSLT 屬性值範本語法 {$*field-name*}。同時，參考的關聯元素必須在 XDS 文件中的 <jdbc:statement> 元素前，或者必須呈現爲 <jdbc:statement> 元素的子代。或者，如果不複製關聯元素做爲 <jdbc:statement> 元素的子代，也可以將包含關聯元素的元素 src-entry-id 複製到 <jdbc:statement> 元素上。在下列範例中，這兩種方法都以粗體顯示：

```
<input xmlns:jdbc="urn:dirxml:jdbc"> <modify class-name="usr">
<association>idu=1,table=usr,schema=indirect</association> <modify-
attr name="lname"> <add-value> <value>DoeRaeMe</value> </add-value> </
modify-attr> </modify> <jdbc:statement>
           <association>idu=1,table=usr,schema=indirect</association>
<jdbc:sql>UPDATE indirect.usr SET fname = 'John' WHERE idu = {$idu}</
```

```
jdbc:sql> </jdbc:statement> </input>
```

```
<input xmlns:jdbc="urn:dirxml:jdbc"> <modify class-name="usr" src-
entry-id="0"> <association>idu=1,table=usr,schema=indirect</
association> <modify-attr name="lname"> <add-value> <value>DoeRaeMe</
value> </add-value> </modify-attr> </modify> <jdbc:statement src-
entry-id="0"> <jdbc:sql>UPDATE indirect.usr SET fname = 'John' WHERE
idu = {$idu}</jdbc:sql> </jdbc:statement> </input>
```

{*$ 欄位名稱* } 記號必須參考關聯值中之其中一個命名相對可辨識名稱 (Relative Distinguished Name，RDN) 屬性名稱。以上範例只有一個命名屬性：idu。

由於關聯尚未建立，所以 <add> 事件是關聯元素不需要在內嵌式 SQL 陳述式前加上記號的唯一事件。此外，任何使用記號的內嵌式 SQL 陳述式，都必須加在 <add> 事件之後，而不是在它之前。例如：

```
<input xmlns:jdbc="urn:dirxml:jdbc"> <add class-name="usr"> <add-attr
name="lname"> <value>Doe</value> </add-attr> </add> <jdbc:statement>
<jdbc:sql>UPDATE indirect.usr SET fname = 'John' WHERE idu = {$idu}</
jdbc:sql> </jdbc:statement> </input>
```

若要阻止任何人追蹤機密資訊，您可以使用 {$$password} 記號，參考相同文件內直接加在 <password> 元素之前的內容。在下列範例中，password 記號及其參考的值，都以粗體字顯示：

```
<input xmlns:jdbc="urn:dirxml:jdbc"> <add class-name="usr">
<password>some password</password> <add-attr name="fname">
<value>John</value> </add-attr> <add-attr name="lname"> <value>Doe</
value> </add-attr> </add> <jdbc:statement> <jdbc:sql>CREATE USER jdoe
IDENTIFIED BY {$$password}</jdbc:sql> </jdbc:statement> </input>
```

此外，您也可以參考由「應用程式密碼」參數指定爲 {$$$driver-password} 的驅動程式資料庫驗證密碼。請參閱「應用程式密碼」，第 40 頁。目前尚不支援具名密碼替代功能。

如同關聯元素的情況，在 XDS 文件中參考的密碼元素必須在 <jdbc:statement> 元素之前，或者必須呈現爲 <jdbc:statement> 元素的子代。或者，不將密碼元素複製爲 <jdbc:statement> 元素的子代，而是將包含密碼元素之元素的 src-entry-id 複製到 <jdbc:statement> 元素中。在下列範例中，這兩種方法都以粗體字顯示：

```
<input xmlns:jdbc="urn:dirxml:jdbc"> <add class-name="usr">
<password>some password</password> <add-attr name="fname">
<value>John</value> </add-attr> <add-attr name="lname"> <value>Doe</
value> </add-attr> </add> <jdbc:statement> <password>some password</
password> <jdbc:sql>CREATE USER jdoe IDENTIFIED BY {$$password}</
jdbc:sql> </jdbc:statement> </input>
```

```
<input xmlns:jdbc="urn:dirxml:jdbc"> <add class-name="usr" src-entry-
id="0"> <password>some password</password> <add-attr name="fname">
<value>John</value> </add-attr> <add-attr name="lname"> <value>Doe</
```

```
value> </add-attr> </add> <jdbc:statement src-entry-id="0">
<jdbc:sql>CREATE USER jdoe IDENTIFIED BY {$$password}</jdbc:sql> </
jdbc:statement> </input>
```

5.4.4 虛擬觸發

資料庫觸發可以在觸發陳述式之前或之後引發；相同的，內嵌式 SQL 也可以定位在觸發 XDS 事件之前或之後。下列範例顯示如何將 SQL 內嵌在 XDS 事件之前或之後。

觸發前虛擬

```
<input xmlns:jdbc"urn:dirxml:jdbc"> <jdbc:statement>
          <association>idu=1,table=usr,schema=indirect</association>
          <jdbc:sql>UPDATE indirect.usr SET fname = 'John' WHERE
                    idu = {$idu}</jdbc:SQL>     </jdbc:statement>
    <modify class-name="usr">
        <association>idu=1,table=usr,schema=indirect</association>
        <modify-attr name="lname">             <remove-all-values/>
            <add-value>                 <value>Doe</value>
            </add-value>         </modify-attr>     </modify> </input>
```

此 XML 會解析爲：

```
SET AUTOCOMMIT OFF UPDATE indirect.usr SET fname = 'John' WHERE idu =
1; COMMIT; --explicit commit UPDATE indirect.usr SET lname = 'Doe'
WHERE idu = 1; COMMIT; --explicit commit
```

觸發後虛擬

```
<input xmlns:jdbc"urn:dirxml:jdbc">     <modify class-name="usr">
        <association>idu=1,table=usr,schema=indirect</association>
        <modify-attr name="lname">             <remove-all-values/>
            <add-value>                 <value>Doe</value>
            </add-value>     </modify-attr>     </modify>
    <jdbc:statement>         <jdbc:sql>UPDATE indirect.usr SET fname =
'John' WHERE                                    idu = {$idu}</
jdbc:sql>     </jdbc:statement> </input>
```

此 XML 會解析爲：

```
SET AUTOCOMMIT OFF UPDATE indirect.usr SET lname = 'Doe'  WHERE idu =
1; COMMIT; --explicit commit UPDATE indirect.usr SET fname = 'John'
WHERE idu = 1; COMMIT; --explicit commit
```

5.4.5 手動與自動的異動

您可以使用兩個自定屬性，以手動方式將內嵌式 SQL 和 XDS 事件組合在一起：

- jdbc:transaction-type
- jdbc:transaction-id

jdbc:transaction-type

此屬性具有兩個值：manual 和 auto。在預設狀態下，大部份相關的 XDS 事件 (<add>、<modify> 和 <delete>) 都以隱含方式設爲手動異動類型。手動設定可讓 XDS 事件解析爲由一或多個 SQL 陳述式組成的異動。

由於部份 SQL 陳述式 (例如，資料定義語言 (DDL) 陳述式) 通常無法包含在手動異動中，因此，在預設狀態下，內嵌式 SQL 事件會設爲自動異動類型。在下列範例中，屬性以粗體字顯示。

```
<input xmlns:jdbc="urn:dirxml:jdbc"> <add class-name="usr"
jdbc:transaction-type="auto"> <add-attr name="lname"> <value>Doe</
value> </add-attr> </add> <jdbc:statement> <jdbc:sql>UPDATE
indirect.usr SET fname = 'John' WHERE idu = {$idu}</jdbc:sql> </
jdbc:statement> </input>
```

此 XML 會解析爲：

```
SET AUTOCOMMIT ON INSERT INTO indirect.usr(lname) VALUES('Doe'); --
implicit commit UPDATE indirect.usr SET fname = 'John' WHERE idu = 1; -
- implicit commit
```

jdbc:transaction-id

除非元素的 jdbc:transaction-type 屬性值預設爲或明確地設爲 manual，否則「訂閱者」通道會忽略此屬性。下列 XML 顯示手動異動的範例。屬性以粗體字顯示。

```
<input xmlns:jdbc="urn:dirxml:jdbc"> <add class-name="usr"
jdbc:transaction-id="0"> <add-attr name="lname"> <value>Doe</value> </
add-attr> </add> <jdbc:statement jdbc:transaction-type="manual"
                    jdbc:transaction-id="0"> <jdbc:sql>UPDATE
indirect.usr SET fname = 'John' WHERE idu = {$idu}</jdbc:sql> </
jdbc:statement> </input>
```

此 XML 會解析爲：

```
[XXX]SET AUTOCOMMIT OFF INSERT INTO indirect.usr(lname) VALUES('Doe');
UPDATE indirect.usr SET fname = 'John' WHERE idu = 1; COMMIT; --
explicit commit
```

5.4.6 異動隔離層級

除了群組陳述式之外，您可以使用異動來保留資料庫中的資料完整性。異動可以鎖定資料，以防止同時進行存取或修改。異動的隔離層級決定鎖定的設定方式。通常，驅動程式使用的預設隔離層級是足夠的，而且不應該更動。

自定屬性 jdbc:isolation-level 可讓您在必要時調整隔離異動層級。java.sql.Connection 參數會在介面中定義五個可能的值。請參閱 java.sql.Connection (http://java.sun.com/j2se/1.5.0/docs/api/java/sql/Connection.html)。

- none
- read uncommitted
- read committed
- repeatable read
- serializable

除非由描述元檔案置換，否則驅動程式的預設異動隔離層級是 read committed。在手動異動中，將 jdbc:isolation-level 屬性置於異動的第一個元素。後續元素會忽略此屬性。在下列範例中，屬性以粗體字顯示。

```
<input xmlns:jdbc="urn:dirxml:jdbc"> <add class-name="usr"
jdbc:transaction-id="0"                        jdbc:isolation-
level="serializable"> <add-attr name="lname"> <value>Doe</value> </
add-attr> </add> <jdbc:statement jdbc:transaction-type="manual"
jdbc:transaction-id="0"> <jdbc:sql>UPDATE indirect.usr SET fname =
'John' WHERE idu = {$idu}</jdbc:sql> </jdbc:statement> </input>
```

此 XML 會解析爲：

```
SET AUTOCOMMIT OFF SET TRANSACTION ISOLATION LEVEL SERIALIZABLE INSERT
INTO indirect.usr(lname) VALUES('Doe'); UPDATE indirect.usr SET fname
= 'John' WHERE idu = 1; COMMIT; -- explicit commit
```

5.4.7 陳述式類型

「訂閱者」通道雖執行內嵌式 SQL 陳述式，卻不瞭解它們。JDBC 1 介面會定義數種執行不同類型之 SQL 陳述式的方法。下表包含這些方法：

表格 ***5-24*** 執行 *SQL* 陳述式的方法

陳述式類型	執行的方法
SELECT	java.sql.Statement.executeQuery(String query):java.sql.ResultSet
INSERT	java.sql.Statement.executeUpdate(String update):int
UPDATE	java.sql.Statement.executeUpdate(String update):int
DELETE	java.sql.Statement.executeUpdate(String update):int

陳述式類型	執行的方法
CALL 或 EXECUTE SELECT INSERT UPDATE DELETE	java.sql.Statement.execute(String sql):boolean

最簡單的解決方案是將所有的 SQL 陳述式映射到 java.sql.Statement.execute(String sql):boolean 方法。在預設狀態下，「訂閱者」通道會使用此方法。

有些協力廠商驅動程式 (特別是 Oracle 的 JDBC 驅動程式) 會錯誤地實作用於決定此方法所產生之結果集數的方法。於是，驅動程式因而陷入無限迴路中，導致 CPU 使用率過高。若要避免這個問題，您可以使用任意 <jdbc:statement> 元素上的 jdbc:type 屬性，將其包含的 SQL 陳述式映射到下列方法，而非預設的方法：

- java.sql.Statement.executeQuery(String query):java.sql.ResultSet
- java.sql.Statement.executeUpdate(String update):int

jdbc:type 屬性具有兩個值：update 和 query。針對 INSERT、UPDATE 或 DELETE 陳述式，將值設爲 update。針對 SELECT 陳述式，將值設爲 query。在沒有此屬性的情況下，驅動程式會將所有的 SQL 陳述式映射到預設的方法。如果置於 <jdbc:statement> 以外的任何元素，則會忽略此屬性。

建議：

- 將 jdbc:type="query" 屬性值置於所有 SELECT 陳述式。
- 將 jdbc:type="update" 屬性值置於所有 INSERT、UPDATE 和 DELETE 陳述式。
- 不在預存程序 / 函數呼叫上放置任何屬性值。

下列 XML 顯示 jdbc:type 屬性的範例。屬性以粗體字顯示。

```
<input xmlns:jdbc="urn:dirxml:jdbc"> <add class-name="usr"> <add-attr
name="lname"> <value>Doe</value> </add-attr> </add> <jdbc:statement
jdbc:type="update"> <jdbc:sql>UPDATE indirect.usr SET fname = 'John'
WHERE idu = {$idu}</jdbc:sql> </jdbc:statement> </input>
```

5.4.8 SQL 查詢

爲了完全支援資料庫的查詢功能，並避免將原始 SQL 查詢轉譯爲 XDS 格式所帶來的困難，驅動程式會支援原始 SQL 查詢處理。您可以使用與其他任何 SQL 陳述式完全相同的方法，將 select 陳述式內嵌於 XDS 文件。

例如，假設表格 usr 具有下列內容：

表格 ***5-25*** 範例內容

idu	fname	lname
1	John	Doe

以下 XML 文件會產生含有單一結果集的輸出文件。

```
<input xmlns:jdbc="urn:dirxml:jdbc"> <jdbc:statement
jdbc:type="query"> <jdbc:sql>SELECT * FROM indirect.usr</jdbc:sql> </
jdbc:statement> </input>


<output xmlns:jdbc="urn:dirxml:jdbc"> <jdbc:result-set jdbc:number-of-
rows="1"> <jdbc:row jdbc:number="1"> <jdbc:column jdbc:name="idu"
jdbc:position="1"  jdbc:type="java.sql.Types.BIGINT <jdbc:value>1</
jdbc:value> </jdbc:column>             <jdbc:column jdbc:name="fname"
jdbc:position="2" jdbc:type="java.sql.Types.VARCHAR>
<jdbc:value>John</jdbc:value> </jdbc:column> <jdbc:column
jdbc:name="lname" jdbc:position="3"
                         jdbc:type="java.sql.Types.VARCHAR>
<jdbc:value>Doe</jdbc:value> </jdbc:column> </jdbc:row> </jdbc:result-
set> <status level="success"/> </output>
```

無論結果集是否包含列，SQL 查詢都會產生單一 <jdbc:result-set> 元素。如果結果集爲空，則 jdbc:number-of-rows 屬性會設爲零。

您可以在文件中內嵌一個以上的查詢。SQL 查詢不需要讓驅動程式看到同步化綱要中參考的表格 / 檢視窗。不過，XDS 查詢需要。

5.4.9 資料定義語言 (DDL) 陳述式

一般來說，由於大部份資料庫都不允許混合的資料操作語言 (DML) 和資料定義語言 (DDL) 異動，所以不可能在資料庫觸發中執行 DDL 陳述式。雖然虛擬觸發沒有克服此執行屬性限制，卻允許讓資料定義語言 (DDL) 陳述式當成 XDS 事件的副作用來執行。

例如：

```
<input xmlns:jdbc="urn:dirxml:jdbc"> <add class-name="usr"> <add-attr
name="fname"> <value>John</value> </add-attr> <add-attr name="lname">
<value>Doe</value> </add-attr> </add> <jdbc:statement>
<jdbc:sql>CREATE USER jdoe IDENTIFIED BY novell</jdbc:sql> </
jdbc:statement> </input>
```

此 XML 會解析爲：

```
SET AUTOCOMMIT OFF INSERT INTO indirect.usr(fname, lname)
VALUES('John', 'Doe'); COMMIT; -- explicit commit SET AUTOCOMMIT ON
CREATE USER jdoe IDENTIFIED BY novell; -- implicit commit
```

使用 jdbc:transaction-id 和 jdbc:transaction-type 屬性將資料操作語言 (DML) 和資料定義語言 (DDL) 陳述式組成爲單一異動，會使異動在大多數資料庫上復原。因爲資料定義語言 (DDL) 陳述式通常都當成不同異動來執行，所以上面範例中的 insert 陳述式可能會成功，而且 create user 陳述式可能會復原。

不過，不可能是 insert 陳述式失敗，而 create user 陳述式成功。驅動程式會在第一個異動復原的點上，停止執行鏈結異動。

5.4.10 邏輯操作

因爲在單一異動中通常不可能混合資料操作語言 (DML) 和資料定義語言 (DDL) 陳述式，所以單一事件可包含一或多個異動。您可以使用 jdbc:op-id 和 jdbc:op-type，將多個異動組成爲單一邏輯操作。此時，操作的所有成員都會處理爲和狀態相關的單一單位。如果一個成員有錯誤，所有的成員都會傳回相同的狀態層級。同樣地，所有的成員都共享相同的狀態類型。

```
<input xmlns:jdbc="urn:dirxml:jdbc"> <add class-name="usr" jdbc:op-
id="0" jdbc:op-type="password-set-operation"> <add-attr name="fname">
<value>John</value> </add-attr> <add-attr name="lname"> <value>Doe</
value> </add-attr> <password>Doe{$idu}</password> </add>
<jdbc:statement jdbc:op-id="0"> <jdbc:sql>CREATE USER jdoe IDENTIFIED
BY {$$password} </jdbc:sql> </jdbc:statement> </input>
```

除了邏輯操作中的第一個元素外，所有元素都會忽略 jdbc:op-type 屬性。

5.4.11 使用內嵌式 SQL 實作密碼設定

密碼的啓始設定，通常都會藉由建立資料庫使用者帳戶來完成。假設在「訂閱者」通道上產生 <add> 事件的情況下，下列舉出了由 XSLT 樣式表所產生的輸出範例，而這些樣式表將密碼設定當成 XDS <add> 事件的副作用來實作：

```
<input xmlns:jdbc="urn:dirxml:jdbc"> <add class-name="usr" jdbc:op-
id="0" jdbc:op-type="password-set-operation"> <add-attr name="fname">
<value>John</value> </add-attr> <add-attr name="lname"> <value>Doe</
value> </add-attr> <password>Doe{$idu}</password> </add>
<jdbc:statement jdbc:op-id="0"> <jdbc:sql>CREATE USER jdoe IDENTIFIED
BY {$$password} </jdbc:sql> </jdbc:statement> </input>
```

在邏輯上，<add> 事件會由 jdbc:op-id 和 jdbc:op-type 屬性結合至 CREATE USER 資料定義語言 (DDL) 陳述式。

JDBCv2.xml 範例組態檔案中的 User DDL「指令轉換」樣式表包含範例 XSLT 範本，其會將使用者帳戶建立資料定義語言 (DDL) 陳述式結合至支援它們的所有資料庫的 <add> 事件。

5.4.12 使用內嵌式 SQL 實作修改密碼

密碼的啓始設定，通常都會藉由變更現有的資料庫使用者帳戶來完成。假設在「訂閱者」通道上產生 <modify-password> 事件，則下列是由實作 modify-password 之 XSLT 樣式表所產生的輸出範例：

```
<input xmlns:jdbc="urn:dirxml:jdbc"> <modify-password jdbc:op-id="0"
jdbc:op-type="password-set-operation"> <password>new password</
password> </modify-password> <jdbc:statement jdbc:op-id="0">
<jdbc:sql>ALTER USER jdoe IDENTIFIED BY {$$password} </jdbc:sql> </
jdbc:statement> </input>
```

在邏輯上，<modify-password> 事件會由 jdbc:op-id 和 jdbc:op-type 屬性結合至 ALTER USER 資料定義語言 (DDL) 陳述式。

JDBCv2.xml 範例組態中的 User DDL「指令轉換」樣式表包含範例 XSLT 範本，其會將密碼維護資料定義語言 (DDL) 陳述式結合至支援它們的所有資料庫的 <modify-password> 事件。

5.4.13 實作檢查物件密碼

與密碼設定不同，檢查物件密碼不需要內嵌式 SQL 陳述式或屬性。只需要使用者帳戶名稱。這可以從關聯值 (假設以手動方式維護關聯)、目錄屬性或資料庫欄位取得。如果是儲存在目錄或資料庫中，必須發出查詢才能取得值。

JDBCv2.xml 範例組態檔會將資料庫使用者帳戶名稱儲存在資料庫欄位中。

附註：有些資料庫 (例如，Sybase Adpative Server Enterprise 和 Microsoft SQL Server) 會區分使用者帳戶名稱與登入帳戶名稱。因此，您可能需要儲存兩個名稱，而不是只儲存一個。

若要實作檢查物件密碼，可將 dest-dn 屬性值附加至 <check-object-password> 事件。在以下範例中，dest-dn 屬性以粗體字顯示：

```
<input xmlns:jdbc="urn:dirxml:jdbc"> <check-object-password dest-
dn="jdoe"> <password>whatever</password> </check-object-password> </
input>
```

5.4.14 最佳作法

基於效能因素，呼叫包含多個 SQL 陳述式的單一預存程序 / 函數，比在 XDS 文件中內嵌多個陳述式好。

在下列範例中，優先使用單一預存程序或函數。

單一預存程序

```
<input xmlns:jdbc="urn:dirxml:jdbc"> <add class-name="usr"> <add-attr
name="fname"> <value>John</value> </add-attr> <add-attr name="lname">
<value>Doe</value> </add-attr> </add> <jdbc:statement> <jdbc:sql>CALL
PROCEDURE set_name('John', 'Doe')</jdbc:sql> </jdbc:statement> </
input>
```

多個內嵌式陳述式

```
<input xmlns:jdbc="urn:dirxml:jdbc"> <add class-name="usr"> <add-attr
name="lname"> <value>Doe</value> </add-attr> </add> <jdbc:statement>
        <jdbc:sql>UPDATE indirect.usr SET fname = 'John'
                  WHERE idu = {$idu}</jdbc:sql>     </jdbc:statement>
    <jdbc:statement>         <jdbc:sql>UPDATE indirect.usr SET lname =
'Doe'                    WHERE idu = {$idu}</jdbc:sql>     </
jdbc:statement> </input>
```

用於呼叫預存程序或函數的語法，因資料庫而不同。如需其他資訊，請參閱「預存程序和函數 JDBC 呼叫語法」，第 125 頁。

協力廠商 JDBC 驅動程式

6

6.1 協力廠商 JDBC 驅動程式互通性

設計 Identity Manager Driver for JDBC 的目的，是要與特定的協力廠商 JDBC 驅動程式集 (而非特定的資料庫集) 進行互通。事實上，協力廠商 JDBC 驅動程式 (非資料庫) 是 Driver for JDBC 是否針對給定的資料庫運作的主要決定要素。一般而言，如果 Driver for JDBC 與給定的協力廠商 JDBC 驅動程式互通良好，則它與協力廠商驅動程式支援的資料庫和資料庫版本都能互通良好。

強烈建議您儘可能使用由主要企業資料庫廠商提供的協力廠商 JDBC 驅動程式 (例如本節中所列出者)。這些通常是免費且成熟的驅動程式，而且確知與 Driver for JDBC 及其目標資料庫互通良好。您可以使用其他協力廠商驅動程式，但是 Novell® 並不支援它們。

一般而言，大部份協力廠商驅動程式都是反向相容。然而，即使如此，也不會是正向相容。無論何時升級資料庫伺服器，也應該同時更新與此產品搭配使用的協力廠商驅動程式。

除非另有指示，否則通常都會建議您使用最新版本的協力廠商驅動程式。

6.2 JDBC 驅動程式類型

類型 **1**

部份爲 Java，並透過原始 ODBC 驅動程式與資料庫伺服器間接通訊的協力廠商 JDBC 驅動程式。

類型 1 驅動程式當做 JDBC-ODBC 橋接器使用。Sun 會提供 JDBC-ODBC 橋接器驅動程式，以供實驗性使用，並在無其他協力廠商 JDBC 驅動程式類型可用的情況下使用。

類型 **2**

部份爲 Java，並透過其原始用戶端應用程式介面 (API) 與資料庫伺服器間接通訊的協力廠商 JDBC 驅動程式。

類型 **3**

純 Java，並透過中介軟體伺服器與資料庫伺服器間接通訊的協力廠商 JDBC 驅動程式。

類型 **4**

純 Java，並與資料庫伺服器直接通訊的協力廠商 JDBC 驅動程式。

6.2.1 使用何種類型？

類型 3 和 4 驅動程式通常比類型 1 和 2 驅動程式穩定。類型 1 和 2 驅動程式通常比類型 3 和 4 驅動程式快。類型 2 和 3 驅動程式通常比類型 1 和 4 驅動程式安全。

由於 Identity Manager 使用目錄做爲其資料儲存，而資料庫通常要比目錄快得多，所以效能並不是主要的問題。然而，穩定性卻是一個問題。因此，建議您儘可能使用類型 3 或 4 協力廠商 JDBC 驅動程式。

重要：如果您選擇類型 1 或類型 2 驅動程式 (包含機器碼者) 與 Driver for JDBC 搭配使用，請使用 「遠端載入器」以確保目錄程序的完整性。

6.3 協力廠商 Jar 檔案佈置

下表會識別在 Identity Manager 或 「遠端載入器」伺服器上應放置協力廠商 JDBC 驅動程式 jar 檔案的路徑 (假設使用的是預設安裝路徑)。

6.3.1 Identity Manager 檔案路徑

下表指出依平台在 「身份管理」伺服器上放置協力廠商 JDBC 驅動程式 jar 檔案的位置。

表格 ***6-1*** *jar* 檔案的位置：*Identity Manager* 伺服器

平台	目錄路徑
NetWare®	sys:\system\lib
Solaris、Linux 或 AIX	/usr/lib/dirxml/classes (eDirectory 8.8 之前的版本) /opt/novell/eDirectory/lib/dirxml/classes (eDirectory 8.8)
Windows NT/2000	novell\NDS\lib

6.3.2 遠端載入器檔案路徑

下表指出依平台在 「遠端載入器」伺服器上放置協力廠商 JDBC 驅動程式 jar 檔案的位置。

表格 ***6-2*** *jar* 檔案的位置：遠端載入器

平台	目錄路徑
Solaris、Linux 或 AIX	/usr/lib/dirxml/classes (eDirectory 8.8 之前的版本) /opt/novell/eDirectory/lib/dirxml/classes (eDirectory 8.8)
Windows NT/2000	novell\RemoteLoader\lib

6.4 受支援的協力廠商 JDBC 驅動程式

6.4.1 協力廠商 JDBC 驅動程式功能

下表概述協力廠商 JDBC 驅動程式功能：

表格 ***6-3*** 協力廠商 *JDBC* 驅動程式功能

驅動程式	支援加密傳輸？	支援自動產生之金鑰的取回？
BEA* Weblogic* jDriver	否	否
IBM DB2 UDB 類型 3	否	否
IBM DB2 UDB 類型 4	否	否
Informix	否	否
Microsoft 2000	否	否
MySQL Connector/J	是	是
Oracle Thin Client	是	否
PostgreSQL	是 *	否
Sybase jConnect	是	否

* 適用於 JDBC 3 (Java 1.4) 版本和更新版本。

6.4.2 JDBC URL 語法

下表列出受支援的協力廠商 JDBC 驅動程式的 URL 語法：

表格 ***6-4*** *URL* 語法

協力廠商 JDBC 驅動程式	JDBC URL 語法
Oracle Thin Client	jdbc:oracle:thin:@*IP 位址* :1521:*sid*
IBM DB2 UDB 類型 3	jdbc:db2:///*IP 位址* :6789/*database-name*
IBM DB2 UDB 類型 4，通用	jdbc:db2:///*IP 位址 :50000/ 資料庫名稱*
BEA Weblogic jDriver	jdbc:weblogic:mssqlserver4: *資料庫名稱* @*IP 位址* :1433
Microsoft SQL Server	jdbc:microsoft:sqlserver:///*IP 位址或 dns 名稱* :1433;DatabaseName= *資料庫名稱*
Sybase jConnect	jdbc:sybase:Tds:*ip-address*:2048/ *資料庫名稱*
MySQL Connector/J	jdbc:mysql:///*IP 位址* :3306/ *資料庫名稱*
Informix	jdbc:informix-sqli:///*IP 位址* :1526/ *資料庫名稱* :informixserver= *伺服器 ID*
PostgreSQL	jdbc:postgresql:///*IP 位址* :5432/ *資料庫名稱*

此資訊可以與「驗證網路位置」參數共同使用。如需此參數的相關資訊，請參閱「驗證網路位置」，第 40 頁。

6.4.3 JDBC 驅動程式類別名稱

下表列出受支援的協力廠商 JDBC 驅動程式的完整 Java 類別名稱。

表格 ***6-5*** 協力廠商 *JDBC* 驅動程式的類別名稱

協力廠商 JDBC 驅動程式	類別名稱
BEA Weblogic jDriver	weblogic.jdbc.mssqlserver4.Driver
IBM DB2 UDB 類型 3	COM.ibm.db2.jdbc.net.DB2Driver
IBM DB2 UDB 類型 4，通用	com.ibm.db2.jcc.DB2Driver
Informix	com.informix.jdbc.IfxDriver
Microsoft 2000	com.microsoft.jdbc.sqlserver.SQLServerDriver
MySQL Connector/J	org.gjt.mm.mysql.Driver
Oracle Thin Client	oracle.jdbc.driver.OracleDriver
PostgreSQL	org.postgresql.Driver
Sybase jConnect 5.5	com.sybase.jdbc2.jdbc.SybDriver

此資訊可以與「JDBC 驅動程式類別名稱」參數共同使用。如需此參數的相關資訊，請參閱「協力廠商 JDBC 驅動程式類別名稱」，第 42 頁。

6.4.4 BEA Weblogic jDriver for Microsoft SQL Server

表格 **6-6** *BEA Weblogic jDriver*

受支援的資料庫版本：	Microsoft SQL Server 6.5、7.x、8.x (2000)
類別名稱	weblogic.jdbc.mssqlserver4.Driver
類型	4
URL 語法	jdbc:weblogic:mssqlserver4: *資料庫名稱*@*IP 位址*:1433
下載指示	免費註冊，並下載最新版的 Weblogic 伺服器。執行安裝程式。weblogic.jar 檔案安裝在 *安裝目錄*/server/lib 目錄中。 BEA 下載中心 (http://commerce.bea.com/showallversions.jsp?family=WLS)
檔名	weblogic.jar
文件 URL	jDriver 文件 (http://e-docs.bea.com/wls/docs81/mssqlserver4/)

附註：受支援的協力廠商驅動程式清單中包含 BEA Weblogic 驅動程式，以提供對 Microsoft SQL server 7 的 JDBC 存取。Microsoft 的驅動程式僅支援第 8 版 (2000)。

相容性

BEA Weblogic 驅動程式是反向相容的。資料庫伺服器和驅動程式不經常進行更新。

安全性

BEA Weblogic 驅動程式不支援加密傳輸。

已知問題

- BEA Weblogic 驅動程式不是免費提供的，必須要購買並取得適當授權才行。
- 各驅動程式版本之間包含 UNIQUEIDENTIFIER 欄的關聯值是不一致的。

  較早版本的 BEA Weblogic 驅動程式針對原始 UNIQUEIDENTIFIER 欄傳回非標準的 java.sql.Types (http://java.sun.com/j2se/1.5.0/docs/api/java/sql/Types.html) 值。爲了彌補，Driver for JDBC 將該非標準類型映射到標準類型 java.sql.Types.BINARY (http://java.sun.com/j2se/1.5.0/docs/api/java/sql/Types.html)，因爲它可以鏡像複製 16 位元值的原始資料庫類型到最佳程度。此映射會產生 Base64 編碼的關聯值。

  更新版本的 BEA Weblogic 驅動程式會傳回標準類型 java.sql.CHAR (http://java.sun.com/j2se/1.5.0/docs/api/java/sql/Types.html)。此映射會產生非 Base64 編碼的關聯值，藉由使用較早版本 BEA Weblogic 驅動程式，使所產生的所有關聯失效。此變更事實上會中斷反向相容性。

  此問題的最佳解決方案是繼續使用較早版本的 BEA Weblogic 驅動程式。如果您必須升級，則必須移除所有失效的關聯，並重新關聯所有之前關聯的物件。
- 當在 AIX 上呼叫方法 java.sql.Connection.getConnection(String url, String username, String password) 時，BEA Weblogic 驅動程式會送出 java.lang.IllegalMonitorStateException (http://java.sun.com/j2se/1.5.0/docs/api/java/lang/IllegalMonitorStateException.html)。

6.4.5 IBM DB2 Universal Database JDBC 驅動程式

IBM DB2 驅動程式可以是類型 3 或類型 4。

類型 **3**

表格 ***6-7*** *IBM DB2* 驅動程式：類型 *3*

受支援的資料庫版本：	7.x
類別名稱：	COM.ibm.db2.jdbc.net.DB2Driver
類型	3
URL 語法：	jdbc:db2:///*IP 位址* :6789/ *資料庫名稱*
下載指示：	從資料庫伺服器複製檔案。
	file:/// *資料庫安裝目錄* /java
檔案名稱：	db2java.zip
文件 URL：	DB2 資訊中心 (http://publib.boulder.ibm.com/infocenter/db2v7luw)
	JDBC 程式設計 (http://publib.boulder.ibm.com/infocenter/db2v7luw/index.jsp?topic=/com.ibm.db2v7.doc/db2a0/db2a0159.htm)

重要：類型 3 驅動程式會取代爲第 8 版。

相容性

IBM DB2 驅動程式的最大特性是對版本的區分極爲嚴格。DB2 的主要或次要版本 (包含 FixPack) 之間是不相容的。因此，建議您使用安裝在資料庫伺服器上的檔案。

重要：即使僅在 FixPack 層級，每次更新目標資料庫時，都必須在 Identity Manager 或「遠端載入器」伺服器上更新 IBM DB2 驅動程式。

安全性

IBM DB2 驅動程式不支援加密傳輸。

已知問題

- 版本不符合通常會導致與連接性相關的失敗。

  IBM DB2 驅動程式的最常見問題是由於驅動程式 / 資料庫版本不符合所導致。版本不符合的前兆是與連接性相關的失敗，例如「CLI0601E 陳述式識別指標無效或陳述式已關閉」。要補救該問題，請以安裝在資料庫伺服器上的版本覆寫 Identity Manager 或「遠端載入器」伺服器上的 db2java.zip 檔案。

- 要診斷和補救資料庫伺服器上 Java 相關的錯誤，其難度是非常高的。

  當您嘗試安裝和執行以 Java 編寫之使用者定義的預存程序和函數時，可能會產生許多錯誤條件和錯誤碼。診斷這些錯誤很花時間，而且困難重重。記錄檔案 (資料庫伺服器

上的 db2diag.log) 通常可以提供額外的除錯資訊。此外，所有錯誤碼也會被記錄下來，而且可以在線上使用。

類型 **4**：通用驅動程式

表格 ***6-8*** *IBM DB2* 驅動程式：類型 *4*

受支援的資料庫版本	8.x
類別名稱	com.ibm.db2.jcc.DB2Driver
類型	4
URL 語法	jdbc:db2:///*IP 位址* :50000/ *資料庫名稱*
下載指示	下載爲最新 FixPack 的一部份 (建議)。 IBM 支援與下載 (http://www.ibm.com/support/us/) 或 從資料庫伺服器複製檔案。 file:/// *資料庫安裝目錄* /java
檔名	db2jcc.jar、db2jcc_license_cu.jar、db2jcc_javax.jar (選擇性)
文件 URL	DB2 資訊中心 (http://publib.boulder.ibm.com/infocenter/db2help) DB2 通用 JDBC 驅動程式 (http://publib.boulder.ibm.com/infocenter/db2help/index.jsp?topic=/com.ibm.db2.udb.doc/ad/t0010264.htm) DB2 通用 JDBC 驅動程式下的安全性 (http://publib.boulder.ibm.com/infocenter/db2help/index.jsp?topic=/com.ibm.db2.udb.doc/ad/cjvjcsec.htm)

附註：與類型 3 驅動程式不同的是，類型 4 驅動程式只有最小的已定義錯誤碼集。這種缺少會限制 Driver for JDBC 辨識連接性、重試、驗證和嚴重錯誤等不同狀況的能力。

相容性

IBM DB2 驅動程式是反向相容的。然而，它不使用資料庫第 7 版。資料庫伺服器會經常更新。驅動程式不經常更新。

安全性

IBM DB2 驅動程式支援各種驗證安全性機制，但是不支援加密傳輸。

已知問題

- 要診斷和補救資料庫伺服器上 Java 相關的錯誤，其難度是非常高的。

  當您嘗試安裝和執行以 Java 編寫之使用者定義的預存程序和函數時，會產生許多錯誤條件和錯誤碼。診斷這些錯誤很花時間，而且困難重重。記錄檔案 (資料庫伺服器上的 db2diag.log) 通常可以提供額外的除錯資訊。此外，所有錯誤碼也會被記錄下來，而且可以在線上使用。

6.4.6 Informix JDBC 驅動程式

表格 **6-9** *Informix JDBC* 驅動程式

受支援的資料庫版本	Dynamic Server 7.x、9.x
類別名稱	com.informix.jdbc.IfxDriver
類型	4
URL 語法	jdbc:informix-sqli://*IP 位址* :1526/ *資料庫名稱* :informixserver= *伺服器 ID*
下載指示	下載 URL (http://www-306.ibm.com/software/data/informix/tools/jdbc)
檔名	ifxjdbc.jar、ifxjdbcx.jar (選擇性)
文件 URL	Informix 資訊中心 (http://publib.boulder.ibm.com/infocenter/db2v7luw) Informix JDBC 驅動程式 (http://www-306.ibm.com/software/data/informix/pubs/library/jdbc_2.html)

相容性

Informix 驅動程式是反向相容的。資料庫伺服器和驅動程式不經常更新。

安全性

Informix 驅動程式不支援加密傳輸。

ANSI 相容資料庫的必要參數設定

下表列出必須對 Driver for JDBC 明確設定的驅動程式參數，以針對 ANSI 相容資料庫與 Informix 驅動程式互通。

表格 **6-10** *ANSI* 相容資料庫的驅動程式設定

顯示名稱	標籤名稱	值
支援中繼資料取回中的綱要？	supports-schemas-in-metadata-retrieval 請參閱「支援中繼資料取回中的綱要？」，第 58 頁。	false
強制使用者名稱大小寫：	force-username-case 請參閱「強制使用者名稱大小寫」，第 56 頁。	upper

動態參數預設值

下表列出 Driver for JDBC 在執行時期以隱含方式設定的驅動程式相容性參數。請勿置換這些設定。

表格 **6-11** 不可置換的 *Informix JDBC* 設定

顯示名稱	標籤名稱	值
函數傳回方法：	function-return-method 請參閱 「函數傳回方法」，第 57 頁。	結果集

已知問題

- 綱要名稱無法用於針對 ANSI 相容資料庫取回中繼資料。將驅動程式相容性參數 「支援中繼資料取回中的綱要？」，第 58 頁設爲布林值 False。可用於中繼資料取回的資料庫物件，是通過資料庫驗證之資料庫使用者可見的那些物件。綱要修飾詞無法用於識別資料庫物件。因此，若要避免命名衝突 (例如，owner1.table1、owner2.table1)，請只授予資料庫驗證使用者同步化物件的 SELECT 權限。
- 當針對 ANSI 相容資料庫使用時，使用者名稱必須大寫。將驅動程式相容性參數 「支援中繼資料取回中的綱要？」，第 58 頁設爲 upper。

6.4.7 Microsoft SQL Server 2000 Driver for JDBC

表格 **6-12** *Microsoft SQL Server 2000 Driver* 設定

受支援的資料庫版本：	8 (2000)
類別名稱	com.microsoft.jdbc.sqlserver.SQLServerDriver
類型	4
URL 語法	jdbc:microsoft:sqlserver://*IP 位址或 dns 名稱*:1433;DatabaseName=*資料庫名稱*
下載指示	Microsoft JDBC 下載 (http://www.microsoft.com/downloads/results.aspx?sortCriteria=date&OSID=&productID=&CategoryID=&freetext=jdbc&DisplayLang=en&DisplayEnglishAlso=)
檔名	msbase.jar、mssqlserver.jar、msutil.jar

相容性

SQL Server 2000 驅動程式是反向相容的。然而，它不使用資料庫第 7 版。資料庫伺服器和驅動程式不經常更新。

安全性

SQL Server 2000 驅動程式不支援加密傳輸。

URL 內容

使用分號 (;) 分隔 URL 內容。

下表列出 SQL Server 2000 驅動程式之 SelectMethod URL 內容的值。

表格 ***6-13*** *SelectMethod URL* 內容的值

合法的值	描述
direct	預設值。不允許單一連接中有多個使用中陳述式
cursor	允許單一連接中有多個使用中陳述式

動態參數預設值

下表列出 Driver for JDBC 在執行時期以隱含方式設定的驅動程式相容性參數。請勿明確地置換這些設定。

表格 ***6-14*** 不可置換的 *SQL Server 2000* 設定

顯示名稱	標籤名稱	值
重複使用陳述式？	reuse-statements	false

已知問題

- 由於連接是複製的，所以無法啓動手動異動。

  不允許相同連接上使用並行陳述式的實作異常，會導致 SQL Server 2000 驅動程式發生最常見的問題。與其他協力廠商實作不同的是，在給定的連接上 SQL Server 2000 驅動程式一次只能有一個使用中的 java.sql.Statement (http://java.sun.com/j2se/1.5.0/docs/api/java/sql/Statement.html) 物件。

  如果您嘗試使用一個以上的陳述式物件，則會發出下列錯誤：因爲具有複製的連接，所以無法啓動手動異動模式。此錯誤只有在驅動程式相容性參數 「重複使用陳述式？」，第 53 頁設爲布林值 True 時才會發生。最好的作法是，絕不明確設定此參數，而是使用動態預設值。

  或者，將分隔內容 ;SelectMethod=cursor 置於 URL 字串的結尾處。如需此問題的其他資訊，請諮詢下列支援文章：

  - DataDirect Technologies* 的 Document 30096 (http://knowledgebase.datadirect.com/kbase.nsf/SupportLink+Online/30096?OpenDocument)
  - Microsoft 的 Article 313181 (http://support.microsoft.com/default.aspx?scid=kb%3Ben-us%3B313181)

- 各驅動程式版本之間包含 UNIQUEIDENTIFIER 欄的關聯值是不一致的。

  較早版本的 SQL Server 2000 驅動程式針對原始 UNIQUEIDENTIFIER 欄傳回非標準的 java.sql.Types (http://java.sun.com/j2se/1.5.0/docs/api/java/sql/Types.html) 值。爲了彌補，Driver for JDBC 將該非標準類型映射到標準類型 java.sql.Types.BINARY (http://java.sun.com/j2se/1.5.0/docs/api/java/sql/Types.html)，因爲它可以鏡像複製 16 位元值的原始資料庫類型到最佳程度。此映射會產生 Base64 編碼的關聯值。

  更新版本的 SQL Server 2000 驅動程式會傳回標準類型 java.sql.CHAR (http://java.sun.com/j2se/1.5.0/docs/api/java/sql/Types.html)。此映射會產生非 Base64 編碼的關聯值，藉由使用較早版本 SQL Server 2000 驅動程式，使所產生的所有關聯失效。此變更事實上會中斷反向相容性。

此問題的最佳解決方案是繼續使用較早版本的 SQL Server 2000 驅動程式。如果您必須升級，請移除所有失效的關聯，並重新關聯所有之前關聯的物件。

6.4.8 MySQL Connector/J JDBC 驅動程式

表格 ***6-15*** *MySQL Connector/J JDBC* 驅動程式的設定

受支援的資料庫版本	3.x、4.x
類別名稱	org.gjt.mm.mysql.Driver
類型	4
URL 語法	jdbc:mysql:///*IP 位址* :3306/ *資料庫名稱*
下載指示	下載並擷取。jar 檔案位於*擷取目錄* /mysql-connector-java- *版本*目錄。 MySQL Connector/J (http://www.mysql.com/products/connector/j/)
檔名	mysql-connector-java- *版本* -bin.jar
文件 URL	MySQL Connector/J 文件 (http://dev.mysql.com/doc/refman/5.0/en/java-connector.html) 透過 SSL 進行連接 (http://dev.mysql.com/doc/refman/5.0/en/cj-using-ssl.html)

另請參閱 「產生 / 取回方法 (全域表格)」，第 62 頁。

相容性

Connector/J 驅動程式是反向相容的。資料庫伺服器會經常更新。驅動程式不經常更新。

安全性

Connector/J 驅動程式支援 JSSE (Java Secure Sockets Extension) SSL 加密傳輸。

MyISAM 表格的必要參數設定

下表列出您必須設定的驅動程式參數，如此 Driver for JDBC 便可以針對 MyISAM 表格與 Connector/J 驅動程式互通。

表格 ***6-16*** *MyISAM* 表格的設定

顯示名稱	標籤名稱	值
使用手動異動？	use-manual-transactions	false

6.4.9 Oracle Thin Client JDBC 驅動程式

表格 ***6-17*** *Oracle Thin Client* 設定

受支援的資料庫版本	8i、9i、10g
類別名稱	oracle.jdbc.driver.OracleDriver
類型	4
URL 語法	jdbc:oracle:thin:@*IP 位址*:1521:*sid*
下載指示	免費註冊並下載。 Oracle Technology Network (http://otn.oracle.com/software/tech/java/sqlj_jdbc/content.html)
1.1 檔名	classes111.zip、nls_charset11.zip (選擇性)
1.2-3 檔名	classes12.zip、ocrs12.zip (選擇性)、nls_charset12.zip (選擇性)
1.4 檔名	ojdbc14.jar、ocrs12.zip (選擇性)
文件 URL	Oracle 進階安全性 (http://www.cs.umb.edu/cs634/ora9idocs/network.920/a96573/asojbdc.htm) JDBC FAQ (http://www.oracle.com/technology/tech/java/sqlj_jdbc/htdocs/jdbc_faq.htm)

相容性

Thin Client 驅動程式是反向相容的。資料庫伺服器和驅動程式不經常更新。

Oracle 釋出各種 Java 虛擬機器 (JVM) 的 Thin Client 驅動程式。即使全都使用此產品，仍建議您使用 1.4 版。

安全性

Thin Client 驅動程式支援「Oracle 進階安全性」加密傳輸。

動態參數預設值

下表列出 Driver for JDBC 在執行時期以隱含方式設定的驅動程式相容性參數。請勿明確地置換這些設定。

表格 ***6-18*** 不可置換的 *Oracle Thin Client* 設定

顯示名稱	標籤名稱	值
傳回的結果集數目：	handle-stmt-results	single

連接內容

下表列出此驅動程式的重要連接內容。

表格 **6-19** *Oracle Thin Client*：連接內容

內容	重要性
includeSynonyms	如果此內容的值是 true，則可以使用同義字欄中繼資料。
ORACLE.NET.ENCRYPTION_CLIENT	定義用戶端要與伺服器交涉的安全性層級。
ORACLE.NET.ENCRYPTION_TYPES_CLIENT	定義要使用的加密演算法。
ORACLE.NET.CRYPTO_CHECKSUM_CLIENT	定義要針對資料完整性與伺服器交涉的安全性層級。
ORACLE.NET.CRYPTO_CHEKSUM_TYPES_CLIENT	定義要使用的資料完整性演算法。

已知問題

- 透過執行內嵌式 SQL 陳述式觸發的高 CPU 使用率：

  此驅動程式發生的最常見問題是高 CPU 使用率。因此，此驅動程式會一直指出，從呼叫到方法 java.sql.Statement.execute(String stmt) 有更多的結果，其可能導致無限迴路情形。只有在下列各項都發生時，才會發生此情形：

  - 正在執行之驅動程式相容性參數「傳回的結果集數目」，第 54 頁中，single、no 或 one 以外的值。
  - 正在執行內嵌式 SQL 陳述式。
  - 未明確指定陳述式類型。

  若要避免產生高 CPU 使用率，請遵循下列指示：

  - 不要明確地設定此參數。使用動態預設值。
  - 一直將 jdbc:type 屬性置於內嵌式 <jdbc:statement> 元素上。

    附註：jdbc 名稱空間字首必須映射至 urn:dirxml:jdbc。

- 無法取回同義字欄中繼資料：

  連接內容 includeSynonyms 必須設為 true。

- 無法查看同義字表格主索引鍵條件約束：

  此問題的唯一已知解決方案是使用檢視窗。

6.4.10 PostgreSQL JDBC 驅動程式

表格 **6-20** *PostgreSQL JDBC* 驅動程式設定

受支援的資料庫版本	6.x、7.x、8.x
類別名稱	org.postgresql.Driver
類型	4
URL 語法	jdbc:postgresql:///*IP 位址* :5432/ *資料庫名稱*
下載指示	JDBC 驅動程式下載 (http://jdbc.postgresql.org/download.html)

文件 URL	JDBC 驅動程式文件 (http://jdbc.postgresql.org/documentation/docs.html) 使用 SSL (http://jdbc.postgresql.org/documentation/80/ssl.html)

附註：PostgreSQL 檔名會依資料庫版本而不同。

相容性

PostgreSQL 驅動程式的最新組建是透過伺服器 7.2 版反向相容。資料庫伺服器和驅動程式會經常更新。

安全性

PostgreSQL 驅動程式支援 JDBC 3 驅動程式版本的 SSL 加密傳輸。

6.4.11 Sybase Adaptive Server Enterprise JConnect JDBC 驅動程式

表格 ***6-21*** *Sybase Adaptive Server Enterprise* 驅動程式的設定

受支援的資料庫版本	Adaptive Server* Enterprise 11.x、12.x
類別名稱	com.sybase.jdbc2.jdbc.SybDriver (針對 jconn2.jar) com.sybase.jdbc3.jdbc.SybDriver (針對 jconn3.jar)
類型	4
URL 語法	jdbc:sybase:Tds:*IP 位址* :2048/ *資料庫名稱*
下載指示	Sybase 下載 (http://www.sybase.com/detail?id=1009796)
檔名	jconn2.jar 或 jconn3.jar
文件 URL	jConnect 文件 (http://sybooks.sybase.com/onlinebooks/group-jc/jcg0600e/prjdbc)

相容性

Adaptive Server 驅動程式是反向相容的。資料庫伺服器和驅動程式不經常更新。

安全性

Adaptive Server 驅動程式支援 SSL 加密傳輸。若要啓用 SSL 加密，您必須透過 SYBSOCKET_FACTORY 連接內容指定自定插槽實作。如需設定連接內容的其他資訊，請參閱 「連接內容」，第 49 頁。

連接內容

SYBSOCKET_FACTORY 內容可用來指定支援加密傳輸之自定插槽實作的類別名稱。

6.5 不受支援的協力廠商 JDBC 驅動程式

6.5.1 IBM Toolbox for Java/JTOpen

表格 ***6-22*** *IBM Toolbox for Java/JTOpen* 的設定

資料庫	IBM Toolbox for Java/JTOpen ◆ iSeries Toolbox for Java (別名) ◆ AS/400 Toolbox for Java (別名)
類別名稱	com.ibm.as400.access.AS400JDBCDriver
類型	4
URL 語法	jdbc:as400://ip-address/database-name
下載指示	JTOpen 的下載 URL ◆ JTOpen (http://jt400.sourceforge.net) ◆ Toolbox for Java/JTOpen (http://www-03.ibm.com/servers/eserver/iseries/toolbox/downloads.html)
檔名	jt400.jar
文件 URL	Toolbox for Java/JTOpen (http://www-03.ibm.com/servers/eserver/iseries/toolbox/)

如果您是使用 IBM Toolbox for Java/JTOpen 驅動程式，則必須手動輸入「JDBC 驅動程式類別名稱」和「驗證網路位置」參數的值。設定不是自動填入的。請參閱「協力廠商 JDBC 驅動程式類別名稱」，第 42 頁和「驗證網路位置」，第 40 頁。

6.5.2 最小協力廠商 JDBC 驅動程式要求

Driver for JDBC 可能不會與所有協力廠商 JDBC 驅動程式互通。如果您使用不受支援的協力廠商 JDBC 驅動程式，則它必須符合下列要求：

- 支援必要的中繼資料方法

  如需 Driver for JDBC 所發出之必要和選擇性 java.sql.DatabaseMetaData 方法呼叫的目前清單，請參閱附錄 D 「java.sql.DatabaseMetaData 方法」，第 149 頁 。

- 支援其他必要的 JDBC 方法

  如需 Driver for JDBC 所使用之必要 JDBC 方法的清單，請參閱附錄 D 「java.sql.DatabaseMetaData 方法」，第 149 頁 。您可以結合使用此清單與協力廠商驅動程式文件，以識別可能的不相容性。

6.5.3 使用其他協力廠商 JDBC 驅動程式時的考量

- 因爲 Driver for JDBC 直接取決於協力廠商 JDBC 驅動程式的實作，所以那些實作中的錯誤會導致此產品發生故障。

  爲了協助您針對協力廠商 JDBC 驅動程式進行除錯，Driver for JDBC 支援下列各項：

  - 進行 JDBC 應用程式介面 (API) 層級 (層級 6) 的追蹤
  - 協力廠商 JDBC 驅動程式 (層級 7) 追蹤
- 支援預存程序或函數可能是失敗點。
- 您可能需要撰寫自定驅動程式描述元檔案。

  具體來說，您需要對正在使用之協力廠商驅動程式的錯誤碼和 SQL 狀態進行分類。

6.6 安全性問題

爲了確保 Identity Manager Driver for JDBC 與協力廠商驅動程式之間擁有安全連接，建議您進行下列操作：

- 以遠端的方式在資料庫伺服器上執行 Driver for JDBC。
- 使用 SSL 爲 Identity Manager 伺服器與資料庫伺服器之間的通訊加密。

如果您無法遠端執行 Driver for JDBC，則可以使用類型 2 或類型 3 JDBC 驅動程式。這些驅動程式類型通常會透過中介軟體伺服器或其他 JDBC 驅動程式類型無法使用的用戶端應用程式介面 (API)，以促進更高的安全性。部份類型 4 驅動程式支援加密的傳輸，但是加密是例外而非規則。

受支援的資料庫

7

7.1 資料庫互通

Identity Manager Driver for JDBC 是設計用來與特定的 JDBC 驅動程式實作 (而非特定的資料庫集) 進行互通。如此，受支援資料庫的清單主要會由受支援的協力廠商 JDBC 驅動程式功能所驅動。次要因素則是測試資源。

7.2 受支援的資料庫

下列資料庫或資料庫版本已經過測試，並建議與此產品搭配使用：

表格 ***7-1*** 受支援的資料庫

資料庫	次要版本
IBM DB2 Universal Database (UDB) 7	7.2 或更新版本
IBM DB2 Universal Database (UDB) 8	8.1 或更新版本
Informix Dynamic Server (IDS)	9.40 或更新版本
Microsoft SQL Server 7	7.5、Service Pack 4 或更新版本
Microsoft SQL Server 8 (2000)	Service Pack 3a 或更新版本
MySQL 3	3.23.58 或更新版本
MySQL 4	4.1 或更新版本
Oracle 8i	Release 3 (8.1.7) 或更新版本
Oracle 9i	Release 2 (9.2.0.1) 或更新版本
Oracle 10g	Release 1 (10.0.2.1) 或更新版本
PostgreSQL 7	7.4.6 或更新版本
Sybase Adaptive Server Enterprise (ASE) 12	12.5 或更新版本

您可以使用 Driver for JDBC 並搭配其他資料庫或資料庫版本。不過，Novell® 不支援它們。若要與 Driver for JDBC 互通，資料庫必須符合下列要求：

- 支援 SQL-92 項目層級文法。
- 可以由 JDBC 存取。

7.3 資料庫特性

7.3.1 資料庫功能

表格 *7-2* 資料庫功能

資料庫	綱要	檢視窗	身份欄	序列	預存程序	函數	觸發	取代觸發
IBM DB2 UDB 7	X	X	X	0	X[1]	X[1]	X	0
IBM DB2 UDB 8	X	X	X	0	X[1]	X[1]	X	X
Informix IDS 9	X	X	X[2]	0	X[3]	X	X	0
MS SQL 7	X	X	X	0	X	0	X	0
MS SQL 8	X	X	X	0	X	X	X	X
MySQL 4	0	0	X[4]	0	0	0	0	0
Oracle 8i、9i、10g	X	X	0	X	X	X	X	X
Postgres 7	X	X	X[5]	X	X	X	X[6]	X[6]
Sybase ASE 12	X	X	X	0	X	0	X	0

[1] DB2 原始支援以 Java 撰寫的預存程序或函數。若要藉由使用原始 SQL 程序語言來寫入程序，請在資料庫伺服器上安裝 C 編譯器。

[2] Informix 身份欄關鍵字爲 SERIAL8。

[3] Informix 預存程序無法傳回值。

[4] MySQL 身份欄關鍵字爲 AUTO_INCREMENT。

[5] 您可以使用 Postgres 序列物件將預設值提供給主索引鍵欄，從而有效地模擬身份欄。

Postgres 具有稱爲規則的原始建構元。這個建構元可用來有效地模擬觸發和取代觸發。它還支援使用以各種程序設計語言撰寫的觸發或取代觸發。

7.3.2 目前時戳陳述式

下表列出資料庫用來取回目前日期和時間的 SQL 陳述式：

表格 ***7-3*** 時戳陳述式

資料庫	目前時戳陳述式	ANSI 相容
IBM DB2 UDB	SELECT (CURRENT TIMESTAMP) FROM SYSIBM.SYSDUMMY1 FETCH FIRST 1 ROW ONLY	否
Informix IDS	SELECT FIRST 1 (CURRENT YEAR TO FRACTION(5)) FROM INFORMIX.SYSTABLES	否
MSSQL	SELECT (CURRENT_TIMESTAMP)	是
MySQL	SELECT (CURRENT_TIMESTAMP)	是
Oracle	SELECT (SYSDATE) FROM SYS.DUAL	否
PostgreSQL	SELECT (CURRENT_TIMESTAMP)	是
Sybase ASE	SELECT GETDATE()	否

7.3.3 預存程序和函數 JDBC 呼叫語法

下表列出用於呼叫預存程序或函數的 SQL 語法。對於在內嵌式 SQL 陳述式中格式化程序和函數呼叫而言，這會非常有用。

表格 ***7-4*** 呼叫預存程序或函數

資料庫	預存程序 / 函數 JDBC 呼叫語法
IBM DB2 UDB	{call *綱要名稱* . *程序名稱* (*參數清單*)}
Informix IDS	EXECUTE [PROCEDURE | FUNCTION] *綱要名稱* . *常式名稱* (*參數清單*)
MSSQL	EXECUTE *綱要名稱* . *程序名稱* (*參數清單*)
MySQL	(NA)
Oracle[1]	CALL *綱要名稱* . *程序名稱* (*參數清單*)
PostgreSQL	SELECT **綱要名稱** . **程序名稱** (**參數清單**)
Sybase ASE	EXECUTE *綱要名稱* . *程序名稱* (*參數清單*)

[1] Oracle 的 JDBC 實作不支援做爲字串呼叫函數。

7.3.4 左外部結合運算子

下表依資料庫列出外部結合運算子。

表格 *7-5* 外部結合運算子

資料庫	左外部結合運算子	ANSI 相容
IBM DB2 UDB	LEFT OUTER JOIN	是
Informix IDS	LEFT OUTER JOIN	是
MSSQL	*=	否
MySQL	LEFT OUTER JOIN	是
Oracle	(+)	否
PostgreSQL	LEFT OUTER JOIN	是
Sybase ASE	*=	否

附註：10g 版的 Oracle 支援 ANSI 相容的左外部結合運算子 LEFT OUTER JOIN。

7.3.5 未分隔的識別碼區分大小寫

表格 *7-6* 未分隔的識別碼區分大小寫

資料庫	區分大小寫？
IBM DB2 UDB	否
Informix IDS	否
MSSQL	否
MySQL	是
Oracle	否
PostgreSQL	否
Sybase ASE	是

7.3.6 受支援的異動隔離層級

表格 *7-7* 受支援的異動隔離層級

資料庫	無	讀取未認可	讀取認可	可重複讀取	可序列化	URL
IBM DB2 UDB	0	X	X[1]	X	X	設定 JDBC 異動隔離層級 (http://publib.boulder.ibm.com/infocenter/db2help/index.jsp?topic=/com.ibm.db2.udb.doc/ad/tjvjdiso.htm)

資料庫	無	讀取未認可	讀取認可	可重複讀取	可序列化	URL
MySQL (InnoDB 表格類型)	0	X	X	X[1]	X	InnoDB 異動隔離層級 (http://dev.mysql.com/doc/mysql/en/innodb-transaction-isolation.html)
Oracle	0	0	X[1]	0	X	JDBC 異動最佳化 (http://www.oracle.com/technology/oramag/oracle/02-jul/o42special_jdbc.html)
PostgreSQL	0	0[2]	X[1]	0[2]	X	異動隔離 (http://www.postgresql.org/docs/current/static/transaction-iso.html)

[1] 這是此資料庫的預設隔離層級。[2] 可以設定，但是別名爲受支援的隔離層級。

7.3.7 認可關鍵字

下表識別受支援資料庫的認可關鍵字：

表格 ***7-8*** 認可關鍵字

資料庫	認可關鍵字
IBM DB2 UDB	COMMIT
Informix IDS	COMMIT WORK[1]
MSSQL	GO
MySQL	COMMIT
Oracle	COMMIT
PostgreSQL	COMMIT
Sybase ASE	GO

[1] 適用於記錄和 ANSI 相容的資料庫。非記錄資料庫不支援異動。

7.3.8 IBM DB2 Universal Database (UDB)

下表列出此資料庫的內容。

表格 ***7-9*** *IBM DB2 UDB* 的內容

內容	值
目前時戳陳述式	SELECT (CURRENT TIMESTAMP) FROM SYSIBM.SYSDUMMY1 FETCH FIRST 1 ROW ONLY

內容	值
預存程序 / 函數呼叫語法	{call *綱要名稱* . *程序名稱* (*參數清單*)}
區分大小寫？	否
認可關鍵字	COMMIT
左外部結合運算子	LEFT OUTER JOIN

動態預設值

下表列出 Driver for JDBC 在執行時期以隱含方式設定的資料庫相容性參數。請勿明確地置換這些設定。

表格 ***7-10*** 以動態方式設定的 *IBM DB2 Universal Database* 設定

顯示名稱	標籤名稱	值
目前時戳陳述式：	current-timestamp-stmt	SELECT (CURRENT TIMESTAMP) FROM SYSIBM.SYSDUMMY1 FETCH FIRST 1 ROW ONLY
「時戳轉譯程式」類別：	time-translator-class	com.novell.nds.dirxml.driver.jdbc.db.DB2Timestamp

已知問題

- 時戳格式是專屬的。請參閱 「已知問題」，第 132 頁。

7.3.9 Informix Dynamic Server (IDS)

下表列出此資料庫的內容。

表格 ***7-11*** *Informix Dynamic Server* 的設定

內容	值
目前時戳陳述式	SELECT FIRST 1 (CURRENT YEAR TO FRACTION(5)) FROM INFORMIX.SYSTABLES
預存程序 / 函數呼叫語法	EXECUTE [PROCEDURE | FUNCTION] *綱要名稱* . *程序名稱* (參數清單)
區分大小寫？	否
認可關鍵字	COMMIT WORK[1]
左外部結合運算子	LEFT OUTER JOIN

[1] 適用於記錄和 ANSI 相容的資料庫。非記錄資料庫不支援異動。

動態預設值

下表列出 Driver for JDBC 在執行時期以隱含方式設定的資料庫相容性參數。請勿明確地覆寫這些設定。

表格 ***7-12*** *Informix Dynamic Server* 設定

顯示名稱	標籤名稱	值
目前時戳陳述式：	current-timestamp-stmt	SELECT FIRST 1 (CURRENT YEAR TO FRACTION(5)) FROM INFORMIX.SYSTABLES

已知問題

- 除非在建立表格時明確地將小數位數 (小數點右邊的位數) 設爲 0，否則無法將 NUMERIC 或 DECIMAL 欄用做主索引鍵。在預設狀態下，小數位數設爲 255。

7.3.10 Microsoft SQL Server

下表列出此資料庫的內容：

表格 ***7-13*** *Microsoft SQL Server* 的設定

內容	值
目前時戳陳述式	SELECT (CURRENT_TIMESTAMP)
預存程序 / 函數呼叫語法	EXECUTE *綱要名稱* . *程序名稱* (*參數清單*)
區分大小寫？	否
認可關鍵字	GO
左外部結合運算子	*=

動態預設值

下表列出 Driver for JDBC 在執行時期以隱含方式設定的資料庫相容性參數。請勿明確地覆寫這些設定。

表格 ***7-14*** 以動態方式設定的 *Microsoft SQL Server* 設定

顯示名稱	標籤名稱	值
插入時新增預設值？	add-default-values-on-view-insert	true
左外部結合運算子：	left-outer-join-operator	*=

7.3.11 MySQL

下表列出此資料庫的內容。

表格 ***7-15*** *MySQL* 的設定

內容	值
目前時戳陳述式	SELECT (CURRENT_TIMESTAMP)
預存程序 / 函數呼叫語法	(NA)
區分大小寫？	是
認可關鍵字	COMMIT
左外部結合運算子	LEFT OUTER JOIN

動態預設值

下表列出此資料庫在執行時期以動態方式設定的資料庫相容性參數。

表格 ***7-16*** 以動態方式設定的 *MySQL* 設定

顯示名稱	標籤名稱	值
支援在中繼資料取回時使用綱要？	supports-schemas-in-metadata-retrieval	false

已知問題

- 在將 TIMESTAMP 欄啓始設爲 0 或 NULL 之後進行更新時，會一律將其設爲目前的日期和時間。若要彌補此行爲，建議您將 「Identity Vault 時間和時戳」語法映射至 DATETIME 欄。

7.3.12 Oracle

下表列出此資料庫的內容：

表格 ***7-17*** *Oracle* 的設定

內容	值
目前時戳陳述式	SELECT (SYSDATE) FROM SYS.DUAL
預存程序 / 函數呼叫語法	CALL *綱要名稱* . *程序名稱* (*參數清單*)
區分大小寫？	否
認可關鍵字	COMMIT
左外部結合運算子	(+)

動態預設值

下表列出 Driver for JDBC 在執行時期以隱含方式設定的資料庫相容性參數。請勿明確地覆寫這些設定。

表格 **7-18** 以動態方式設定的 *Oracle* 設定

顯示名稱	標籤名稱	值
左外部結合運算子	left-outer-join-operator	(+)
排除過濾器運算式	exclude-table-filter	BIN\$.{22}==\$0
鎖定陳述式產生器類別	lock-generator-class	com.novell.nds.dirxml.driver.jdbc.db.lock.OraLockGenerator

附註：預設排除過濾器會從 Oracle 10g 中可見之同步化綱要略去的表格加以省略。

限制

- 在觸發中無法參考 LONG、LONG RAW 和 BLOB 欄。您無法藉由在觸發 (包含取代觸發) 中使用 :NEW 修飾詞來參考這些類型的欄。

7.3.13 PostgreSQL

下表列出此資料庫的內容：

表格 **7-19** *PostgreSQL* 的設定

內容	值
目前時戳陳述式	SELECT (CURRENT_TIMESTAMP)
預存程序 / 函數呼叫語法	SELECT *綱要名稱* . *程序名稱* (*參數清單*)
區分大小寫？	否
認可關鍵字	COMMIT
左外部結合運算子	LEFT OUTER JOIN

已知問題

- PostgreSQL 不支援 <check-object-password> 事件。您藉由手動將項目插入至 pg_hba.conf 檔案來控制驗證。

7.3.14 Sybase Adaptive Server Enterprise (ASE)

下表列出此資料庫的內容：

表格 **7-20** *Sybase ASE* 的設定

內容	值
目前時戳陳述式	SELECT GETDATE()
預存程序 / 函數呼叫語法	EXECUTE *綱要名稱* . *程序名稱* (*參數清單*)

內容	值
區分大小寫？	是
認可關鍵字	GO
左外部結合運算子	*=

動態預設值

下表列出 Driver for JDBC 在執行時期以隱含方式設定的資料庫相容性參數。請勿明確地覆寫這些設定。

表格 ***7-21*** 以動態方式設定的 *Sybase ASE* 設定

顯示名稱	標籤名稱	值
目前時戳陳述式	current-timestamp-stmt	SELECT GETDATE()
左外部結合運算子	left-outer-join-operator	*=
「時戳轉譯程式」類別	time-translator-class	com.novell.nds.dirxml.driver.jdbc.db.SybaseTimestamp

已知問題

- 填補和截短二進位值。

  若要確保二進位值 ANSI 相容的填補和截短行爲，請確定二進位欄類型 (而不是 IMAGE) 符合下列準則：

  - 它們的大小與映射至它們的 eDirectory™ 屬性大小完全相同。
  - 它們會限制爲 NOT NULL。
  - 它們會新增至 「發行者建立規則」和「訂閱者建立規則」。

  如果它們限制爲 NULL，則對 eDirectory 有效之行尾的零會截短。如果二進位欄超出它們各自的 eDirectory 屬性大小，則額外的 0 會附加至該值。

  建議的解決方案是在同步化二進位值時只使用 IMAGE 資料類型。
- 秒的 DATETIME 分數會四捨五入。「Sybase 時戳」最精確可到 1/300 秒 (大約 .003 秒)。資料庫伺服器則四捨五入到最接近的 1/300 秒，而不是 1/1000 秒 (.001 秒或 1 毫秒)。
- 時戳格式是專屬的。

關聯公用程式

8

「關聯公用程式」會使 Driver for JDBC 1.0 或更新版本相關聯物件的關聯正常化。它還會提供簡化驅動程式管理的幾種其他功能。

此版本的公用程式與 Driver for JDBC 1.0 和更新版本相容，並且取代所有先前的版本。

8.1 獨立操作

「關聯公用程式」支援七種獨立操作：

表格 ***8-1*** 獨立操作

操作	描述	讀寫功能
1	列出與驅動程式相關聯的物件 (預設值)。	唯讀
2	列出與驅動程式有多個關聯的物件。	唯讀
3	列出與驅動程式有無效關聯的物件。 關聯在下列情況下無效： ◆ 關聯格式錯誤。 例如，關聯遺失綱要 RDN、遺失表格 RDN 或綱要關鍵字拼錯。 ◆ 關聯包含未映射至目標資料庫中識別碼的資料庫識別碼。 例如，關聯包含對不存在之表格的映射。 ◆ 關聯未映射至任何列，或者映射至多重列。 如果關聯沒有映射至列，則它會中斷。同樣，如果關聯映射至一個列以上，則它們不是唯一的。	唯讀
4	列出需要正常化的物件。 正常化的關聯有效、順序正確並且使用正確的大小寫。正常情況下，對於不區分大小寫的資料庫而言爲大寫，而對於區分大小寫的資料庫而言爲大小寫混合。	唯讀
5	將操作 4 期間列出的物件關聯正常化。	寫入
6	列出要修改的物件關聯。 允許基於搜尋準則全域取代綱要、表格和欄名。 此操作需要兩個參數 (oldRDN 和 newRDN)。請參閱「編輯關聯」，第 135 頁。	唯讀

操作	描述	讀寫功能
7	修改在操作 6 期間列出的物件關聯。 此操作需要兩個參數 (oldRDN 和 newRDN)。請參閱 「編輯關聯」，第 135 頁。	寫入

8.2 開始之前

修改關聯可能會導致問題。如果關聯損毀，則 Identity Manager 會停止運作。因此，只在必要時才使用寫入操作。爲了避免不小心損毀關聯，「關聯公用程式」會爲所有寫入操作建立一個復原 ldiff 檔案。

使用公用程式之前檢視下列注意事項：

- 與驅動程式一樣，「關聯公用程式」會假設資料庫識別碼是未分隔的 (不使用引號且不包含任何特殊字元)。
- 一同更新與驅動程式相關聯的所有物件關聯。

  重要：極爲重要的是您同時會更新與驅動程式相關聯的所有物件關聯。

  若要查看與特定驅動程式相關聯的所有物件，請在與特定驅動程式例項相關聯的 Identity Manager 伺服器上執行「關聯公用程式」。

  LDAP 搜尋基礎必須包含與特定驅動程式相關聯的所有物件。

  附註：若要確保能完整包含，建議您將網路樹的根容器做爲搜尋基礎使用。
- 確保提供給此公用程式之目標資料庫的 JDBC URL 與驅動程式所使用的 URL 相同。當資料庫實際上是區分大小寫時將此公用程式指向不區分大小寫的資料庫，可能會產生正常化爲錯誤大小寫的關聯。
- 由於「關聯公用程式」是在本地端執行，所以它會使用不安全的連接。因此，Identity Vault LDAP 伺服器的組態必須暫時設定爲接受純文字密碼。視您所使用的協力廠商 JDBC 驅動程式而定，此公用程式所建立的資料庫連接可能不安全。

  附註：建議您在執行此公用程式之後，變更資料庫上驅動程式的驗證密碼。

8.3 使用關聯公用程式

針對安裝在 Identity Manager 伺服器上的每個驅動程式例項執行一次「關聯公用程式」。在安裝目錄 \jdbc\util 目錄中，批次檔案 association.bat 或 shell 程序檔 association.sh (依您的平台而定) 會啓動公用程式。

包含關聯公用程式參數的內容檔案會提供給每個受支援的資料庫。這些檔案位於 安裝目錄 \jdbc\util 目錄中。

資料庫	內容檔名
IBM DB2 Universal Database	properties_db2.txt
Informix Dynamic Server	properties_ifx_ansi.txt1 properties_ifx_log.txt properties_ifx_no_log.txt
Microsoft SQL Server	properties_ms.txt

資料庫	內容檔名
MySQL	properties_my.txt
Oracle	properties_ora.txt
PostgreSQL	properties_pg.txt
Sybase Adaptive Server Enterprise	properties_syb.txt

[1] 此公用程式不使用 Informix ANSI 相容的資料庫。

附註：如需如何從指令行執行公用程式的相關資訊，請參閱 安裝目錄 \tools\util 目錄中的 run.bat。

1. 停止驅動程式。
2. 執行 「關聯公用程式」以識別並移除無關的關聯 (操作 2 和 3)。

   與此產品相關聯的所有物件都不應具有多個關聯。請針對每個物件手動移除無關的關聯。操作 3 可協助您識別出多個關聯中實際上有效的關聯。在瞭解這種狀況之後，您可能會丟棄無關的關聯。
3. 執行 「關聯公用程式」以識別並修復無效的關聯 (操作 3，可能還有操作 6 和 7)。

   做爲一般規則，如果該問題是分離的，則手動編輯每個無效的關聯。如果問題重複且影響大量的關聯，請考慮使用操作 6 和 7。這個公用程式可以取代全域中錯誤的識別碼，但是無法在它們尚不存在的位置將其插入或移除。
4. 執行 「關聯公用程式」，以使關聯正常化 (操作 4 和 5)。

8.4 編輯關聯

「關聯公用程式」對於操作 6 和 7 (即搜尋和取代) 需要兩個參數 (oldRDN 和 newRDN)。

參數中的第一個值 (例如，schema) 是搜尋準則。第二個值 (例如，old) 是取代值。在某些情況下，您可以使用萬用字元 * 來使搜尋準則或取代值統一。

可以使用三種類型的搜尋和取代操作：

選項	描述	範例
取代綱要名稱	使用綱要 new 取代綱要 old (僅在右邊才支援萬用字元)。	oldRDN: schema=old newRDN: schema=new
取代表格名稱	使用表格 new 取代表格 old。不支援萬用字元。	oldRDN: table=old newRDN: table=new
取代欄名	使用 new 欄取代 old 欄。右邊需要使用萬用字元，但左邊不支援它們。	oldRDN: old=* newRDN: new=*

解除安裝 IDM Driver for JDBC

9

重要：我們建議您將預先設定的驅動程式和資料庫程序檔當作一體來進行安裝或解除安裝。爲了防止產生無意的不相符，資料庫程序檔和預先設定的驅動程式會包含具有版本編號、目標資料庫名稱和資料庫版本的標題。

9.1 刪除 IDM 驅動程式物件

刪除 Novell® Identity Vault 物件時，您必須先刪除所有子物件，然後才能刪除父物件。例如，您必須先刪除「發行者」通道上的所有規則和樣式表，才能刪除「發行者」物件。同樣地，您必須先刪除「發行者」和「訂閱者」物件，然後才能刪除「驅動程式」物件。

若要從 Identity Vault 移除驅動程式物件，請執行下列動作：

1. 在 Novell iManager 中，按一下「*Identity Manager*」>「*Identity Manager* 概觀」。
2. 選取驅動程式集。
3. 從「Identity Manager 概觀」頁中，按一下「刪除驅動程式」。
4. 選取您要刪除的驅動程式，然後按一下「確定」。

9.2 執行產品解除安裝程式

解除安裝的程序會依平台而異。

若要在 Windows 上解除安裝 Identity Manager Driver for JDBC，請使用「控制台」中的「新增或移除程式」。

9.3 執行資料庫解除安裝程序檔

本節可協助您執行資料庫解除安裝 SQL 程序檔。

- 「安裝 IBM DB2 Universal Database (UDB)」，第 32 頁
- 「安裝 Informix Dynamic Server (IDS)」，第 32 頁
- 「安裝 Microsoft SQL Server」，第 33 頁
- 「MySQL 解除安裝」，第 138 頁
- 「安裝 Oracle」，第 34 頁
- 「安裝 PostgreSQL」，第 34 頁
- 「安裝 Sybase Adaptive Server Enterprise (ASE)」，第 34 頁

9.3.1 IBM DB2 Universal Database (UDB) 解除安裝

DB2 的目錄網路位置是 安裝目錄 \jdbc\sql\db2_udbl\install。

1. 略去 idm、indirect 和 direct 作業系統使用者帳戶。
2. 如果您尚未進行此操作，請變更安裝程序檔中的管理員帳戶名稱和密碼。
3. 使用指令行處理器 (CLP) 執行程序檔 uninstall.sql。

   例如：db2 -f uninstall.sql

   重要：此程序檔不會在第 7 版以後版本的「指令中心」介面上執行。它會使用 僅 ' 行接續字元。「指令中心」的後來版本無法辨識此字元。

   ---

4. 刪除 idm_db2.jar 檔案。

9.3.2 Informix Dynamic Server (IDS) 解除安裝

Informix SQL 程序檔的目錄網路位置是 安裝目錄 \jdbc\sql\informix_ids\install。

1. 略去 idm 作業系統使用者帳戶。
2. 啓動用戶端，例如「SQL 編輯器」。
3. 以使用者 informix 或其他具有資料庫管理員 (DBA) 權限的使用者身份登入伺服器。

   在預設狀態下，informix 的密碼爲 informix。

   如果您以 informix 之外的使用者身份執行程序檔，請在執行之前先變更安裝程序檔內所有的 informix 參考。
4. 如果您不使用具有預設密碼的 informix 帳戶，請變更安裝程序檔中的資料庫管理員 (DBA) 帳戶名稱和密碼 (如果您尚未進行此操作)。
5. 根據您所安裝的資料庫類型而定，從 ansi (執行屬性、ANSI 相容)、log (執行屬性、非 ANSI 相容) 或 no_log (非執行屬性、非 ANSI 相容) 子目錄開啓並執行 uninstall.sql。

9.3.3 Microsoft SQL Server 解除安裝

Microsoft SQL Server 程序檔的目錄網路位置是安裝目錄 \jdbc\sql\mssql\install。

1. 啓動用戶端，例如「查詢分析器」。
2. 以使用者 sa 的身份登入您的資料庫伺服器。

   在預設狀態下，sa 使用者不具有密碼。
3. 開啓並執行第一個安裝程序檔 uninstall.sql。

   「查詢分析器」中的執行快速鍵是 F5。

9.3.4 MySQL 解除安裝

MySQL SQL 程序檔的目錄網路位置是 安裝目錄 \jdbc\sql\mysql\install。

1. 從 MySQL 用戶端 (例如 mysql)，以使用者 root 或者其他具有管理權限的使用者身份登入。

   例如，從指令行執行 mysql -u root -p

在預設狀態下，root 使用者不具有密碼。

2 執行解除安裝程序檔 uninstall.sql。

例如：mysql> \. c:\uninstall.sql

請勿使用分號來終止此陳述式。

9.3.5 Oracle 解除安裝

Oracle SQL 程序檔的目錄網路位置是 install-dir\jdbc\sql\oracle\install。

1 從 Oracle 用戶端 (例如 SQL Plus)，以使用者 SYSTEM 身份登入。

在預設狀態下，SYSTEM 的密碼是 MANAGER。

如果您以具有密碼 MANAGER 之 SYSTEM 以外的使用者身份執行程序檔，請在執行之前變更程序檔中 SYSTEM 的所有參考。

2 執行解除安裝程序檔 uninstall.sql。

例如：SQL> @c:\uninstall.sql

9.3.6 PostgreSQL 解除安裝

PostgreSQL 程序檔的目錄網路位置是 安裝目錄 \jdbc\sql\postgres\install。執行 Postgres 指令的目錄網路位置是 *Postgres* 安裝目錄 /pgsql/bin。

1 從 Postgres 用戶端 (例如 psql)，以使用者 postgres 身份登入 idm 資料庫。

例如，從 UNIXC 指令行執行 ./psql -d idm postgres

在預設狀態下，Postgres 使用者不具有密碼。

2 從 psql 內，執行程序檔 uninstall.sql。

例如：idm=# \i uninstall.sql

3 略去資料庫 idm。

例如，從 UNIX 指令行執行 ./dropdb idm

4 從 pg_hba.conf 檔案移除或註解 idm 使用者的項目。

例如：

```
#host    idm          idm    255.255.255.255   255.255.255.0
```

5 重新啓動 Postgres 伺服器，以使對 pg_hba.conf 檔案進行的變更生效。

9.3.7 Sybase Adaptive Server Enterprise (ASE) 解除安裝

Sybase SQL 程序檔的目錄網路位置是 安裝目錄 \jdbc\sql\sybase_ase\install。

1 從 Sybase 用戶端 (例如 isql)，以使用者 sa 身份登入。

2 執行安裝程序檔 uninstall.sql。

例如，從指令行執行 isql -U sa -P -i uninstall.sql

在預設狀態下，sa 帳戶沒有密碼。

最佳作法

A

下節列出使用 Driver for JDBC 的重要最佳作法。您可以在第 4 章 「設定 Identity Manager Driver for JDBC 組態」, 第 37 頁和第 5 章 「進階組態」, 第 77 頁中找到其他資訊。

安全性 / 效能：

- 基於效能和安全性的原因，請儘可能在資料庫伺服器上以遠端方式執行驅動程式。務必在 Identity Vault 和「遠端載入器」服務之間啓用 SSL 加密。
- 每當 Driver for JDBC 沒有在資料庫伺服器上遠端執行時，都應該啓用協力廠商驅動程式的 SSL 加密。如需受支援協力廠商驅動程式之安全性功能的相關資訊，請參閱「協力廠商 JDBC 驅動程式」，第 107 頁。
- 在生產環境中，關閉追蹤。

其他：

- 若爲直接同步化，將 "pk_" (不區分大小寫) 做爲一或多個檢視窗欄名的字首。
- 若爲直接和間接同步化，在邏輯資料庫類別間使用不同的主索引鍵欄名。
- 如果置於事件記錄 table_key 欄位中的主索引鍵值包含下列字元，則將它們分隔 (用雙引號)：, ; ' + = \ " < > 只有在主索引鍵欄爲二進位類型時才這樣操作。
- 當 Identity Vault 爲主索引鍵值的授權來源時，建議將 GUID (而非 CN) 做爲主索引鍵使用。不同於 CN，GUID 爲單一值並且不會變更。
- 從連結子表格和父表格的發行觸發外部索引鍵欄省略。
- 如果主索引鍵欄是靜態的 (它們不會變更)，請勿將它們包含在發行觸發中。
- 將 jdbc:type="query" 屬性值置於所有內嵌式 SELECT 陳述式上。將 jdbc:type="update" 屬性值置於所有內嵌式 INSERT、UPDATE 和 DELETE 陳述式上。

FAQ

B

B.1 無法查看表格或檢視窗

問題：爲何驅動程式無法看到我的表格或檢視窗？

回答：驅動程式只能同步化具有明確主索引鍵條件約束的表格，以及含有一或多個字首爲 "pk_" (不區分大小寫) 之欄位的檢視窗。驅動程式使用這些條件約束來決定在建構關聯時要使用的欄位。因此，驅動程式會忽略所有未限制的表格。如果您嘗試同步化缺少必要條件約束的表格或檢視窗，請新增這些條件約束或者同步化爲具有必要條件約束的中介表格。

另一種可能性是驅動程式缺少查看表格的必要資料庫權限。通常，可見度取決於是否具有 SELECT 權限。

B.2 與表格一起同步化

問題：如何與位於多個綱要中的表格同步化？

回答：請進行下列其中一項操作：

- 將表格別名化至同步化綱要。
- 同步化至同步化綱要中的中介表格，並且跨綱要邊界移動資料。
- 使用檢視窗。
- 使用「表格 / 檢視窗名稱」參數建立虛擬綱要。
  請參閱 「表格 / 檢視窗名稱」，第 47 頁。

B.3 處理事件記錄表格中的列

問題：爲何驅動程式不處理「事件記錄表格」中的列？

回答：請進行下列幾項操作：

1. 檢查相關列的 perpetrator 欄位，並確定值未設爲驅動程式的資料庫使用者名稱。

如果「發行者」通道「允許迴路」參數設爲布林值 False (預設值)，則「發行者」通道會檢查 perpetrator 欄位以偵測迴路事件。請參閱 「允許迴路？」，第 71 頁。

「允許迴路」參數設爲布林值 False 時，「發行者」通道會忽略 perpetrator 欄位值等於驅動程式資料庫使用者名稱的所有記錄。驅動程式資料庫使用者名稱是使用「驗證 ID」參數指定的。請參閱 「驗證 ID」，第 39 頁。

2 確定記錄的 status 欄位設爲 N (新)。

不會處理 status 欄位未設爲 N 的記錄。

3 確定有明確地認可變更。

變更在明確認可之前通常是暫定的。

B.4 管理資料庫使用者帳戶

問題：驅動程式可以管理資料庫使用者帳戶嗎？

回答：可以。您可以使用內嵌式 SQL 來管理資料庫帳戶。如需相關資訊，請參閱 「在 XDS 事件中內嵌 SQL 陳述式」，第 95 頁。

B.5 同步化大型資料類型

問題：驅動程式可以同步化大型二進位和字串資料類型嗎？

回答：可以。可以訂閱和發行大型二進位和字串資料類型。使用查詢回覆事件類型，可以發行大型二進位和字串資料類型。如需其他資訊，請參閱 「事件類型」，第 90 頁。

B.6 發行緩慢

問題：爲何發行緩慢？

回答：如果事件記錄表格包含大量的列，請爲表格編製索引。所有資料庫安裝程序檔中都會提供範例索引。藉由使用追蹤層級 3，您可以檢視驅動程式用來維護事件記錄的陳述式。

您還可以進一步修改安裝程序檔中的索引，以改進發行效能。將索引置於事件記錄表格之外的表格空間或實體磁碟，也可以改進發行效能。

此外，在生產環境中，除非將處理的列定期移至其他表格，否則將「刪除已處理的列」參數設爲布林值 False。請參閱 「刪除已處理的列？」，第 70 頁。

B.7 同步化多個類別

問題：驅動程式可以同步化多個類別嗎？

回答：可以。然而，主索引鍵欄名在邏輯資料庫類別之間必須是唯一的。例如，如果 *class1* 映射至主索引鍵欄名是 *key1* 的 *table1*，*class2* 映射至主索引鍵欄名是 *key2* 的 *table2*，則 *key1* 的名稱不可以等於 *key2*。

無論使用哪個同步化模型，始終都要能滿足這個要求。

B.8 加密的傳輸

問題：驅動程式支援加密的傳輸嗎？

回答：不支援。驅動程式與給定資料庫的通訊方式是根據正在使用的協力廠商驅動程式而定。部份協力廠商驅動程式支援加密的傳輸，而其他的則不支援。即使支援加密的傳輸，也不存在標準化的方式可在協力廠商 JDBC 驅動程式之間啓用加密。

這問題的一般解決方案是在遠端執行 Driver for JDBC 和協力廠商驅動程式。此方法允許 Driver for JDBC 和協力廠商驅動程式同時在資料庫伺服器本地上執行。然後，使用 SSL 將在 Metadirectory 引擎和 Driver for JDBC 之間網路傳輸的所有資料加密。

另一種可能的方法是使用類型 3 或類型 2 協力廠商 JDBC 驅動程式。資料庫中介軟體和用戶端 API 通常會提供加密的傳輸機制。

B.9 映射多重值屬性

問題：如何將多重值屬性映射至單一值資料庫欄位？

回答：請參閱「將多值屬性映射至單一值資料庫欄位」，第 86 頁。

B.10 同步化亂碼字串

問題：驅動程式爲何要同步化亂碼字串？

回答：資料庫和協力廠商驅動程式可能正在使用不相容的字元編碼。請調整協力廠商驅動程式所使用的字元編碼。

如需相關資訊，請參閱 Sun 定義的字元編碼值 (http://java.sun.com/j2se/1.5.0/docs/guide/intl/encoding.doc.html)。

B.11 執行多個 Driver for JDBC 例項

問題：如何在同一驅動程式集中執行多個 Driver for JDBC 例項？ 例項需要同一個協力廠商 JBDC 驅動程式的不同版本 (例如，Oracle JDBC 驅動程式或 IBM DB2 Type 3 JDBC 驅動程式)。

回答：使用「遠端載入器」在個別的「Java 虛擬機器 (JVM)」中載入每個 Driver for JDBC 例項。當在同一個 Java 虛擬機器 (JVM) 中本地執行時，相同協力廠商類別的不同版本會發生衝突。

受支援的資料類型

C

Driver for JDBC 可以同步化所有 JDBC 1 資料類型和 JDBC 2 資料類型的小型子集。JDBC 資料類型映射至資料庫原生資料類型的方式，是根據協力廠商驅動程式而定。

下列清單包含受支援的 JDBC 1 java.sql.Types (http://java.sun.com/j2se/1.5.0/docs/api/java/sql/Types.html)。

- 數值類型：
  - java.sql.Types.BIGINT
  - java.sql.Types.BIT
  - java.sql.Types.DECIMAL
  - java.sql.Types.DOUBLE
  - java.sql.Types.NUMERIC
  - java.sql.Types.REAL
  - java.sql.Types.FLOAT
  - java.sql.Types.INTEGER
  - java.sql.Types.SMALLINT
  - java.sql.Types.TINYINT
- 字串類型：
  - java.sql.Types.CHAR
  - java.sql.Types.LONGCHAR
  - java.sql.Types.VARCHAR
- 時間類型：
  - java.sql.Types.DATE
  - java.sql.Types.TIME
  - java.sql.Types.TIMESTAMP
- 二進位類型：
  - java.sql.Types.BINARY
  - java.sql.Types.VARBINARY
  - java.sql.Types.LONGVARBINARY

下列清單包含受支援的 JDBC 2 java.sql.Types (http://java.sun.com/j2se/1.5.0/docs/api/java/sql/Types.html)。

- 大型物件 (LOB) 類型：
  - java.sql.Types.CLOB
  - java.sql.Types.BLOB

java.sql.DatabaseMetaData 方法

D

本節列出必要的和選擇性的 java.sql.DatabaseMetaData (http://java.sun.com/j2se/1.5.0/docs/api/java/sql/DatabaseMetaData.html) 方法。

只有在同步化過濾器參數未設爲「排除所有表格 / 檢視窗」時，才需要下列 JDBC 1 方法：

- getColumns(java.lang.String catalog, java.lang.String schemaPattern, java.lang.String tableNamePattern, java.lang.String columnNamePattern):java.sql.ResultSet
- getPrimaryKeys(java.lang.String catalog, java.lang.String schema, java.lang.String table):java.sql.ResultSet
- getTables(java.lang.String catalog, java.lang.String schemaPattern, java.lang.String tableNamePattern, java.lang.String[] types):java.sql.ResultSet
- storesLowerCaseIdentifiers():boolean
- storesMixedCaseIdentifiers():boolean
- storesUpperCaseIdentifiers():boolean

選擇性的 JDBC 1 方法：

- dataDefinitionCausesTransactionCommit():boolean
- dataDefinitionIgnoredInTransactions():boolean
- getColumnPrivileges(String catalog, String schema, String table, String columnNamePattern):java.sql.ResultSet
- getDatabaseProductName():java.lang.String
- getDatabaseProductVersion():java.lang.String
- getDriverMajorVersion():int
- getDriverMinorVersion():int
- getDriverName():java.lang.String
- getDriverVersion():java.lang.String
- getExportedKeys(java.lang.String catalog, java.lang.String schema, java.lang.String table):java.sql.ResultSet
- getMaxStatements():int
- getMaxConnections():int
- getMaxColumnsInSelect():int
- getProcedureColumns(String catalog, String schemaPattern, String procedureNamePattern, String columnNamePattern):java.sql.ResultSet
- getSchemas():java.sql.ResultSet
- getTableTypes():java.sql.ResultSet
- getUserName():java.lang.String
- supportsColumnAliasing():bolean
- supportsDataDefinitionAndDataManiuplationTransactions():boolean

- supportsDataManipulationTransactionsOnly():boolean
- supportsLimitedOuterJoins():boolean
- supportsMultipleTransactions():boolean
- supportsSchemasInDataManipulation():boolean
- supportsSchemasInProcedureCalls():boolean
- supportsTransactionIsolationLevel(int level):boolean
- supportsTransactions():boolean

選擇性的 JDBC 2 方法：

- supportsBatchUpdates():boolean

選擇性的 JDBC 3 方法：

- supportsGetGeneratedKeys():boolean

JDBC 介面方法

E

本節列出 Driver for JDBC 使用的 JDBC 介面方法 (非 java.sql.DatabaseMetaData (http://java.sun.com/j2se/1.5.0/docs/api/java/sql/DatabaseMetaData.html) 方法)。方法依類別組織。

協力廠商 JDBC 驅動程式廠商經常會依方法列出瑕疵或已知問題。您可以結合使用下列方法與協力廠商 JDBC 驅動程式文件，以排解疑難或預期可能的互通問題。

- java.sql.DriverManager (http://java.sun.com/j2se/1.5.0/docs/api/java/sql/DriverManager.html)
- java.sql.CallableStatement (http://java.sun.com/j2se/1.5.0/docs/api/java/sql/CallableStatement.html)
- java.sql.Connection (http://java.sun.com/j2se/1.5.0/docs/api/java/sql/Connection.html)
- java.sql.PreparedStatement (http://java.sun.com/j2se/1.5.0/docs/api/java/sql/PreparedStatement.html)
- java.sql.ResultSet (http://java.sun.com/j2se/1.5.0/docs/api/java/sql/ResultSet.html)
- java.sql.ResultSetMetaData (http://java.sun.com/j2se/1.5.0/docs/api/java/sql/ResultSetMetaData.html)
- java.sql.Statement (http://java.sun.com/j2se/1.5.0/docs/api/java/sql/Statement.html)
- java.sql.Timestamp (http://java.sun.com/j2se/1.5.0/docs/api/java/sql/Timestamp.html)

下表列出 Driver for JDBC 使用的 java.sql.DriverManager (http://java.sun.com/j2se/1.5.0/docs/api/java/sql/DriverManager.html) 方法：

表格 ***E-1*** *java.sql.DriverManager 方法*

方法簽名	JDBC 版本	必要？
getConnection(String url, java.util.Properties info):java.sql.Connection	1	是 [1]
getConnection(String url, java.util.Properties info):java.sql.Connection	1	是 [1]
setLogStream(java.io.PrintStream out):void	1	否

[1] 一種方法或其他方法。

下表列出 Driver for JDBC 使用的 java.sql.CallableStatement (http://java.sun.com/j2se/1.5.0/docs/api/java/sql/CallableStatement.html) 方法：

表格 ***E-2*** *java.sql.CallableStatement 方法*

方法簽名	JDBC 版本	必要？
getBigDecimal(int parameterIndex, int scale):java.math.BigDecimal	1	是
getBoolean(int parameterIndex):boolean	1	是

方法簽名	JDBC 版本	必要？
getBoolean(String parameterName):boolean	3	否
getByte(int parameterIndex):byte	1	是
getByte(String parameterName):byte	3	否
getBytes(int parameterIndex):byte[]	1	是
getBytes(String parameterName):byte[]	3	否
getDate(int parameterIndex):java.sql.Date	1	是
getDate(String parameterName):java.sql.Date	3	否
getDouble(int parameterIndex):double	1	是
getDouble(String parameterName):double	3	否
getFloat(int parameterIndex):float	1	是
getFloat(String parameterName):float	3	否
getInt(int parameterIndex):int	1	是
int getInt(String parameterName)	3	否
getLong(int parameterIndex):long	1	是
getLong(String parameterName):long	3	否
getShort(int parameterIndex):short	1	是
getShort(String parameterName):short	3	否
getString(int parameterIndex):String	1	是
getString(String parameterName):String	3	否
getTime(int parameterIndex):java.sql.Time	1	是
getTime(String parameterName):java.sql.Time	3	否
getTimestamp(int parameterIndex):java.sql.Timestamp	1	是
getTimestamp(String parameterName):java.sql.Timestamp	3	否
registerOutParameter(int parameterIndex, int sqlType):void	1	是
wasNull():boolean	1	是

下表列出 Driver for JDBC 使用的 java.sql.Connection (http://java.sun.com/j2se/1.5.0/docs/api/java/sql/Connection.html) 方法：

表格 ***E-3*** *java.sql.Connection* 方法

方法簽名	JDBC 版本	必要？
close():void	1	是

方法簽名	JDBC 版本	必要？
commit():void	1	否
createStatement():java.sql.Statement	1	是
getAutoCommit():boolean	1	否
getMetaData():java.sql.DatabaseMetaData	1	是
getTransactionIsolation():int	1	否
getWarnings():java.sql.SQLWarning	1	否
isClosed():boolean	1	否
prepareCall(String sql):java.sql.CallableStatement	1	否
prepareStatement(String sql):java.sql.PreparedStatement	1	是
rollback():void	1	否
setAutoCommit(boolean autoCommit):void	1	否
setTransactionIsolation(int level):void	1	否

下表列出 Driver for JDBC 使用的 java.sql.PreparedStatement (http://java.sun.com/j2se/1.5.0/docs/api/java/sql/PreparedStatement.html) 方法：

表格 ***E-4*** *java.sql.PreparedStatement* 方法

方法簽名	JDBC 版本	必要？
clearParameters() :void	1	否
execute():boolean	1	是
executeQuery():java.sql.ResultSet	1	是
executeUpdate():int	1	是
setBigDecimal(int parameterIndex, java.math.BigDecimal x):void	1	是
setBoolean(int parameterIndex, boolean x):void	1	是
setByte(int parameterIndex, byte x):void	1	是
setBytes(int parameterIndex, byte x[]):void	1	是
setDate(int parameterIndex, java.sql.Date x):void	1	是
setDouble(int parameterIndex, double x):void	1	是
setFloat(int parameterIndex, float x):void	1	是
setInt(int parameterIndex, int x):void	1	是
setLong(int parameterIndex, long x):void	1	是
setNull(int parameterIndex, int sqlType):void	1	是
setShort(int parameterIndex, short x):void	1	是

方法簽名	JDBC 版本	必要？
setString(int parameterIndex, String x):void	1	是
setTime(int parameterIndex, java.sql.Time x):void	1	是
setTimestamp(int parameterIndex, java.sql.Timestamp x):void	1	是

下表列出 Driver for JDBC 使用的 java.sql.ResultSet (http://java.sun.com/j2se/1.5.0/docs/api/java/sql/ResultSet.html) 方法：

表格 ***E-5*** *java.sql.ResultSet* 方法

方法簽名	JDBC 版本	必要？
close():void	1	是
getBigDecimal(int columnIndex, int scale):java.math.BigDecimal	1	是
getBigDecimal(String columnName, int scale):java.math.BigDecimal	1	是
getBinaryStream(int columnIndex):java.io.InputStream	1	是
getBinaryStream(String columnName)java.io.InputStream	1	是
getBoolean(int columnIndex):boolean	1	是
getBoolean(String columnName):boolean	1	是
getByte(int columnIndex):byte	1	是
getByte(String columnName):byte	1	是
getBytes(int columnIndex):byte[]	1	是
getBytes(String columnName):byte[]	1	是
getDate(int columnIndex):java.sql.Date	1	是
getDate(String columnName)java.sql.Date	1	是
getFloat(int columnIndex):float	1	是
getFloat(String columnName):float	1	是
getInt(int columnIndex):int	1	是
getInt(String columnName):int	1	是
getLong(int columnIndex):long	1	是
getLong(String columnName):long	1	是
getMetaData():java.sql.ResultSetMetaData	1	否
getShort(int columnIndex):short	1	是
getShort(String columnName):short	1	是
getString(int columnIndex):String	1	是
getString(String columnName):String	1	是

方法簽名	JDBC 版本	必要？
getTime(int columnIndex):java.sql.Time	1	是
getTime(String columnName):java.sql.Time	1	是
getTimestamp(int columnIndex):java.sql.Timestamp	1	是
getTimestamp(String columnName):java.sql.Timestamp	1	是
getWarnings():java.sql.SQLWarning	1	否

下表列出 Driver for JDBC 使用的 java.sql.ResultSetMetaData (http://java.sun.com/j2se/1.5.0/docs/api/java/sql/ResultSetMetaData.html) 方法：

表格 ***E-6*** *java.sql.ResultSetMetaData* 方法

方法簽名	JDBC 版本	必要？
getColumnCount():int	1	是
getColumnName(int column):String	1	否
getColumnType(int column):int	1	否

下表列出 Driver for JDBC 使用的 java.sql.Statement (http://java.sun.com/j2se/1.5.0/docs/api/java/sql/Statement.html) 方法：

表格 ***E-7*** *java.sql.Statement* 方法

方法簽名	JDBC 版本	必要？
addBatch(java.lang.String sql):void	2	否
clearBatch():void	2	否
clearWarnings():void	1	否
close():void	1	是
execute(java.lang.String sql):boolean	1	是
executeBatch():int[]	2	否
executeUpdate(String sql):int	1	是
executeQuery(String sql):java.sql.ResultSet	1	是
getGeneratedKeys():java.sql.ResultSet	3	否
getMoreResults():boolean	1	否
getResultSet():java.sql.ResultSet	1	是
getUpdateCount():int	1	否
getWarnings():java.sql.SQLWarning	1	否

下表列出 Driver for JDBC 使用的 java.sql.Timestamp (http://java.sun.com/j2se/1.5.0/docs/api/java/sql/Timestamp.html) 方法：

表格 ***E-8*** *java.sql.Timestamp* 方法

方法簽名	JDBC 版本	必要？
getNanos():int	1	是
getTime():long	1	是
setNanos(int n):void	1	是
setTime(long time):void	1	是
toString ():String	1	是

協力廠商 JDBC 驅動程式描述元 DTD

F

本節包含協力廠商 JDBC 描述元檔案的文件類型定義 (DTD)。

```
<?xml version="1.0" encoding="UTF-8"?> <!ELEMENT actions (exec-sql |
check-for-closed-connection | fetch-metadata | rollback)*> <!ELEMENT
add-default-values-on-view-insert (#PCDATA)> <!ELEMENT authentication
(sql-state | error-code | sql-state-class | error-code-range |
actions)*> <!ELEMENT check-for-closed-connection EMPTY> <!ELEMENT
column-position-comparator (#PCDATA)> <!ELEMENT connection-properties
(property*)> <!ELEMENT connectivity (sql-state | error-code | sql-
state-class | error-code-range | actions)*> <!ELEMENT current-
timestamp-stmt (#PCDATA)> <!ELEMENT error-code (value)> <!ATTLIST
error-code description CDATA #IMPLIED > <!ELEMENT error-code-range
(from, to)> <!ATTLIST error-code-range description CDATA #IMPLIED >
<!ELEMENT errors (connectivity | authentication | retry | fatal)*>
<!ELEMENT exclude-table-filter (#PCDATA)> <!ELEMENT exec-sql
(#PCDATA)> <!ELEMENT fatal (sql-state | error-code | sql-state-class |
error-code-range | actions)*> <!ELEMENT fetch-metadata EMPTY>
<!ELEMENT from (#PCDATA)> <!ELEMENT function-return-method (#PCDATA)>
<!ELEMENT handle-stmt-results (#PCDATA)> <!ELEMENT identity (name?,
target-database?, jdbc-type?, jdbc-class?)> <!ELEMENT import
(#PCDATA)> <!ELEMENT imports (import*)> <!ELEMENT include-table-filter
(#PCDATA)> <!ELEMENT jdbc-class (#PCDATA)> <!ELEMENT jdbc-driver
(imports?, identity, (metadata-override | connection-properties | sql-
type-map | options | errors)*)> <!ELEMENT jdbc-type (#PCDATA)>
<!ELEMENT key (#PCDATA)> <!ELEMENT left-outer-join-operator (#PCDATA)>
<!ELEMENT lock-generator-class (#PCDATA)> <!ELEMENT metadata-override
(supports-schemas-in-procedure-calls?)> <!ELEMENT minimal-metadata
(#PCDATA)> <!ELEMENT name (#PCDATA)> <!ELEMENT options (lock-
generator-class | supports-schemas-in-metadata-retrieval | time-
translator-class | column-position-comparator | use-manual-
transactions | minimal-metadata | transaction-isolation-level | use-
single-connection | exclude-table-filter | include-table-filter |
left-outer-join-operator | current-timestamp-stmt | add-default-
values-on-view-insert | reuse-statements | function-return-method |
handle-stmt-results)*> <!ELEMENT property (key, value)> <!ELEMENT
retry (sql-state | error-code | sql-state-class | error-code-range |
actions)*> <!ELEMENT reuse-statements (#PCDATA)> <!ELEMENT rollback
EMPTY> <!ELEMENT sql-state (value)> <!ATTLIST sql-state description
CDATA #IMPLIED > <!ELEMENT sql-state-class (value)> <!ATTLIST sql-
state-class description CDATA #IMPLIED > <!ELEMENT sql-type-map
(type*)> <!ELEMENT supports-schemas-in-metadata-retrieval (#PCDATA)>
<!ELEMENT supports-schemas-in-procedure-calls (#PCDATA)> <!ELEMENT
target-database (#PCDATA)> <!ELEMENT time-translator-class (#PCDATA)>
<!ELEMENT to (#PCDATA)> <!ELEMENT transaction-isolation-level
(#PCDATA)> <!ELEMENT type (from, to)> <!ELEMENT use-manual-
```

```
transactions (#PCDATA)> <!ELEMENT use-single-connection (#PCDATA)>
<!ELEMENT value (#PCDATA)>
```

協力廠商 JDBC 驅動程式描述元輸入 DTD

G

本節包含協力廠商 JDBC 描述元輸入檔案的文件類型定義 (DTD)。

```
<?xml version="1.0" encoding="UTF-8"?> <!ELEMENT actions (exec-sql |
check-for-closed-connection | fetch-metadata | rollback)*> <!ELEMENT
add-default-values-on-view-insert (#PCDATA)> <!ELEMENT authentication
(sql-state | error-code | sql-state-class | error-code-range |
actions)*> <!ELEMENT check-for-closed-connection EMPTY> <!ELEMENT
column-position-comparator (#PCDATA)> <!ELEMENT connection-properties
(property*)> <!ELEMENT connectivity (sql-state | error-code | sql-
state-class | error-code-range | actions)*> <!ELEMENT current-
timestamp-stmt (#PCDATA)> <!ELEMENT error-code (value)> <!ATTLIST
error-code description CDATA #IMPLIED > <!ELEMENT error-code-range
(from, to)> <!ATTLIST error-code-range description CDATA #IMPLIED >
<!ELEMENT errors (connectivity | authentication | retry | fatal)*>
<!ELEMENT exclude-table-filter (#PCDATA)> <!ELEMENT exec-sql
(#PCDATA)> <!ELEMENT fatal (sql-state | error-code | sql-state-class |
error-code-range | actions)*> <!ELEMENT fetch-metadata EMPTY>
<!ELEMENT from (#PCDATA)> <!ELEMENT function-return-method (#PCDATA)>
<!ELEMENT handle-stmt-results (#PCDATA)> <!ELEMENT include-table-
filter (#PCDATA)> <!ELEMENT jdbc-driver (metadata-override |
connection-properties | sql-type-map | options | errors)*> <!ELEMENT
key (#PCDATA)> <!ELEMENT left-outer-join-operator (#PCDATA)> <!ELEMENT
lock-generator-class (#PCDATA)> <!ELEMENT metadata-override (supports-
schemas-in-procedure-calls?)> <!ELEMENT minimal-metadata (#PCDATA)>
<!ELEMENT options (lock-generator-class | supports-schemas-in-
metadata-retrieval | time-translator-class | column-position-
comparator | use-manual-transactions | minimal-metadata | transaction-
isolation-level | use-single-connection | exclude-table-filter |
include-table-filter | left-outer-join-operator | current-timestamp-
stmt | add-default-values-on-view-insert | reuse-statements |
function-return-method | handle-stmt-results)*> <!ELEMENT property
(key, value)> <!ELEMENT retry (sql-state | error-code | sql-state-
class | error-code-range | actions)*> <!ELEMENT reuse-statements
(#PCDATA)> <!ELEMENT rollback EMPTY> <!ELEMENT sql-state (value)>
<!ATTLIST sql-state description CDATA #IMPLIED > <!ELEMENT sql-state-
class (value)> <!ATTLIST sql-state-class description CDATA #IMPLIED >
<!ELEMENT sql-type-map (type*)> <!ELEMENT supports-schemas-in-
metadata-retrieval (#PCDATA)> <!ELEMENT supports-schemas-in-procedure-
calls (#PCDATA)> <!ELEMENT time-translator-class (#PCDATA)> <!ELEMENT
to (#PCDATA)> <!ELEMENT transaction-isolation-level (#PCDATA)>
<!ELEMENT type (from, to)> <!ELEMENT use-manual-transactions
(#PCDATA)> <!ELEMENT use-single-connection (#PCDATA)> <!ELEMENT value
(#PCDATA)>
```

資料庫描述元 DTD

H

本節包含資料庫描述元檔案的文件類型定義 (DTD)。

```
<?xml version="1.0" encoding="UTF-8"?> <!ELEMENT add-default-values-
on-view-insert (#PCDATA)> <!ELEMENT column-position-comparator
(#PCDATA)> <!ELEMENT current-timestamp-stmt (#PCDATA)> <!ELEMENT
database (imports?, identity, options?)> <!ELEMENT exclude-table-
filter (#PCDATA)> <!ELEMENT function-return-method (#PCDATA)>
<!ELEMENT handle-stmt-results (#PCDATA)> <!ELEMENT include-table-
filter (#PCDATA)> <!ELEMENT identity (name?, regex-name?, regex-
version?)> <!ELEMENT import (#PCDATA)> <!ELEMENT imports (import*)>
<!ELEMENT left-outer-join-operator (#PCDATA)> <!ELEMENT lock-
generator-class (#PCDATA)> <!ELEMENT minimal-metadata (#PCDATA)>
<!ELEMENT name (#PCDATA)> <!ELEMENT options (lock-generator-class |
supports-schemas-in-metadata-retrieval | time-translator-class |
column-position-comparator | use-manual-transactions | minimal-
metadata | transaction-isolation-level | use-single-connection |
exclude-table-filter | include-table-filter | left-outer-join-operator
| current-timestamp-stmt | add-default-values-on-view-insert | reuse-
statements | function-return-method | handle-stmt-results)*> <!ELEMENT
regex-name (#PCDATA)> <!ELEMENT regex-version (#PCDATA)> <!ELEMENT
reuse-statements (#PCDATA)> <!ELEMENT supports-schemas-in-metadata-
retrieval (#PCDATA)> <!ELEMENT time-translator-class (#PCDATA)>
<!ELEMENT transaction-isolation-level (#PCDATA)> <!ELEMENT use-manual-
transactions (#PCDATA)> <!ELEMENT use-single-connection (#PCDATA)>
```

資料庫描述元輸入 DTD

本節包含資料庫描述元輸入檔案的文件類型定義 (DTD)。

```
<?xml version="1.0" encoding="UTF-8"?> <!ELEMENT add-default-values-
on-view-insert (#PCDATA)> <!ELEMENT column-position-comparator
(#PCDATA)> <!ELEMENT current-timestamp-stmt (#PCDATA)> <!ELEMENT
exclude-table-filter (#PCDATA)> <!ELEMENT function-return-method
(#PCDATA)> <!ELEMENT handle-stmt-results (#PCDATA)> <!ELEMENT include-
table-filter (#PCDATA)> <!ELEMENT database (options?)> <!ELEMENT left-
outer-join-operator (#PCDATA)> <!ELEMENT lock-generator-class
(#PCDATA)> <!ELEMENT minimal-metadata (#PCDATA)> <!ELEMENT options
(lock-generator-class | supports-schemas-in-metadata-retrieval | time-
translator-class | column-position-comparator | use-manual-
transactions | minimal-metadata | transaction-isolation-level | use-
single-connection | exclude-table-filter | include-table-filter |
left-outer-join-operator | current-timestamp-stmt | add-default-
values-on-view-insert | reuse-statements | function-return-method |
handle-stmt-results)*> <!ELEMENT reuse-statements (#PCDATA)> <!ELEMENT
supports-schemas-in-metadata-retrieval (#PCDATA)> <!ELEMENT time-
translator-class (#PCDATA)> <!ELEMENT transaction-isolation-level
(#PCDATA)> <!ELEMENT use-manual-transactions (#PCDATA)> <!ELEMENT use-
single-connection (#PCDATA)>
```

規則範例：無觸發未來事件處理

J

下列範例假設 "commence" 屬性存在，並進行下列幾項操作：

- 保留事件應處理之時間的時戳值
- 包含整數或 Java 字串時戳值。請參閱 「時間語法」，第 42 頁。

```
<policy  xmlns:Timestamp="http://www.novell.com/nxsl/java/
java.sql.Timestamp" xmlns:TimestampUtil="http://www.novell.com/nxsl/
java/com.novell.nds.dirxml.driver.jdbc.db.TimestampUtil"
xmlns:jdbc="urn:dirxml:jdbc"> <rule> <description>Get commencement
date from datasource.</description> <conditions> <and> <if-xpath
op="true">.</if-xpath> </and> </conditions> <actions> <do-set-local-
variable name="commence"> <arg-string> <token-src-attr class-
name="User" name="commence"/> </arg-string> </do-set-local-variable>
</actions> </rule>


<rule> <description>Break if commencement date unavailable.</
description> <conditions> <and> <if-local-variable name="commence"
op="equal"/> </and> </conditions> <actions> <do-break/> </actions> </
rule>


<rule> <description>Parse times.</description> <conditions> <and> <if-
xpath op="true">.</if-xpath> </and> </conditions> <actions> <do-set-
local-variable name="dbTime"> <arg-object> <token-xpath
expression="Timestamp:valueOf(@jdbc:database-local-time)"/> </arg-
object> </do-set-local-variable> <do-set-local-variable
name="eventTime"> <arg-object> <token-xpath
expression="Timestamp:valueOf($commence)"/> </arg-object> </do-set-
local-variable> </actions> </rule><rule> <description>Is commencement
date after database time?</description> <conditions> <and> <if-xpath
op="true">.</if-xpath> </and> </conditions> <actions> <do-set-local-
variable name="after"> <arg-string> <token-xpath
expression="TimestampUtil:after($eventTime, $dbTime)"/> </arg-string>
</do-set-local-variable> </actions> </rule>


<rule> <description>Retry if future event.</description> <conditions>
<and> <if-local-variable name="after" op="equal">true</if-local-
variable> </and> </conditions> <actions> <do-status level="retry">
<arg-string> <token-text xml:space="preserve">Future event detected.</
token-text> </arg-string> </do-status> </actions> </rule></policy>
```

文件更新

K

本節包含 Identity Manager Driver for JDBC 的新資訊或更新資訊。

該文件在 Web 上以兩種格式提供：HTML 和 PDF。HTML 和 PDF 文件都會與本節所列出的文件變更保持同步。

如果您需要瞭解所使用之 PDF 文件的副本是否爲最新版本，請檢查該 PDF 檔的發行日期。日期位於標題頁。

新文件或更新文件的發行日期如下所示：

K.1 2005 年 12 月 14 日

表格 ***K-1*** *2005 年 12 月 14 日的更新*

位置	變更
「驅動程式參數」，第 40 頁	在資訊階層中將區段向上移動一層，您便可以輕鬆地尋找驅動程式參數。
表格 5-10 頁上 88	更新事件類型表格。
「事件類型」，第 90 頁	更新表格 5-12 和說明表格的文字。
「Oracle Thin Client JDBC 驅動程式」，第 118 頁	更新「已知問題」一節。

K.2 2006 年 4 月 24 日

表格 ***K-2*** *2006 年 4 月 19 日的更新*

位置	變更
「訂閱者通道」，第 16 頁	更正圖形，以顯示「檢視窗」而不是「中介表格」。

K.3 2006 年 5 月 1 日

表格 ***K-3*** *2006 年 5 月 1 日的更新*

位置	變更
「IBM Toolbox for Java/JTOpen」，第 121 頁	新增此主題。

K.4 2006 年 5 月 12 日

表格 ***K-4*** *2006 年 5 月 12 日的更新*

位置	變更
「IBM Toolbox for Java/JTOpen」，第 121 頁	新增一段關於已輸入範例組態檔中設定值的說明性段落。您需要手動輸入值。它們不是自動新增的。