# Sun Java System Application Server 9.1 管理指南

Sun Microsystems, Inc.
4150 Network Circle
Santa Clara, CA 95054
U.S.A.

文件號碼：820–4607–11
2007 年 12 月

# 目錄

# 圖清單

# 表清單

# 範例清單

# 前言

「Sun Java System Application Server 9.1 管理指南」說明如何從管理主控台配置、管理和部署 Application Server 子系統與元件。

此前言說明整個 Sun Java™ 系統 Application Server 文件集的相關資訊與慣例。

## Application Server 文件集

Application Server 文件集描述部署規劃與系統安裝。Application Server 文件的統一資源定址器 (URL) 爲 `http://docs.sun.com/coll/1343.4` 和 `http://docs.sun.com/coll/1777.1`。如需有關 Application Server 的簡介，請依循下表排列順序參閱其中列出的書籍。

**表 P–1** Application Server 文件集中的書籍

| 書籍標題 | 說明 |
|---|---|
| 文件中心 | 依作業及主旨編排的 Application Server 文件主題。 |
| 版本說明 | 軟體與文件的最新資訊。包含支援硬體、作業系統、Java 開發工具組 (JDK™)，以及資料庫驅動程式的完整表格形式摘要。 |
| 快速入門指南 | 如何開始使用 Application Server 產品。 |
| 安裝指南 | 安裝 軟體及其元件。 |
| 部署規劃指南 | 評估系統需求和企業狀況，確保以最適合站點的方式部署 Application Server。此外還說明了部署伺服器時應該注意的常見問題。 |
| 應用程式部署指南 | 將應用程式及應用程式元件部署到 Application Server。包含有關部署描述元的資訊。 |
| 開發者指南 | 建立與實作您要在 Application Server (遵循適用於 Java EE 元件及 API 的開放式 Java 標準模型) 上執行的 Java Platform, Enterprise Edition (Java EE 平台) 應用程式。其中包括有關開發者工具、安全性、除錯和建立生命週期模組的資訊。 |
| Java EE 5 教學課程 | 使用 Java EE 5 平台技術及 API 來開發 Java EE 應用程式。 |
| Java WSIT 教學課程 | 使用 Web 服務互通技術 (WSIT) 開發 Web 應用程式。說明如何、何時及爲何要使用 WSIT 技術，以及每一項技術所支援的功能和選項。 |

表 P–1 Application Server 文件集中的書籍 *(續)*

| 書籍標題 | 說明 |
|---|---|
| 管理指南 | Application Server 的系統管理，包括配置、監視、安全性、資源管理，以及 Web 服務管理。 |
| 高可用性管理指南 | 高可用性資料庫安裝後的配置和管理說明。 |
| 管理參考資料 | 編輯 Application Server 配置檔案 `domain.xml`。 |
| 升級與遷移指南 | 從舊版的 Application Server 升級，或從其他廠商的 Application Server 遷移 Java EE 應用程式。本指南也說明相近的產品發行版本和配置選項之間的哪些差異可能會導致產品規格不相容。 |
| 效能調校指南 | 調校 Application Server 以提昇效能。 |
| 疑難排解指南 | 解決 Application Server 問題。 |
| 錯誤訊息參照 | 解決 Application Server 錯誤訊息。 |
| 參考手冊 | 與 Application Server 一起提供的公用程式指令；以線上手冊樣式編寫。其中包含 `asadmin` 指令行介面。 |

# 相關文件

Application Server 可單獨購買或隨附於 Sun Java Enterprise System (Java ES) 購買。後者是一種軟體基礎架構，支援分散在網路或網際網路環境中的企業應用程式。如果您將 Application Server 做爲 Java ES 的元件購買，則應熟讀位於 `http://docs.sun.com/coll/1286.3` 和 `http://docs.sun.com/coll/1412.3` 的系統文件。Java ES 及其元件的所有相關文件 URL 爲 `http://docs.sun.com/prod/entsys.5`。

如需有關其他獨立 Sun Java System 伺服器產品的文件資訊，請移至下列位置：

- (`http://docs.sun.com/coll/1343.4`) 和 Message Queue 文件 (`http://docs.sun.com/coll/1777.1`)
- (`http://docs.sun.com/coll/1224.1`) 和 Directory Server 文件 (`http://docs.sun.com/coll/1632.1`)
- (`http://docs.sun.com/coll/1308.3`) 和 Web Server 文件 (`http://docs.sun.com/coll/1425.2`)

Application Server 隨附套件的 Javadoc™ 工具參考資料位於 `http://glassfish.dev.java.net/nonav/javaee5/api/index.html`。此外，以下資源可能會有用：

- Java EE 5 規格 (`http://java.sun.com/javaee/5/javatech.html`)
- Java EE Blueprints (`http://java.sun.com/reference/blueprints/index.html`)

如需有關在「NetBeans™ 整合開發環境 (IDE)」中建立企業應用程式的資訊，請參閱 `http://www.netbeans.org/kb/55/index.html`。

如需有關 Application Server 隨附的 Java DB 資料庫的資訊，請參閱 `http://developers.sun.com/javadb/`。

GlassFish Samples 專案是應用程式範例的集合，展示了多項 Java EE 技術。Java EE 軟體開發套件 (SDK) 隨附 GlassFish Samples，您也可以從「GlassFish Samples」專案頁面 (網址爲 `https://glassfish-samples.dev.java.net/`) 取得。

# 預設路徑和檔案名稱

下表說明在本書中使用的預設路徑和檔案名稱。

**表 P–2** 預設路徑和檔案名稱

| 預留位置 | 說明 | 預設值 |
|---|---|---|
| *as-install* | 表示 Application Server 的基底安裝目錄。 | Solaris™ 作業系統的 Java ES 安裝：<br>`/opt/SUNWappserver/appserver`<br>Linux 作業系統的 Java ES 安裝：<br>`/opt/sun/appserver/`<br>其他 Solaris 和 Linux 的安裝 (非超級使用者)：<br>*user's-home-directory*`/SUNWappserver`<br>其他 Solaris 和 Linux 的安裝 (超級使用者)：<br>`/opt/SUNWappserver`<br>Windows，所有安裝：<br>*SystemDrive*`:\Sun\AppServer` |
| *domain-root-dir* | 表示包含所有網域的目錄。 | Java ES Solaris 安裝：<br>`/var/opt/SUNWappserver/domains/`<br>Java ES Linux 安裝：<br>`/var/opt/sun/appserver/domains/`<br>所有其他安裝：<br>*as-install*`/domains/` |
| *domain-dir* | 表示網域的目錄。<br>在配置檔案中，您可能會看到 *domain-dir* 顯示如下：<br>`${com.sun.aas.instanceRoot}` | *domain-root-dir*`/`*domain-dir* |

表 P–2 預設路徑和檔案名稱 *(續)*

| 預留位置 | 說明 | 預設值 |
|---|---|---|
| *instance-dir* | 表示伺服器實例的目錄。 | *domain-dir*/*instance-dir* |

# 印刷排版慣例

下表描述本書在印刷排版上所做的變更。

表 P–3 印刷排版慣例

| 字體 | 含義 | 範例 |
|---|---|---|
| AaBbCc123 | 指令名稱、檔案名稱和目錄名稱以及螢幕畫面輸出 | 編輯 .login 檔案。<br>使用 ls -a 列示所有檔案。<br>machine_name% you have mail. |
| **AaBbCc123** | 您所鍵入的資訊，與螢幕畫面輸出相對應。 | machine_name% **su**<br>Password: |
| *AaBbCc123* | 將用實際的名稱或數值取代的預留位置 | 用來移除檔案的指令爲 rm *filename*。 |
| *AaBbCc123* | 書名、新的字彙或術語以及要強調的詞 (請注意，部分強調的項目在線上會以粗體顯示) | 請參閱使用者指南中的第 6 章。<br>**快取**是儲存在本機的副本。<br>請**勿**儲存檔案。 |

# 符號慣例

下表說明本書可能會使用的符號。

表 P–4 符號慣例

| 符號 | 說明 | 範例 | 含義 |
|---|---|---|---|
| [ ] | 包含選擇性引數和指令選項。 | ls [-l] | 無需 -l 選項。 |
| { \| } | 包含爲所需指令選項提供的一組選擇。 | -d {y\|n} | -d 選項要求您使用 y 引數或 n 引數。 |
| ${ } | 表示變數參照。 | ${com.sun.javaRoot} | 參照 com.sun.javaRoot 變數的值。 |

表 P–4 符號慣例 (續)

| 符號 | 說明 | 範例 | 含義 |
|---|---|---|---|
| - | 連接需同時按下的多個按鍵。 | Control-A | 同時按 Control 鍵和 A 鍵。 |
| + | 連接需連續按下的多個按鍵。 | Ctrl+A+N | 按 Control 按鍵，然後鬆隋A依次按後面的按鍵。 |
| → | 表示圖形化使用者介面中的功能表項目選取。 | [檔案] → [新建] → [範本] | 從 [檔案] 功能表中，選擇 [新建]。從 [新建] 子功能表中，選擇 [範本]。 |

# 文件、支援和訓練

Sun 網站提供以下其他資源的相關資訊：

- 文件 (http://www.sun.com/documentation/)
- 支援 (http://www.sun.com/support/)
- 訓練 (http://www.sun.com/training/)

# 搜尋 Sun 產品文件

除了從 docs.sun.com℠ 網站搜尋 Sun 產品文件之外，您也可以藉由在搜尋欄位中鍵入下列語法來使用搜尋引擎：

*search-term* `site:docs.sun.com`

例如，若要搜尋「broker」，請鍵入：

```
broker site:docs.sun.com
```

若要在搜尋中加入其他 Sun 網站 (例如，java.sun.com、www.sun.com 和 developers.sun.com)，請在搜尋欄位中，使用 `sun.com` 取代 `docs.sun.com`。

# 協力廠商網站參考

本文件中提供了協力廠商 URL 以供參考，另亦提供其他相關的資訊。

**備註 –** Sun 對本文件中提到的協力廠商網站的可用性不承擔任何責任。對於此類網站或資源中的 (或透過它們所取得的) 任何內容、廣告、產品或其他材料，Sun 並不表示認可，也不承擔任何責任。對於因使用或依靠此類網站或資源中的 (或透過它們所取得的) 任何內容、產品或服務而造成的、名義上造成的或連帶產生的任何實際或名義上之損壞或損失，Sun 概不負責，也不承擔任何責任。

# Sun 歡迎您提出寶貴意見

Sun 致力於提高文件品質，因此誠心歡迎您提出意見與建議。若要分享您的意見，請至 http://docs.sun.com，然後按一下 [Send Comments (傳送意見)]。線上表單提供了完整的文件標題和文件號碼。文件號碼是一個七位或九位的數字，可以在書的標題頁面或文件的 URL 中找到。例如，本書的文件號碼爲 820-4607 。

◆ ◆ ◆ 第 1 章

# 1 入門

Sun Java™ System Application Server 管理包含多項作業，例如部署應用程式、建立並配置網域、伺服器實例和資源；控制 (啓動和停止) 網域和伺服器實例、管理設定檔和叢集、監視並管理效能，以及診斷問題並進行疑難排解。它包含以下小節：



## Application Server 概觀

Sun Java System Application Server 提供了與 Java EE 相容的伺服器，可用來開發和部署 Java EE 應用程式和 Java Web 服務。主要功能包括可縮放式作業事件管理、容器管理式的持續性執行階段、高效能 Web 服務、叢集、高可用性、安全性以及整合功能。本小節包含下列主題：



### 用法設定檔

每個管理網域都與一個用法設定檔相關聯，可識別該網域的功能。Application Server 提供下列設定檔：

- **開發者**：若是在開發環境中執行網域，且應用程式不需要 NSS 金鑰庫或叢集功能 (如負載平衡及階段作業持續性)，請使用此設定檔。
- **叢集**：如果需要建立叢集，但不需要高可用性資料庫 (HADB) 或 NSS 金鑰庫，請使用此設定檔。

- **企業**：如果需要 HADB 和 NSS，請使用此設定檔。僅當分開安裝 HADB 和 NSS，或將 Application Server 安裝成 Java Enterprise System (JES) 的元件時，才能使用此設定檔。如需有關如何將企業設定檔與 Application Server 9.1 搭配使用的資訊，請參閱第 28 頁的「使用企業設定檔」

---

**備註 –** 只有企業設定檔支援從 Application Server 8.x Enterprise Edition 升級。如果是從 Application Server 8.x Platform Edition 升級，請使用開發者設定檔。如需有關升級程序的更多資訊，請參閱「Sun Java System Application Server 9.1 Update 1 Upgrade and Migration Guide」中的第 2 章「Upgrading an Application Server Installation」。

---

網域爲使用者應用程式提供預先配置的執行階段。用法設定檔有助於區分 Application Server 二進位碼與執行階段配置。設定檔讓您使用相同的 Application Server 安裝，即可藉由適合不同用途的設定檔建立不同網域。例如，開發者可能希望使用 Application Server 來瞭解最新的 Java EE 規格。這個開發者不需要嚴格的安全性設定。但另一個想在生產環境中部署應用程式的使用者則需要原本就安全的環境。

表 1–1 列出每個設定檔的可用功能：

**表 1–1** 每個設定檔的可用功能

| 功能 | 開發者設定檔 | 叢集設定檔 | 企業設定檔 |
|---|---|---|---|
| 安全性存放區 | JKS | JKS | NSS |
| 叢集/獨立實例 | 不可用 | 可用 | 可用 |
| 安全性管理員 | 已停用 | 已啓用 | 已啓用 |
| HADB | 不可用 | 不可用 | 可用 |
| 負載平衡 | 不可用 | 可用 | 可用 |
| 節點代理程式 | 不可用 | 可用 | 可用 |

## 使用企業設定檔

若要使用企業設定檔，請執行下列作業：

1. 分別下載並安裝 NSS 與 HADB。
2. 如下所示，修改 `asenv.conf` 檔案：
   - `AS_HADB` 指向 HADB 的安裝資料夾。
   - `AS_NSS` 指向提供 NSS 共用物件的資料夾。
   - `AS_NSS_BIN` 指向儲存 NSS 二進位碼 (例如 `certutil`) 的資料夾。

### 將舊版網域升級到 Application Server 9.1

您可以使用 `start-domain` 指令，將 Application Server 8.x 或 9.0 網域升級到 Application Server 9.1。請使用下列其中一種方式來升級網域：

- 執行 Application Server 二進位碼的就地升級。

  當您在指向舊版 Application Server 的網域上執行 `start-domain` 時，`asadmin` 將呼叫 `asupgrade` 指令，並且會自動就地升級網域。
- 執行 Application Server 二進位碼的同時升級。

  在先前安裝的網域上執行 `start-domain`。`asupgrade` 指令可將網域升級到最新 Application Server 安裝的網域根目錄。在此案例中，會在 `asenv.conf` 的 `AS_DEF_DOMAINS_PATH` 中定義升級的目標目錄。

## 什麼是 Application Server？

Application Server 是一個平台，支援的服務包含從 Web 發佈到企業範圍內的作業事件處理，同時可讓開發人員基於 JavaServer Pages (JSP™)、Java servlet 以及 Enterprise JavaBeans™ (EJB™) 技術建置應用程式。

Application Server 9.1 叢集和企業設定檔提供進階的叢集和容錯移轉技術。這些功能讓您執行可延伸且具有高可用性的 Java EE 應用程式。

- **叢集** - 叢集是一組 Application Server 實例，這些實例以單一邏輯實體的形式一起運作。叢集中的每個 Application Server 實例均部署有相同配置和相同的應用程式。

  透過將 Application Server 實例增加至叢集來實現水平比例縮放，從而增加系統容量。可以在不中斷服務的情況下將 Application Server 實例增加至叢集。HTTP、RMI/IIOP 和 JMS 負載平衡系統會將請求分散到叢集中運作狀態良好的 Application Server 實例中。
- **高可用性** - 可用性允許對叢集中的 Application Server 實例進行容錯移轉保護。如果一個 Application Server 實例出現故障，則其他 Application Server 實例將接管指定給該故障伺服器的階段作業。階段作業資訊儲存在高可用性資料庫 (HADB) 中。HADB 支援 HTTP 階段作業和有狀態階段作業 Bean 的持續性。

## Application Server 架構

本小節說明圖 1–1，該圖顯示了 Application Server 的整體架構。

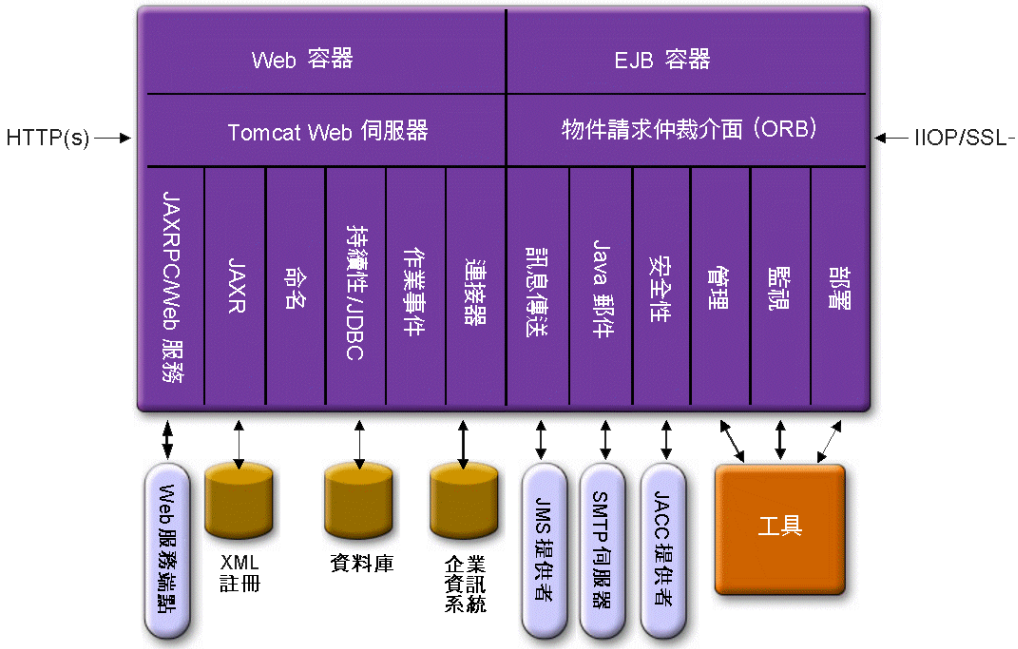


圖 1–1 Application Server 架構

- **容器** - 容器是一種執行階段環境，它爲 Java EE 元件提供安全性和作業事件管理等服務。圖 1–1 顯示了兩種類型的 Java EE 容器：Web 和 EJB。Web 元件 (例如 JSP 頁面和 Servlet) 在 Web 容器內執行。企業 Bean (EJB 技術的元件) 在 EJB 容器內執行。
- **用戶端存取** - 在執行階段，瀏覽器用戶端透過 HTTP (在網際網路中使用的協定) 與 Web 伺服器進行通訊來存取 Web 應用程式。HTTPS 協定用於需要安全通訊的應用程式。企業 Bean 用戶端透過 IIOP 協定或 IIOP/SSL (安全) 協定與物件請求代理程式 (ORB) 通訊。Application Server 具有分別用於 HTTP 通訊協定、HTTPS 通訊協定、IIOP 協定和 IIOP/SSL 協定的偵聽程式。每個偵聽程式專用特定的連接埠號碼。
- **Web 服務** - 您可在 Java EE 平台上部署 Web 應用程式，以提供 Java API for XML-Based RPC (JAX-RPC) 實作的 Web 服務。Java EE 應用程式或元件還可以是其他 Web 服務的用戶端。應用程式通過用於 XML 登錄的 Java API (JAXR) 存取 XML 登錄。
- **用於應用程式的服務** - Java EE 平台旨在使容器爲應用程式提供服務。圖 1–1 顯示了以下服務：
  - 命名 - 命名和目錄服務可將物件連結到名稱。Java EE 應用程式透過查找物件的 JNDI 名稱來找到物件。JNDI 代表 Java 命名和目錄介面 API。
  - 安全性 - Java 容器授權合約 (JACC) 是一組爲 Java EE 容器定義的安全性合約。依照用戶端的身份，容器可限制對容器資源和服務的存取。
- **作業事件管理** - 作業事件是不可分割的工作單元。例如，在銀行帳戶之間轉帳是一個作業事件。作業事件管理服務用於確定完全完成作業事件或將作業事件轉返。

## 存取外部系統

Java EE 平台使應用程式能夠存取 Application Server 之外的系統。應用程式通過稱爲資源的物件連線到這些系統。管理員的職責之一是資源配置。Java EE 平台可以透過以下 API 和元件存取外部系統：

- **JDBC** - 資料庫管理系統 (DBMS) 提供了用於儲存、組織和擷取資料的功能。大多數企業應用程式將資料儲存在關聯式資料庫中，這些應用程式透過 JDBC API 存取關聯式資料庫。由於資料庫中的資訊儲存在磁碟上並在應用程式結束之後仍然存在，因此通常將資料庫中的資訊稱爲持續性資訊。Application Server 隨附軟體包括 Java DB 資料庫。
- **訊息傳送** - 訊息傳送是軟體元件或應用程式之間的一種通訊方法。訊息傳送用戶端可以向任何其他用戶端傳送訊息，也可以從任何其他用戶端接收訊息。應用程式通過 Java 訊息傳送服務 (JMS) API 存取訊息傳送提供者。Application Server 包含 JMS 提供者。
- **連接器** - Java EE 連接器架構可整合 Java EE 應用程式和現有企業資訊系統 (EIS)。應用程式透過稱爲連接器或資源配接卡的可移植 Java EE 元件存取 EIS。
- **JavaMail** - 透過 JavaMail API，應用程式連線至 SMTP 伺服器以傳送和接收電子郵件。
- **伺服器管理** -圖 1-1 右下角顯示 Application Server 管理員可執行的一些作業。例如，管理員部署 (安裝) 應用程式並監視伺服器的效能。這些作業透過 Application Server 提供的管理工具來執行。

# 管理工具

Application Server 提供了下列管理工具和 API：


- 第 31 頁的 「管理主控台」
- 第 32 頁的 「指令行介面 (asadmin 公用程式)」
- 第 32 頁的 「JConsole」
- 第 33 頁的 「Application Server Management Extension (AMX)」


## 管理主控台

管理主控台是一種基於瀏覽器的工具，具有易於導覽的介面和線上說明。管理伺服器 (也稱爲 Domain Administration Server 或 DAS) 必須在執行狀態下，才能使用管理主控台。若要啓動管理主控台，必須知道管理伺服器主機名稱和連接埠號碼。您當初安裝 Application Server 時已選擇伺服器的連接埠號碼，或使用預設的連接埠 4848。您也已經指定使用者名稱和主密碼。

若要啓動管理主控台，請在 Web 瀏覽器中鍵入以下內容：

```
http://hostname:port
```

例如：

```
http://kindness.sun.com:4848
```

如果管理主控台在安裝了 Application Server 的機器上執行，請將 `localhost` 指定爲主機名稱。

在 Windows 上，從 [開始] 功能表啓動 Application Server 管理主控台。

安裝程式將建立使用預設連接埠號碼 4848 的預設管理網域 (名爲 `domain1`)，並會建立獨立於 Domain Administration Server (DAS) 的實例。安裝之後，還可以建立其他管理網域。每個網域都具有自己的網域管理伺服器，該伺服器具有唯一的連接埠號碼。爲管理主控台指定 URL 時，請務必使用要管理的網域的連接埠號碼。

如果配置中包括遠端伺服器實例，請建立節點代理程式以便管理和簡化遠端伺服器實例。節點代理程式負責建立、啓動、停止和刪除伺服器實例。使用指令行介面 (CLI) 指令可以設定節點代理程式。

## 指令行介面 (`asadmin` 公用程式)

`asadmin` 公用程式是 Sun Java System Application Server 的指令行介面。使用 `asadmin` 公用程式及其相關指令，可執行管理主控台所提供的相同管理作業集。Solaris 上的預設安裝根目錄爲 `/opt/SUNWappserver`。

若要啓動 `asadmin` 公用程式，請移至 *as-install*/bin 目錄並輸入：

```
$ ./asadmin
```

若要列出 `asadmin` 中的可用指令，請使用：

```
asadmin> help
```

也可以在 Shell 的指令提示符號下發出 `asadmin` 指令：

```
$ asadmin help
```

若要檢視指令的語法和範例，請鍵入 `help` 並在其後鍵入指令名稱 。例如：

```
asadmin> help create-jdbc-resource
```

所指定指令的 `asadmin help` 資訊可顯示此指令的 Unix 線上手冊。另外，在「Sun Java System Application Server 9.1 Reference Manual」 中還提供 HTML 和 PDF 格式的線上手冊。

## JConsole

Java 2, Platform Standard Edition 5.0 中引入了 Java Monitoring and Management Console (JConsole)。JConsole 用於監視 Sun Java System Application Server。您可以使用 JConsole [遠端] 標籤或 [進階] 標籤連線至 Application Server。

- [遠端] 標籤：識別使用者名稱、密碼、管理伺服器主機和 JMS 連接埠號碼 (預設爲 8686)，然後選取 [連線]。
- [進階] 標籤：將 `JMXServiceURL` 識別爲服務：`jmx:rmi:///jndi/rmi://host:jms-port/jmxrmi`，然後選取 [連線]。`JMXServerURL` 會列印在 `server.log` 檔案中，也會輸出在網域建立指令的指令視窗上。

### Application Server Management Extension (AMX)

Application Server **M**anagement e**X**tension 是一個 API，它可顯示所有 Application Server 配置，並可將 JMX 管理 Bean 做爲實作 AMX 介面的、易於使用的用戶端動態代理伺服器來進行監視。

如需使用 Application Server Management Extension 的更多資訊，請參閱「Sun Java System Application Server 9.1 Developer's Guide」中的第 20 章「Using the Application Server Management Extensions」。

# Application Server 指令和概念

Application Server 包含一或多個網域。網域是管理界限或環境。每個網域都有一部與之關聯的管理伺服器 (也稱爲 Domain Administration Server 或 DAS)，並由零個或多個獨立的實例和/或叢集所組成。每個叢集都有一個或多個同質的伺服器實例。伺服器實例是在單一實體機器上執行 Application Server 的單一 Java 虛擬機器 (JVM)。網域中的伺服器實例 (不論是獨立的或是叢集的) 都可在不同的實體主機上執行。

本小節包含下列主題：


- 第 33 頁的 「網域」
- 第 34 頁的 「Domain Administration Server (DAS)」
- 第 34 頁的 「叢集」
- 第 34 頁的 「節點代理程式」
- 第 34 頁的 「伺服器實例」
- 第 37 頁的 「Application Server 指令」


## 網域

網域是一起管理的實例群組。但是，一個 Application Server 實例只能屬於一個網域。除了充當管理界限外，網域還提供基本的安全性結構，不同的管理員可以藉此管理 Application Server 實例的特定群組 (網域)。透過將伺服器實例群組至單獨的網域中，不同的組織和管理員可以共用單一 Application Server 安裝。每個網域都有自己的獨立於其他網域的配置、記錄檔和應用程式部署區域。如果變更某個網域的配置，其他網域的配置不會受到影響。

Sun Java System Application Server 安裝程式會建立預設的管理網域 (名爲 domain1)。另外也會建立相關聯的網域管理伺服器 (名爲 server)。您必須提供管理伺服器連接埠號碼。預設的管理伺服器連接埠是 4848。安裝程式還會查詢管理使用者名稱和主密碼。安裝之後，還可以建立其他管理網域。

## Domain Administration Server (DAS)

每個網域都具有自己的 Domain Administration Server (DAS)，而該伺服器具有唯一的連接埠號碼。管理主控台會與特定的 DAS 進行通訊，以管理相關聯的網域。每個管理主控台階段作業都可讓您配置並管理特定的網域。

Domain Administration Server (DAS) 是特別指定的 Application Server 實例，負責主控管理應用程式。DAS 將認證管理員、接受來自管理工具的請求，並與網域中的伺服器實例進行通訊以執行請求。DAS 有時也稱爲管理伺服器或預設伺服器。由於它是唯一在 Sun Java System Application Server 安裝時所建立的伺服器實例而且可用於部署，因此被稱爲預設伺服器。DAS 即是具有附加管理功能的伺服器實例。

每個管理主控台階段作業都允許您配置並管理單一網域。若建立了多個網域，則必須啓動其他管理主控台階段作業以管理其他網域。爲管理主控台指定 URL 時，請務必使用與您要管理的網域相關聯的 DAS 連接埠號碼。

## 叢集

叢集是共用相同應用程式、資源和配置資訊集的已命名伺服器實例集合。一個伺服器實例只可以屬於一個叢集。叢集透過將負載分散於多部機器上，來實現伺服器實例的負載平衡。叢集透過實例層級的容錯移轉，以達到高可用性的目的。從管理的觀點來看，叢集代表虛擬化的實體，在此實體上，對叢集進行的作業 (如應用程式的部署) 將作用於組成叢集的所有實例上。

## 節點代理程式

網域中的每個節點都需要簡易代理程式 (例如僅託管 JMX 執行階段)，以簡化實例的遠端生命週期管理。其主要目的是依照 DAS 的指示，啓動、停止和建立伺服器實例。節點代理程式也扮演監視程式的角色，並重新啓動失敗的程序。節點代理程式和 DAS 一樣，應只有特定的管理作業才需要，且不需要具有高可用性。不過，節點代理程式爲「永遠開啓」的元件，而且必須加以配置，使其由原生 O/S 節點啓動程式啓動 (如 Solaris/Linux inetd，或做爲 Windows 服務)。節點代理程式對 DAS 並非必要。

## 伺服器實例

伺服器實例是與 Java EE 相容的單一 Java 虛擬機器，可在單一節點上主控 Application Server。每個伺服器實例在網域中都有唯一的名稱。叢集的伺服器實例爲叢集的成員之一，其所接收的應用程式、資源和配置都來自其父系叢集，以確保叢集中的所有實例

都是同質的。非叢集的伺服器實例則不屬於叢集，因此具有獨立的應用程式集、資源和配置。下圖詳細解釋 Application Server 實例。Application Server 實例是 Application Server 的叢集、負載平衡和階段作業持續性功能基本要素。

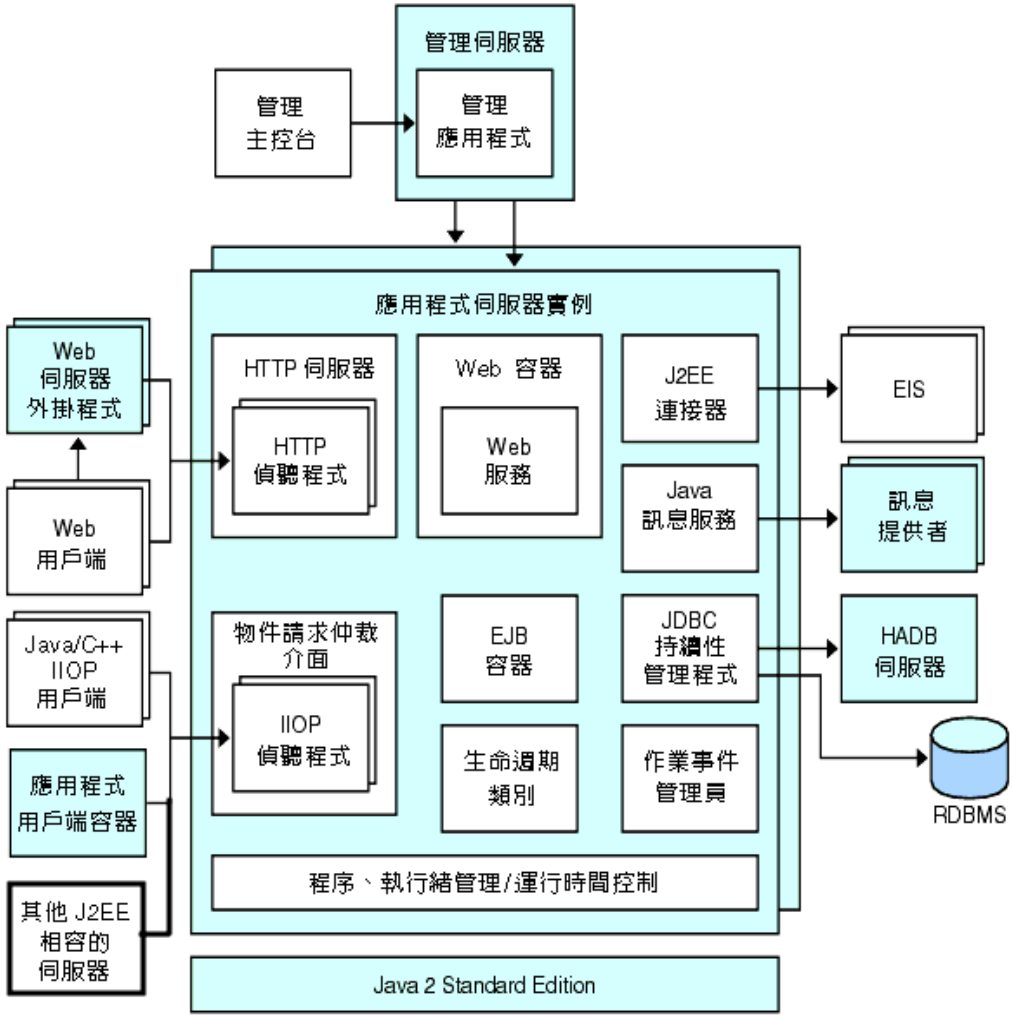


圖 1–2 Application Server 實例

Sun Java System Application Server 在安裝時會建立一個稱爲 `server` 的 Application Server 實例。對於許多使用者而言，一個 Application Server 實例就符合他們的需要了。不過，依據您的環境，您可能想建立一個或多個額外的 Application Server 實例。例如，在開發環境下，您可以使用不同的 Application Server 實例來測試不同的 Application Server 配

置，或比較和測試不同的應用程式部署。由於您可以輕易增加或刪除 Application Server 實例，因此您可以利用這些實例建立暫時的沙箱區域以進行試驗。

此外，您也可以針對每個 Application Server 實例建立虛擬伺服器。在單一安裝的 Application Server 實例內，您可以爲公司或個人提供網域名稱、IP 位址以及某些管理功能。對於使用者而言，看起來好像使用者有自己的 Web 伺服器，但沒有硬體和基本的伺服器維護功能。這些虛擬伺服器並不涵蓋 Application Server 實例。如需有關虛擬伺服器的更多資訊，請參閱第 13 章。

在作業部署中，您可以使用虛擬伺服器代替多個 Application Server 實例，用於多種目的。但是，如果虛擬伺服器不能滿足需求，您也可以使用多個 Application Server 實例。若您停止 Application Server 實例，該實例將停止接受新連線，然後等待所有未處理的連線完成。如果您的機器當機或離線，伺服器將結束，其正在處理的任何請求均可能遺失。

## 定義 Application Server 實例

Application Server 實例是應用程式部署的基礎。每個實例均屬於單一網域，並有自己的目錄結構、配置和已部署的應用程式。每個伺服器實例還包含了 Java EE 平台的 Web 和 EJB 容器。每個新的伺服器實例必須包含對節點代理程式名稱的參考，該名稱定義實例將要駐留的機器。

---

**備註 –** 您無法在開發者網域上建立 Application Server 實例。開發者網域只會自動與預設實例 (`server1`) 相關聯。若要建立多個實例，需要使用叢集設定檔建立網域。如需有關建立網域的資訊，請參閱指令 `create-domain` 的線上手冊，或參閱管理主控台線上說明。

---

您可以建立三類伺服器實例：

- 在**獨立伺服器**實例中，其他伺服器實例或叢集並不共用其配置。
- 在**共用伺服器實例**中，其他實例或叢集可以共用其配置。
- 在**叢集伺服器實例**中，叢集中的其他實例可以共用其配置。

  **叢集**是一組共用相同的應用程式集、資源集和配置資訊集的伺服器實例。伺服器實例可以只屬於一個叢集。叢集用於透過在多台機器上分散負載來增強負載平衡，並透過實例層級的容錯移轉來提供高可用性。

## 檢視一般伺服器資訊

透過 [一般] 標籤可以執行以下作業：

- 按一下 [啓動實例] 以啓動實例。
- 按一下 [停止實例] 以停止實例。
- 按一下 [檢視記錄檔]，以開啓伺服器記錄檢視器。

- 按一下 [自動重建記錄檔]，以自動重建實例的記錄檔。

  該動作將排程記錄檔以進行自動重建。實際的自動重建將在下一次向記錄檔寫入項目時發生。預設伺服器 (DAS) 的自動重建將立即發生，但其他獨立伺服器的自動重建將延遲。
- 按一下 [JNDI 瀏覽] 以瀏覽正在執行的實例的 JNDI 樹。
- 按一下 [恢復作業事件] 以恢復未完成的作業事件。

此外，您可以選取以下標籤以執行其他作業：

- [應用程式] 標籤：部署選取的應用程式。
- [JVM 設定] 標籤：配置 Application Server 所使用的 JVM 一般設定。
- [資源] 標籤：管理選取的資源。
- [特性] 標籤：配置實例特定的特性。
- [記錄] 標籤：配置 Application Server 所使用的記錄層級。
- [監視] 標籤：檢視 JVM、伺服器、執行緒池、HTTP 服務和作業事件服務的監視資料。
- [進階] 標籤：設定用於部署應用程式的一般特性。

---

**備註 –** 如果在開發者設定檔上執行管理主控台，則將無法使用 [啓動實例] 選項和 [應用程式] 和 [JVM 設定] 這類標籤。

---

# Application Server 指令

Application Server 的管理包含多項作業，如網域、叢集、節點代理程式和伺服器實例的建立、配置、控制和管理。本小節包含下列主題：

- 第 41 頁的 「重新建立網域管理伺服器」

## 建立網域

您必須使用 `create-domain` 指令建立網域。以下範例指令將建立名爲 `mydomain` 的網域。管理伺服器在連接埠 5000 上進行偵聽，管理使用者名稱爲 `admin`。該指令提示您輸入管理密碼和主密碼。

```
$ asadmin create-domain --adminport 5000 --adminuser admin mydomain
```

若要爲 `mydomain` 網域啓動管理主控台，請在瀏覽器中輸入以下 URL：

`http://`*hostname*`:5000`

在 Application Server 9.1 中，每個網域都具有一個相關聯的設定檔。如需有關設定檔的資訊，請參閱第 27 頁的 「用法設定檔」。您只能在建立期間選擇網域的設定檔。將 `--profile` 選項與 `create-domain` 指令搭配使用可指定網域的設定檔。若未使用 `--profile` 選項來明確指定設定檔，則預設的設定檔會與網域相關聯。`asadminenv.conf` 檔案中的 `AS_ADMIN_PROFILE` 變數，可定義預設的設定檔。

**注意 –** 除非您具有 HADB 和 Network Security Services (NSS) 金鑰庫，否則請勿建立企業網域。除非您具有 HADB 和 NSS，否則將無法啓動企業網域。

對於前面的 `create-domain` 範例，網域的記錄檔、配置檔案和部署的應用程式現在常駐於以下目錄中：

*domain-root-dir*`/mydomain`

若要在其他位置建立網域目錄，請指定 `--domaindir` 選項。如需指令的完整語法，請鍵入 `asadmin help create-domain` 或 `create-domain`(1)。

## 刪除網域

使用 `asadmin delete-domain` 指令可刪除網域。僅具有網域管理權限的作業系統使用者 (或 root 使用者) 才能成功地執行該指令。例如，若要刪除名爲 `mydomain` 的網域，請鍵入以下指令：

```
$ asadmin delete-domain mydomain
```

## 列出網域

使用 `asadmin list-domains` 指令可找到在機器中建立的網域。若要列出預設 *domain-root-dir* 目錄中的網域，請鍵入以下指令：

```
$ asadmin list-domains
```

若要列出在其他目錄中建立的網域，請指定 --domaindir 選項。

## 啓動網域

啓動網域時，將啓動管理伺服器和 Application Server 實例。啓動 Application Server 實例之後，Application Server 實例將持續執行、偵聽並接受請求。必須單獨啓動各個網域。

若要啓動網域，請鍵入 asadmin start-domain 指令並指定網域名稱。例如，若要啓動預設網域 (domain1)，請鍵入以下指令：

```
$ asadmin start-domain --user admin domain1
```

如果只有一個網域，則可以省略網域名稱。如需完整的指令語法，請鍵入 asadmin help start-domain。如果省略了密碼資料，系統將提示您提供此資料。

## 在 Windows 上啓動預設網域

在 Windows [開始] 功能表中，依次選取 [程式集] -> [Sun Microsystems] -> [Application Server] -> [啓動 Admin Server]。

## 停止網域

停止網域將關閉該網域的管理伺服器和 Application Server 實例。停止網域時，伺服器實例將停止接受新的連線，然後等待所有未完成的連線完成。由於伺服器實例必須完成其關閉程序，因此該程序需要幾秒鐘時間。停止網域時，管理主控台或大多數 asadmin 指令都無法使用。

若要停止網域，請鍵入 asadmin stop-domain 指令並指定網域名稱。例如，若要停止預設網域 (domain1)，請鍵入以下指令：

```
$ asadmin stop-domain domain1
```

如果只有一個網域，則網域名稱是選擇性的。如需完整語法，請鍵入 asadmin help stop-domain。

請參閱管理主控台線上說明，透過管理主控台停止網域。

## 在 Windows 上停止預設網域

在 [開始] 功能表中，依次選取 [程式集] -> [Sun Microsystems] -> [Application Server] -> [停止 Admin Server]。

## 重新啓動網域

重新啓動伺服器與重新啓動網域相同。若要重新啓動網域或伺服器，請停止然後再啓動網域。

## 建立叢集

叢集是使用 create-cluster 指令建立的。以下範例建立了一個名爲 mycluster 的叢集。管理伺服器主機是 myhost，伺服器連接埠是 1234，而管理使用者名稱是 admin。該指令提示您輸入管理密碼。

```
$ asadmin create-cluster --host myhost --port 1234 --user admin mycluster
```

如需完整語法，請鍵入 asadmin help create-cluster。

## 啓動叢集

叢集是使用 start-cluster 指令啓動的。以下範例啓動了名爲 mycluster 的叢集。該指令提示您輸入管理密碼。

```
$ asadmin start-cluster --host myhost --port 1234 --user admin mycluster
```

如需完整語法，請鍵入 asadmin help start-cluster。

## 停止叢集

叢集是使用 stop-cluster 指令停止的。以下範例停止了名爲 mycluster 的叢集。該指令提示您輸入管理密碼。

```
$ asadmin stop-cluster --host myhost --port 1234 --user admin mycluster
```

myhost 是管理伺服器主機，1234 是管理連接埠，admin 則是管理使用者名稱。

如需完整語法，請鍵入 asadmin help stop-cluster。停止某叢集後，即會停止該叢集中的所有伺服器實例。沒有伺服器實例的叢集無法啓動。

## 建立節點代理程式

節點代理程式是使用 create-node-agent 指令建立的。以下範例將建立一個名爲 mynodeagent 的節點代理程式。管理伺服器主機是 myhost，管理伺服器連接埠是 1234，而管理使用者名稱是 admin。此指令通常會提示您輸入管理密碼，但是，當未指定 --savemasterpassword 選項或此選項爲 false 時，指令就不會提示您輸入管理密碼。

```
$ asadmin create-node-agent --host myhost --port 1234 --user admin mynodeagent
```

如需完整語法，請鍵入 asadmin help create-node-agent。

## 啓動節點代理程式

節點代理程式是透過使用 start-node-agent 指令並指定節點代理程式名稱啓動的。例如，若要啓動節點代理程式 mynodeagent，請鍵入下列指令：

```
$ asadmin start-node-agent --user admin mynodeagent
```

如需完整語法，請鍵入 asadmin help start-node-agent。

## 停止節點代理程式

節點代理程式是透過使用 stop-node-agent 指令並指定節點代理程式名稱停止的。例如，若要停止節點代理程式 mynodeagent，請鍵入下列指令：

```
$ asadmin stop-node-agent mynodeagent
```

如需完整語法，請鍵入 asadmin help stop-node-agent。

## 啓動實例

伺服器實例是使用 start-instance 指令啓動的。以下範例啓動了名爲 myinstance 的伺服器實例。該指令提示您輸入管理密碼。

```
$ asadmin start-instance --host myhost --port 1234 --user admin myinstance
```

管理伺服器主機是 myhost，管理連接埠是 1234，而管理使用者名稱是 admin。伺服器實例 myinstance 可以加入叢集，也可以保持獨立。

如需完整語法，請鍵入 asadmin help start-instance。

## 停止實例

您必須使用 stop-instance 指令停止伺服器實例。以下範例停止了名爲 myinstance 的伺服器實例。該指令提示您輸入管理密碼。

```
$ asadmin stop-instance --host myhost --port 1234 --user admin myinstance
```

管理伺服器主機是 myhost，管理連接埠是 1234，而管理使用者名稱是 admin。伺服器實例 myinstance 可以加入叢集，也可以保持獨立。

如需完整語法，請鍵入 asadmin help stop-instance。

## 重新啓動實例

若要重新啓動伺服器實例，請先停止然後再啓動實例。

## 重新建立網域管理伺服器

若要進行鏡像並提供網域管理伺服器 (DAS) 的工作副本，您必須具有：

- 一台包含原始 DAS 的機器 (machine1)。
- 一台包含叢集的機器 (machine2)，該叢集具有執行應用程式並滿足用戶端需要的伺服器實例。該叢集是使用第一台機器上的 DAS 配置的。
- 一台備份機器 (machine3)，當第一台機器當機時，需要在該備份電腦上重新建立 DAS。

**備註 –** 必須保留一份第一台機器上的 DAS 的備份。使用 `asadmin backup-domain` 來備份目前網域。

## ▼ 遷移 DAS

以下步驟用於將 Domain Administration Server 從第一台機器 (machine1) 遷移到第三台機器 (machine3)：

1. **將 Application Server 安裝在第三台機器上，方法與在第一台機器上安裝時相同。**

   爲了可以在第三台機器上正確地復原 DAS 並且不會發生路徑衝突，您必須執行此操作。

   a. **使用指令行 (互動) 模式來安裝 Application Server 管理套裝軟體。若要啓動指令行互動式模式，請使用 `console` 選項呼叫安裝程式：**

      *./bundle-filename* `-console`

      若要使用指令行介面進行安裝，您必須具有 root 許可權。

   b. **若要安裝預設網域，請取消選取該選項。**

      只有具有相同架構且安裝路徑**完全**相同 (即兩台機器使用相同的 *as-install* 和 *domain-root-dir*) 的兩台機器，才支援復原備份的網域。

2. **將第一台機器上的備份 ZIP 檔案複製到第三台機器上的 *domain-root-dir* 目錄中。也可以透過 FTP 方式複製檔案。**

3. **執行 `asadmin restore-domain` 指令，以將 ZIP 檔案復原到第三台機器：**

   `asadmin restore-domain --filename` *domain-root-dir*`/sjsas_backup_v00001.zip domain1`

   可以備份任何網域。但是，在重新建立網域時，網域名稱應與原始網域名稱相同。

4. **變更第三台機器上的 *domain-root-dir*`/domain1/generated/tmp` 目錄的權限，以與第一台機器上相同目錄的權限相符。**

   該目錄的預設許可權爲：`?drwx------?` (或 700)。

   例如：

   `chmod 700` *domain-root-dir* `/domain1/generated/tmp`

   以上範例假定您備份的是 `domain1`。如果備份的是其他名稱的網域，則應使用要備份網域的名稱取代上述的 `domain1`。

5. **變更第三台機器的 `domain.xml` 檔案中的主機特性值：**

6 **更新第三台機器上的** *domain-root-dir*`/domain1/config/domain.xml`。

例如，搜尋 `machine1` 並將其替代爲 `machine3`。這樣，您就可以將：

```
<jmx-connector><property name=client-hostname value=machine1/>...
```

變更爲：

```
<jmx-connector><property name=client-hostname value=machine3/>...
```

7 **將：**

```
<jms-service... host=machine1.../>
```

變更爲：

```
<jms-service... host=machine3.../>
```

8 **在 machine3 上啓動復原的網域：**

```
asadmin start-domain --user admin-user --password admin-password domain1
```

9 **在 machine2 上變更節點代理程式下的 DAS 主機特性值。**

10 **在 machine2 上變更** *as-install*`/nodeagents/`*nodeagent*`/agent/config/das.properties` **中的** `agent.das.host` **特性值。**

11 **在 machine2 上重新啓動節點代理程式。**

---

**備註 –** 使用 `asadmin start-instance` 指令啓動叢集實例，以使這些實例與復原網域同步。

---

# Application Server 配置

Sun Java System Application Server 的配置儲存在 `domain.xml` 檔案中。`domain.xml` 文件可代表 Application Server 的配置狀態。它是指定管理網域的中央儲存庫。此文件包含 Application Server 網域模型的 XML 表示。`domain.xml` 的內容是由以網域 DTD 形式表示的規格規定的。

本小節包含下列主題：


- 第 44 頁的 「變更 Application Server 配置」
- 第 44 頁的 「Application Server 中的連接埠」

# 變更 Application Server 配置

在進行以下任何配置變更時，請重新啓動伺服器以使變更生效：

- 變更 JVM 選項
- 變更連接埠號碼
- 管理 HTTP 服務、IIOP 服務和 JMS 服務
- 管理執行緒池
- 修改下列 JDBC 連線池特性：
  - datasource-classname
  - JDBC 驅動程式特定供應商特性
  - associate-with-thread
  - lazy-connection-association
  - lazy-connection-enlistment
- 修改下列連接器連線池特性：
  - resource-adapter-name
  - connection-definition-name
  - transaction-support
  - 特定供應商特性
  - associate-with-thread
  - lazy-connection-association
  - lazy-connection-enlistment

如需說明，請參閱第 39 頁的 「重新啓動網域」 。

如果使用動態配置，大多數變更在伺服器執行時即可生效。若要進行下列配置變更，請**不要**重新啓動伺服器：

- 部署和取消部署應用程式
- 新增或移除 JDBC、JMS 與連接器資源和池
- 變更記錄層級
- 新增檔案範圍使用者
- 變更監視層級
- 啓用和停用資源和應用程式

請注意，`asadmin reconfig` 指令已停用，並且不再需要此指令。配置變更將動態套用至伺服器。

# Application Server 中的連接埠

下表說明 Application Server 的連接埠偵聽程式。

表 1–2 使用連接埠的 Application Server 偵聽程式

| 偵聽程式 | 預設連接埠號碼 | 說明 |
| --- | --- | --- |
| 管理伺服器 | 4848 | 可透過管理主控台和 asadmin 公用程式存取網域的管理伺服器。對於管理主控台，請在瀏覽器的 URL 中指定連接埠號碼。從遠端執行 asadmin 指令時，請使用 --port 選項指定連接埠號碼。 |
| HTTP | 8080 | Web 伺服器偵聽連接埠上的 HTTP 請求。若要存取已部署的 Web 應用程式和服務，用戶端應連線到此連接埠。 |
| HTTPS | 8181 | 爲安全通訊配置的 Web 應用程式在單獨的連接埠上進行偵聽。 |
| IIOP | | 企業 Bean (EJB 元件) 的遠端用戶端通過 IIOP 偵聽程式存取 Bean。 |
| IIOP、SSL | | 另一個連接埠由爲安全通訊配置的 IIOP 偵聽程式使用。 |
| IIOP、SSL 和相互認證 | | 另一個連接埠由爲相互 (用戶端和伺服器) 認證配置的 IIOP 偵聽程式使用。 |

◆ ◆ ◆ 第 2 章

# Java Business Integration

Java Business Integration (JBI) 實作 Java Business Integration 的 JSR 208 規格 (http://www.jcp.org/en/jsr/detail?id=208)。此規格是依據 Java Community Process (JCP) 開發的標準，做爲實作服務導向架構 (SOA) 的方法。

JBI 爲外掛程式元件界定一個環境，讓這些元件以直接採取 Web 服務描述語言 (WSDL) 2.0 的服務模型進行互動。

如需管理 JBI 執行階段環境的主要元件及其生命週期狀態的詳細資訊，請參閱 Application Server 管理主控台線上說明。如需有關使用 JBI 指令的資訊，請參閱「Sun Java System Application Server 9.1 Reference Manual」。

## JBI 環境

下列各節涵蓋 JBI 環境的主要元件：



## JBI 元件

### 服務引擎

服務引擎是提供本機服務 (也就是 JBI 環境中的服務) 並使用本機或遠端服務的元件。

### 連結元件

連結元件是 JBI 環境外部的用戶或提供者的代理。連結元件一般以標準通訊協定 (例如 FTP、JMS 或 SMTP) 或外部服務呼叫 (例如 SAP 或 WebSphere MQ) 爲基礎。

JBI 元件具有下列生命週期狀態：

- 已啓動
- 已停止
- 關機

JBI 執行階段可以維持 JBI 元件的生命週期狀態。當 Application Server 關機再重新啓動時，JBI 元件會復原至 Application Server 關機時的狀態。

---

**備註 –** JBI 執行階段會嘗試復原至 JBI 元件的「所需」狀態。例如，假設您嘗試啓動 JBI 元件，但它因爲元件發生錯誤而未啓動。如果重新啓動 Application Server，則 JBI 執行階段會嘗試再次啓動此元件。

---

您可以在 JBI 元件上執行下列作業。如需詳細步驟，請登入「管理主控台」，瀏覽到 JBI 節點並按一下 [元件]，然後按一下 [線上說明]。

- 依 JBI 元件的特定生命週期狀態檢視元件。
- 安裝 JBI 元件。
- 解除安裝 JBI 元件。
- 管理 JBI 元件的生命週期狀態。
- 檢視 JBI 元件的一般特性。
- 檢視 JBI 元件的配置資訊。
- 檢視 JBI 元件的描述元。
- 管理 JBI 元件記錄程式。

### JBI 元件記錄程式

您可以使用 Application Server 管理主控台管理 JBI 元件的記錄層級。有的 JBI 元件可以提供數個記錄程式，其他元件則可能不會提供任何記錄程式。不過，系統會針對整個元件自動顯示記錄程式層級。但僅當元件依據預設名稱實作其記錄程式時，記錄程式層級設定才會有效。JBI 元件的提供者可能會針對如何指定記錄層級提供額外文件。

---

**備註 –** JBI 元件的記錄層級通常繼承自父系記錄程式，例如 JBI 記錄程式。若要檢視和設定父系記錄層級，請在「管理主控台」中，依序選取 [常用工作] 及 [Application Server]。然後，在 [Application Server] 面板中，依序選取 [記錄] 及 [記錄層級]。尋找 JBI 模組的下拉式清單，以檢視並設定父系 JBI 記錄層級。

---

## 服務組件

服務組件是服務單元的集合，可佈建目標元件以便共同提供或使用應用程式的特定服務。服務組件一般是在開發工具環境中建立的，例如 NetBeans Enterprise Pack 所提供的開發工具環境。

服務組件有下列生命週期狀態：

- 已啓動
- 關機
- 已停止

JBI 執行階段可維持服務組件的生命週期狀態。當 Application Server 關機再重新啓動時，服務組件會復原至 Application Server 關機時的狀態。

---

**備註 –** JBI 執行階段會嘗試復原至服務組件的「所需」狀態。例如，假設您嘗試啓動服務組件，但它因爲服務組件發生錯誤而未啓動。如果重新啓動 Application Server，則 JBI 執行階段會嘗試再次啓動服務組件。

---

您可以在服務組件上執行下列作業。如需詳細步驟，請登入「管理主控台」，瀏覽到 JBI 節點並按一下 [服務組件]，然後按一下 [線上說明]。

- 檢視所有服務組件，支援依生命週期狀態進行排序及篩選。
- 部署服務組件。
- 取消部署服務組件。
- 管理服務組件的生命週期。
- 檢視服務組件的一般特性。
- 檢視服務組件的描述元。

## 共用程式庫

共用程式庫提供不屬於單一元件專有，一般由多個 JBI 元件共用的 Java 類別。例如，Java EE 服務引擎需要 WSDL 共用程式庫。

您可以在共用程式庫上執行下列作業。如需詳細步驟，請登入「管理主控台」，瀏覽到 JBI 節點並按一下 [共用程式庫]，然後按一下 [線上說明]。

- 檢視所有共用程式庫。
- 安裝共用程式庫。
- 檢視共用程式庫的一般特性。
- 檢視共用程式庫的描述元。
- 解除安裝共用程式庫。

## JBI 描述元

服務組件、JBI 元件和共用程式庫的描述元檔案 (`jbi.xml`) 提供下列資訊：

- 服務組件：列出服務組件所包含的服務單元，以及每個服務單元的目標。一些服務單元也可能顯示有關連線端點的資訊。
- JBI 元件：列出 JBI 元件 (連結元件或服務引擎) 的類型、元件的說明、元件相關類別路徑的資訊，以及其依賴的任何共用程式庫的名稱。

- 共用程式庫：列出共用程式庫的名稱，以及歸檔檔案 (.jar 檔案) 或其包含的類別檔案子目錄的名稱。

◆ ◆ ◆ 第 3 章

# 3 JDBC 資源

本章說明如何配置存取資料庫的應用程式所要求使用的 JDBC 資源。本章包括下列小節：



## JDBC 資源

爲了儲存、組織和擷取資料，大多數應用程式均使用關聯式資料庫。J2EE 應用程式透過 JDBC API 存取關聯式資料庫。

JDBC 資源 (資料源) 爲應用程式提供連線至資料庫的方法。通常，管理員要爲部署在網域中的應用程式存取的每個資料庫建立 JDBC 資源。(但是，可以爲一個資料庫建立多個 JDBC 資源。)

若要建立 JDBC 資源，請指定識別資源的專屬 JNDI 名稱。(請參閱「JNDI 名稱和資源」一節。)JDBC 資源的 JNDI 名稱應在 `java:comp/env/jdbc` 子環境中。例如，薪水帳單資料庫資源的 JNDI 名稱可爲 `java:comp/env/jdbc/payrolldb`。由於所有資源 JNDI 名稱均在 `java:comp/env` 子環境中，因此在管理主控台中指定 JDBC 資源的 JNDI 名稱時，僅需輸入 `jdbc/`*name*。例如，對於薪水帳單資料庫，可指定 `jdbc/payrolldb`。

# JDBC 連線池

若要建立 JDBC 資源，請指定與其關聯的連線池。多個 JDBC 資源可指定單一連線池。

JDBC 連線池是針對特定資料庫的一組可重複使用的連線。由於每建立一個新的實體連線都會耗費時間，因此伺服器維護了可用連線池以提高效能。當應用程式請求連線時，它可以從池中取得一個連線。應用程式關閉連線時，連線將傳回池中。

連線池的特性可能會隨資料庫供應商的不同而有所不同。某些特性是通用的，例如資料庫名稱 (URL)、使用者名稱和密碼。

另請參閱：


- 第 51 頁的 「JDBC 資源」
- 第 52 頁的 「JDBC 資源和連線池如何協同工作」
- 第 53 頁的 「編輯 JDBC 連線池」


# JDBC 資源和連線池如何協同工作

爲了儲存、組織和擷取資料，大多數應用程式均使用關聯式資料庫。J2EE 應用程式透過 JDBC API 存取關聯式資料庫。應用程式存取資料庫之前，必須先取得連線。

以下是在執行階段應用程式連線至資料庫時所發生的情況：

1. 應用程式透過 JNDI API 進行呼叫以獲取與資料庫關聯的 JDBC 資源 (資料源)。

   如果給定了資源的 JNDI 名稱，命名和目錄服務將查找 JDBC 資源。每個 JDBC 資源指定一個連線池。
2. 通過 JDBC 資源，應用程式獲得一個資料庫連線。

   Application Server 秘密地從與該資料庫相對應的連線池中擷取實體連線。池定義資料庫名稱 (URL)、使用者名稱和密碼等連線屬性。
3. 由於已將應用程式連線至資料庫，因此該應用程式可以讀取和修改資料庫中的資料以及將資料增加到資料庫中。

   應用程式透過對 JDBC API 進行呼叫來存取資料庫。JDBC 驅動程式可將應用程式的 JDBC 呼叫翻譯爲資料庫伺服器的協定。
4. 存取資料庫完成之後，應用程式將關閉該連線。

   Application Server 將連線傳回連線池。連線傳回連線池之後，下一個應用程式便可以使用該連線。

# 設定資料庫存取

若要設定資料庫存取：

1. 安裝支援的資料庫產品。

   如需有關 Application Server 支援的資料庫產品的清單，請參閱版本說明。
2. 安裝適用於該資料庫產品的 JDBC 驅動程式。
3. 使網域的伺服器實例可以存取此驅動程式的 JAR 檔案。
4. 建立資料庫。

   通常，應用程式供應程式提供了用於建立和填入資料庫的程序檔。
5. 爲資料庫建立連線池。
6. 建立指向連線池的 JDBC 資源。

現在若要將 JDBC 驅動程式整合到管理網域中，請執行以下操作之一：

1. 使通用類別載入器可以存取該驅動程式。

   將驅動程式的 JAR 檔案和 ZIP 檔案複製到 *domain-dir*`/lib` 目錄或將其類別檔案複製到 *domain-dir*`/lib/ext` 目錄中。
2. 重新啓動網域。
3. 指出驅動程式 JAR 檔案的完全合格路徑名稱。

# 關於 JDBC 連線池

JDBC 連線池是針對特定資料庫的一組可重複使用的連線。使用管理主控台建立連線池時，管理員實際上是在定義與特定資料庫的連線之各種資訊。

建立池之前，您必須首先安裝並整合 JDBC 驅動程式。建置 [建立連線池] 頁面時，必須輸入特定於 JDBC 驅動程式和資料庫供應商的特定資料。繼續操作之前，請先收集以下資訊：

- 資料庫供應商名稱
- 資源類型，例如 `javax.sql.DataSource` (僅限於本機作業事件)
  `javax.sql.XADataSource` (全域作業事件)
- 資料來源類別名稱
- 必需的特性，例如資料庫名稱 (URL)、使用者名稱和密碼

# 編輯 JDBC 連線池

[編輯 JDBC 連線池] 頁面爲您提供了變更現有池的所有設定 (池的名稱除外) 的方法。

1. 變更一般設定。

一般設定的值取決於安裝的特定 JDBC 驅動程式。這些設定是 Java 程式設計語言中的類別名稱或介面名稱。

| 參數 | 說明 |
| --- | --- |
| 資料來源類別名稱 | 實作 `DataSource` 和/或 `XADataSource` API 特定供應商的類別名稱。該類別位於 JDBC 驅動程式中。 |
| 資源類型 | 選項包括 `javax.sql.DataSource` (僅限於本機作業事件)、`javax.sql.XADataSource` (全域作業事件) 和 `java.sql.ConnectionPoolDataSource` (本機作業事件，可能會改善效能)。 |

2. 變更池設定。

   一組實體資料庫連線保存在池中。應用程式請求連線時，將從池中移除該連線；而應用程式釋放該連線之後，連線將傳回到池中。

| 參數 | 說明 |
| --- | --- |
| 池的初始大小和最小大小 | 池中連線的最小數目。此值還決定首次建立池或 Application Server 啓動時，置於池中的連線數目。 |
| 最大池大小 | 池中連線的最大數目。 |
| 池設定大小數量 | 當池分別朝著最大池大小向上調整，以及朝著最小池大小向下調整時，會以批次方式調整。此值確定批次中的連線數目。將此值設置過大會延遲連線建立和資源回收；而將該值設置過小則會導致效率降低。 |
| 閒置逾時 | 連線可以在儲存區中閒置的最大時間 (以秒表示)。一旦超過此時間，即從池中移除該連線。 |
| 最大等待時間 | 請求連線的應用程式在達到連線逾時之前等待的時間。由於預設等待時間過長，應用程式可能會出現無限期當機的情況。 |

3. 變更連線驗證設定。

   您也可以在將連線傳送給應用程式之前，讓 Application Server 驗證連線。當由於網路出現故障或資料庫伺服器當機造成資料庫不可用時，此驗證可允許 Application Server 自動重新建立資料庫連線。連線驗證會耗用額外的時間，並會略微降低效能。

| 參數 | 說明 |
| --- | --- |
| 連線驗證 | 選取 [需要] 核取方塊以啓用連線驗證。 |

| 參數 | 說明 |
| --- | --- |
| 驗證方法 | Application Server 可以使用三種方法來驗證資料庫連線：auto-commit、metadata 和 table。<br><br>auto-commit 和 metadata - Application Server 透過呼叫 `con.getAutoCommit()` 方法和 `con.getMetaData()` 方法來驗證連線。<br><br>**備註 –** 由於許多 JDBC 驅動程式快取了這些呼叫的結果，因此這兩種方法無法始終提供可靠的驗證。請與驅動程式供應商進行核實，以確定這些呼叫是否被快取。<br><br>table - 應用程式將查詢指定的資料庫表格。此表必須存在並且可以存取，但不要求表的列數。請勿使用包含許多列的現有表或經常存取的表。 |
| 表格名稱 | 如果從 [驗證方法] 組合方塊中選取了表，請在此處指定資料庫表格的名稱。 |
| 一旦失敗 | 選取標記爲 [關閉所有連線] 的核取方塊之後，如果單一連線失敗，Application Server 將關閉池中的所有連線，然後重新建立這些連線。如果未選取此核取方塊，則僅在要使用個別連線時才會重新建立這些連線。 |
| 允許非元件呼叫者 | 若要讓非元件呼叫者 (例如 Servlet 篩選和生命週期模組) 使用此連線池，請按一下此核取方塊。 |

4. 變更作業事件隔離設定。

   由於許多使用者通常可以同步運作地存取一個資料庫，因此可能出現一個作業事件在更新資料而另一個作業事件嘗試讀取同一資料的情況。作業事件的隔絕層級定義了正在更新的資料對於其他作業事件的可視程度。如需有關隔絕層級的詳細資訊，請參閱資料庫供應商的文件。

| 參數 | 說明 |
| --- | --- |
| 非作業事件連線 | 如果您要 Application Server 傳回所有非作業事件連線，請按一下核取方塊。 |
| 作業事件隔離 | 使您可以爲該池的連線選取作業事件隔絕層級。如果不指定此參數，連線將使用 JDBC 驅動程式提供的預設隔絕層級進行作業。 |
| 保證的隔離層級 | 該項僅在指定了隔絕層級的情況下才適用。如果選取 [受保證] 核取方塊，則從池中獲取的所有連線都具有相同的隔絕層級。例如，如果上次使用連線時程式化 (使用 `con.setTransactionIsolation`) 變更了連線的隔離層級，這種機制會將狀態變更回指定的隔離層級。 |

5. 變更特性。

在 [附加特性] 表中，可以指定資料庫名稱 (URL)、使用者名稱和密碼等特性。由於隨資料庫供應商的不同，特性也會有所不同，因此請查閱供應商文件以取得詳細資訊。

# 編輯 JDBC 連線池進階屬性

爲了協助診斷連線漏失並改善易於使用性，Application Server 9.1 提供了數種新屬性，讓您在建立連線池時進行配置。

1. 開啓 [進階] 標籤並指定下列屬性。

| 屬性 | 說明 |
|---|---|
| 名稱 | 要編輯之特性所屬的 JDBC 連線池名稱。不過您無法變更池名稱。 |
| 敘述逾時 | 查詢執行時間如果過久，則在經過這段時間後 (以秒爲單位) 即需終止。Application Server 將在建立的敘述上設定「查詢逾時」。預設值 -1 表示屬性未啓用。 |
| 環繞 JDBC 物件 | 設爲 true 時，應用程式將爲 Statement、PreparedStatement、CallableStatement、ResultSet、DatabaseMetaData 取得環繞的 JDBC 物件。預設值爲 false。 |

2. 如下表所示指定連線設定。

| 屬性 | 說明 |
|---|---|
| 最多驗證一次 | 經過這段時間後 (以秒爲單位)，最多只驗證一次連線。這將有助於減少連線的驗證請求數。預設值 0 表示不啓用連線驗證。 |
| 漏失逾時 | 用來追蹤連線池中連線漏失的時間量 (以秒爲單位)。預設值 0 表示停用連線漏失追蹤。如果啓用連線漏失追蹤，可在 [監視資源] 標籤中取得連線漏失數統計資料。若要檢視此標籤，請前往 [Application Server ] > [監視] > [資源]。 |
| 漏失收回 | 如果啓用此選項，則會在完成漏失連線追蹤後，將漏失的連線復原到池中。 |
| 建立重新嘗試次數 | 如果建立新連線失敗，將進行的嘗試次數。預設值 0 表示不再嘗試重新建立連線。 |

| | |
|---|---|
| 重試間隔 | 指定兩次嘗試建立連線的時間間隔 (以秒爲單位)。預設值是 10 秒。唯有當「建立重試次數」值大於 0 時，才能使用此屬性。 |
| 惰性連線登記 | 啓用此選項可將資源登記至作業事件，但資源必須實際用於方法中。 |
| 惰性關聯 | 對連線執行作業時，連線才會產生惰性關聯。此外，完成作業事件及結束元件方法時，也會取消連線的關聯，如此將有助於重複使用實體連線。預設值爲 false。 |
| 與執行緒建立關聯 | 啓用此選項可使連線與執行緒產生關聯，如此當相同的執行緒需要連線時，就可以重複使用與該執行緒已產生關聯的連線，藉此避免產生從池中取得連線的經常性耗用時間。預設值爲 false。 |
| 相符連線 | 使用此選項可開啓/關閉池的連線比對功能。如果管理員知道池中連線一律是同質的，因此從池中挑選連線時不需要由資源配接卡比對，則可將此選項設爲 false。預設值爲 false。 |
| 最大連線使用率 | 指定池可以重複使用連線的次數。重複使用指定次數的連線後，就會關閉連線。例如，這對避免敘述錯誤這類情形很有用。預設值 0 表示不會重複使用連線。 |

# 關於 JDBC 資源

JDBC 資源 (資料源) 爲應用程式提供連線至資料庫的方法。

建立 JDBC 資源之前，先建立 JDBC 連線池。

建立 JDBC 資源時，必須指定：

1. JNDI 名稱。依照慣例，該名稱應以 `jdbc/` 字串開頭。例如：`jdbc/payrolldb` 。請勿遺漏正斜線。
2. 選取要和新的 JDBC 資源相關聯的連線池。
3. 指定資源的設定。
4. 指定具有可用資源的目標 (叢集和獨立伺服器實例)。

# 特定 JDBC 驅動程式的配置

Application Server 9.1 支援以對應的 JDBC 驅動程式連結至任何資料庫管理系統。下列是受支援的 JDBC 驅動程式和資料庫組合。這些組合已經過 Application Server 9.1 測試，並與 J2EE 相容。它們也支援 CMP。


- 第 58 頁的 「Derby 類型 4 驅動程式」
- 第 59 頁的 「DB2 資料庫的 Sun Java System JDBC 驅動程式 」
- 第 60 頁的 「Oracle 8.1.7 和 9.x 資料庫的 Sun Java System JDBC 驅動程式 」
- 第 60 頁的 「Microsoft SQL Server 資料庫的 Sun Java System JDBC 驅動程式」
- 第 61 頁的 「Sybase 資料庫的 Sun Java System JDBC 驅動程式 」
- 第 61 頁的 「IBM DB2 8.1 類型 2 驅動程式」
- 第 62 頁的 「Sybase ASE 12.5 資料庫的 JConnect 類型 4 驅動程式」
- 第 62 頁的 「MM MySQL 類型 4 驅動程式 (非 XA)」


如需目前受支援之 JDBC 驅動程式的最新清單，請參閱「Sun Java System Application Server 9.1 版本說明」。

其他 JDBC 驅動程式可與 Application Server 9.1 搭配使用，但尚未對這些驅動程式進行 J2EE 相容性測試。雖然 Sun 未提供這些驅動程式的產品支援，但 Sun 提供了這些驅動程式與 Application Server 9.1 搭配使用的有限支援。


- 第 63 頁的 「MM MySQL 類型 4 驅動程式 (僅限 XA)」
- 第 63 頁的 「Oracle 8.1.7 和 9.x 資料庫的 Inet Oraxo JDBC 驅動程式 」
- 第 64 頁的 「Microsoft SQL Server 資料庫的 Inet Merlia JDBC 驅動程式」
- 第 65 頁的 「Sybase 資料庫的 Inet Sybelux JDBC 驅動程式」
- 第 65 頁的 「Oracle 8.1.7 和 9.x 資料庫的 Oracle Thin 類型 4 驅動程式」
- 第 66 頁的 「Oracle 8.1.7 和 9.x 資料庫的 OCI Oracle 類型 2 驅動程式」
- 第 67 頁的 「IBM Informix 類型 4 驅動程式」
- 第 67 頁的 「CloudScape 5.1 類型 4 驅動程式」


如需如何整合 JDBC 驅動程式，以及如何使用管理主控台或指令行介面來實作配置的更多資訊，請參閱「Sun Java System Application Server 9.1 管理指南」。

---

**備註 –** 如果執行 `capture-schema` 指令的 Oracle 資料庫使用者不具有模式，則該使用者需要 ANALYZE ANY TABLE 權限。這些權限是由資料庫管理員授予使用者。如需有關 `capture-schema` 的資訊，請參閱「Sun Java System Application Server 9.1 Reference Manual」。

---

## Derby 類型 4 驅動程式

Derby JDBC 驅動程式預設隨附於 Application Server，但 Solaris 隨附安裝除外 (不含 Derby)。因此除非您擁有的是 Solaris 隨附安裝，否則不需要整合此 JDBC 驅動程式與 Application Server。

Derby 驅動程式的 JAR 檔案是 `derbyclient.jar`。

請使用下列設定來配置連線池：

- **名稱**：稍後配置 JDBC 資源時，請使用此名稱。
- **資源類型**：指定適當值。
- **資料庫供應商**：`Derby`
- **資料來源類別名稱**：指定下列其中一項：

  ```
  org.apache.derby.jdbc.ClientDataSource
  org.apache.derby.jdbc.ClientXADataSource
  ```

- **特性**：
  - **user** - 指定資料庫使用者。

    唯有配置 Derby 以使用驗證時，才需要此特性。依預設，Derby 不使用驗證。提供使用者時，它就是表格所在的模式名稱。
  - **password** - 指定資料庫密碼。

    唯有配置 Derby 以使用驗證時，才需要此特性。
  - **databaseName** - 指定資料庫的名稱。
  - **serverName** - 指定資料庫伺服器的主機名稱或 IP 位址。
  - **portNumber** - 如果資料庫伺服器的連接埠號碼與預設值不同，則指定連接埠號碼。
- **URL**：`jdbc:derby://` *serverName*`:` *portNumber*`/`*databaseName* `;create=true`

  唯有在建立不存在的資料庫時，才加入 `;create=true` 部分。

# DB2 資料庫的 Sun Java System JDBC 驅動程式

此驅動程式的 JAR 檔案是 `smbase.jar`、`smdb2.jar` 和 `smutil.jar`。請使用下列設定來配置連線池：

- **名稱**：稍後配置 JDBC 資源時，請使用此名稱。
- **資源類型**：指定適當值。
- **資料庫供應商**：`DB2`
- **資料來源類別名稱**：`com.sun.sql.jdbcx.db2.DB2DataSource`
- **特性**：
  - **serverName** - 指定資料庫伺服器的主機名稱或 IP 位址。
  - **portNumber** - 指定資料庫伺服器的連接埠號碼。
  - **databaseName** - 予以適當設定。
  - **user** - 予以適當設定。
  - **password** - 予以適當設定。

- **URL**：`jdbc:sun:db2://` *serverName*`:` *portNumber*`;databaseName=`*databaseName*

## Oracle 8.1.7 和 9.x 資料庫的 Sun Java System JDBC 驅動程式

此驅動程式的 JAR 檔案是 `smbase.jar`、`smoracle.jar` 和 `smutil.jar`。請使用下列設定來配置連線池：

- **名稱**：稍後配置 JDBC 資源時，請使用此名稱。
- **資源類型**：指定適當值。
- **資料庫供應商**：`Oracle`
- **資料來源類別名稱**：`com.sun.sql.jdbcx.oracle.OracleDataSource`
- **特性**：
  - **serverName** - 指定資料庫伺服器的主機名稱或 IP 位址。
  - **portNumber** - 指定資料庫伺服器的連接埠號碼。
  - **SID** - 予以適當設定。
  - **user** - 予以適當設定。
  - **password** - 予以適當設定。
- **URL**：`jdbc:sun:oracle://` *serverName*`[:` *portNumber*`][;SID=`*databaseName* `]`

## Microsoft SQL Server 資料庫的 Sun Java System JDBC 驅動程式

此驅動程式的 JAR 檔案是 `smbase.jar`、`smsqlserver.jar` 和 `smutil.jar`。請使用下列設定來配置連線池：

- **名稱**：稍後配置 JDBC 資源時，請使用此名稱。
- **資源類型**：指定適當值。
- **資料庫供應商**： `mssql`
- **資料來源類別名稱**：`com.sun.sql.jdbcx.sqlserver.SQLServerDataSource`
- **特性**：
  - **serverName** - 指定資料庫伺服器的主機名稱或 IP 位址和連接埠。
  - **portNumber** - 指定資料庫伺服器的連接埠號碼。
  - **user** - 予以適當設定。
  - **password** - 予以適當設定。
  - **selectMethod** - 設定爲 `cursor`。
- **URL**：`jdbc:sun:sqlserver://` *serverName*`[:` *portNumber*`]`

# Sybase 資料庫的 Sun Java System JDBC 驅動程式

此驅動程式的 JAR 檔案是 `smbase.jar`、`smsybase.jar` 和 `smutil.jar`。請使用下列設定來配置連線池：

- **名稱**：稍後配置 JDBC 資源時，請使用此名稱。
- **資源類型**：指定適當值。
- **資料庫供應商**：`Sybase`
- **資料來源類別名稱**：`com.sun.sql.jdbcx.sybase.SybaseDataSource`
- **特性**：
  - **`serverName`** - 指定資料庫伺服器的主機名稱或 IP 位址。
  - **`portNumber`** - 指定資料庫伺服器的連接埠號碼。
  - **`databaseName`** - 予以適當設定。這是選擇性特性。
  - **`user`** - 予以適當設定。
  - **`password`** - 予以適當設定。
- **URL**：`jdbc:sun:sybase://` *serverName*`[:` *portNumber*`]`

# IBM DB2 8.1 類型 2 驅動程式

DB2 驅動程式的 JAR 檔案是 `db2jcc.jar`、`db2jcc_license_cu.jar` 和 `db2java.zip`。如下所示設定環境變數：

```
LD_LIBRARY_PATH=/usr/db2user/sqllib/lib:${j2ee.home}/lib
DB2DIR=/opt/IBM/db2/V8.1
DB2INSTANCE=db2user
INSTHOME=/usr/db2user
VWSPATH=/usr/db2user/sqllib
THREADS_FLAG=native
```

請使用下列設定來配置連線池：

- **名稱**：稍後配置 JDBC 資源時，請使用此名稱。
- **資源類型**：指定適當值。
- **資料庫供應商**：`DB2`
- **資料來源類別名稱**：`com.ibm.db2.jcc.DB2SimpleDataSource`
- **特性**：
  - **`user`** - 予以適當設定。
  - **`password`** - 予以適當設定。
  - **`databaseName`** - 予以適當設定。
  - **`driverType`** - 設定爲 `2`。
  - **`deferPrepares`** - 設定爲 `false`。

## Sybase ASE 12.5 資料庫的 JConnect 類型 4 驅動程式

Sybase 驅動程式的 JAR 檔案是 `jconn2.jar`。請使用下列設定來配置連線池：

- **名稱**：稍後配置 JDBC 資源時，請使用此名稱。
- **資源類型**：指定適當值。
- **資料庫供應商**：`Sybase`
- **資料來源類別名稱**：指定下列其中一項：

```
com.sybase.jdbc2.jdbc.SybDataSource
com.sybase.jdbc2.jdbc.SybXADataSource
```

- **特性**：
  - **`serverName`** - 指定資料庫伺服器的主機名稱或 IP 位址。
  - **`portNumber`** - 指定資料庫伺服器的連接埠號碼。
  - **`user`** - 予以適當設定。
  - **`password`** - 予以適當設定。
  - **`databaseName`** - 予以適當設定。請勿指定完整的 URL，僅指定資料庫名稱。
  - **`BE_AS_JDBC_COMPLIANT_AS_POSSIBLE`** - 設定爲 `true`。
  - **`FAKE_METADATA`** - 設定爲 `true`。

## MM MySQL 類型 4 驅動程式 (非 XA)

MySQL 驅動程式的 JAR 檔案是 `mysql-connector-java-` *version*`-bin-g.jar`，例如，`mysql-connector-java-3.1.12-bin-g.jar`。請使用下列設定來配置連線池：

- **名稱**：稍後配置 JDBC 資源時，請使用此名稱。
- **資源類型**：指定適當值。
- **資料庫供應商**：`mysql`
- **資料來源類別名稱**：指定下列其中一項：

```
com.mysql.jdbc.jdbc2.optional.MysqlDataSource
```

- **特性**：
  - **`serverName`** - 指定資料庫伺服器的主機名稱或 IP 位址。
  - **`port`** - 指定資料庫伺服器的連接埠號碼。
  - **`user`** - 予以適當設定。
  - **`password`** - 予以適當設定。
  - **`databaseName`** - 予以適當設定。
  - **`URL`** - 如果使用全域作業事件，則可設定此特性，而不是設定 `serverName`、`port` 和 `databaseName`。

MM MySQL 類型 4 驅動程式未提供設定必要 `relaxAutoCommit` 特性的方法，因此必須藉由設定 **URL** 特性來間接設定：

```
jdbc:mysql://host:port/database?relaxAutoCommit="true"
```

## MM MySQL 類型 4 驅動程式 (僅限 XA)

MySQL 驅動程式的 JAR 檔案是 `mysql-connector-java-` *version*`-bin-g.jar`，例如，`mysql-connector-java-3.1.12-bin-g.jar`。請使用下列設定來配置連線池：

- **名稱**：稍後配置 JDBC 資源時，請使用此名稱。
- **資源類型**：指定適當值。
- **資料庫供應商**：`mysql`
- **資料來源類別名稱**：指定下列其中一項：

  ```
  com.mysql.jdbc.jdbc2.optional.MysqlXADataSource
  ```
- **特性**：
  - **serverName** - 指定資料庫伺服器的主機名稱或 IP 位址。
  - **port** - 指定資料庫伺服器的連接埠號碼。
  - **user** - 予以適當設定。
  - **password** - 予以適當設定。
  - **databaseName** - 予以適當設定。
  - **URL** - 如果使用全域作業事件，則可設定此特性，而不是設定 `serverName`、`port` 和 `databaseName`。

    MM MySQL 類型 4 驅動程式未提供設定必要 `relaxAutoCommit` 特性的方法，因此必須藉由設定 **URL** 特性來間接設定：

    ```
    jdbc:mysql://host:port/database?relaxAutoCommit="true"
    ```

## Oracle 8.1.7 和 9.x 資料庫的 Inet Oraxo JDBC 驅動程式

Inet Oracle 驅動程式的 JAR 檔案是 `Oranxo.jar`。請使用下列設定來配置連線池：

- **名稱**：稍後配置 JDBC 資源時，請使用此名稱。
- **資源類型**：指定適當值。
- **資料庫供應商**：`Oracle`
- **資料來源類別名稱**：`com.inet.ora.OraDataSource`

- **特性：**
  - **user** - 指定資料庫使用者。
  - **password** - 指定資料庫密碼。
  - **serviceName** - 指定資料庫的 URL。以下是語法：

    ```
    jdbc:inetora:server:port:dbname
    ```

    例如：

    ```
    jdbc:inetora:localhost:1521:payrolldb
    ```

    在此範例中，`localhost` 是執行 Oracle 伺服器之機器的主機名稱，`1521` 是 Oracle 伺服器的連接埠號碼，`payrolldb` 則是資料庫的 SID。如需有關資料庫 URL 語法的更多資訊，請參閱 Oracle 文件。
  - **serverName** - 指定資料庫伺服器的主機名稱或 IP 位址。
  - **port** - 指定資料庫伺服器的連接埠號碼。
  - **streamstolob** - 如果 BLOB 或 CLOB 資料類型的大小超過 4 KB，且此驅動程式適用於 CMP，則此特性必須設定爲 `true`。
  - **xa-driver-does-not-support-non-tx-operations** - 設定爲值 `true`。選擇性：唯有從相同的連線池擷取非 XA 與 XA 連線時才需要此設定。可能會降低效能。

    除了設定此特性外，建立兩個連線池也是可行的替代方案，也就是讓非 XA 連線與 XA 連線使用不同的連線池。

# Microsoft SQL Server 資料庫的 Inet Merlia JDBC 驅動程式

Inet Microsoft SQL Server 驅動程式的 JAR 檔案是 `Merlia.jar` 。請使用下列設定來配置連線池：

- **名稱**：稍後配置 JDBC 資源時，請使用此名稱。
- **資源類型**：指定適當值。
- **資料庫供應商**：`mssql`
- **資料來源類別名稱**：`com.inet.tds.TdsDataSource`
- **特性：**
  - **serverName** - 指定資料庫伺服器的主機名稱或 IP 位址和連接埠。
  - **port** - 指定資料庫伺服器的連接埠號碼。
  - **user** - 予以適當設定。
  - **password** - 予以適當設定。

## Sybase 資料庫的 Inet Sybelux JDBC 驅動程式

Inet Sybase 驅動程式的 JAR 檔案是 `Sybelux.jar`。請使用下列設定來配置連線池：

- **名稱**：稍後配置 JDBC 資源時，請使用此名稱。
- **資源類型**：指定適當值。
- **資料庫供應商**：`Sybase`
- **資料來源類別名稱**：`com.inet.syb.SybDataSource`
- **特性**：
  - **`serverName`** - 指定資料庫伺服器的主機名稱或 IP 位址。
  - **`portNumber`** - 指定資料庫伺服器的連接埠號碼。
  - **`user`** - 予以適當設定。
  - **`password`** - 予以適當設定。
  - **`databaseName`** - 予以適當設定。不指定完整的 URL，僅指定資料庫名稱。

## Oracle 8.1.7 和 9.x 資料庫的 Oracle Thin 類型 4 驅動程式

Oracle 驅動程式的 JAR 檔案是 `ojdbc14.jar`。請使用下列設定來配置連線池：

- **名稱**：稍後配置 JDBC 資源時，請使用此名稱。
- **資源類型**：指定適當值。
- **資料庫供應商**：`Oracle`
- **資料來源類別名稱**：指定下列其中一項：

  ```
  oracle.jdbc.pool.OracleDataSource
  oracle.jdbc.xa.client.OracleXADataSource
  ```

- **特性**：
  - **`user`** - 予以適當設定。
  - **`password`** - 予以適當設定。
  - **`URL`** - 使用下列語法指定完整的資料庫 URL：

    ```
    jdbc:oracle:thin:[user/password]@host[:port]/service
    ```

    例如：

    ```
    jdbc:oracle:thin:@localhost:1521:customer_db
    ```

  - **`xa-driver-does-not-support-non-tx-operations`** - 設定爲值 `true`。選擇性：唯有當非 XA 與 XA 連線都是從相同的連線池擷取時才需要。可能會降低效能。

除了設定此特性外，建立兩個連線池也是可行的替代方案，也就是讓非 XA 連線與 XA 連線分別使用一個連線池。

---

**備註 –** 您必須在作業事件服務中設定 `oracle-xa-recovery-workaround` 特性，才能使全域作業事件回復機制正常運作。如需詳細資訊，請參閱第 134 頁的「特定資料庫的解決方法」。

使用此驅動程式時，無法在欄中插入超過 2000 個位元組的資料。爲避免發生此問題，請使用 OCI 驅動程式 (JDBC 類型 2)。

---

# Oracle 8.1.7 和 9.x 資料庫的 OCI Oracle 類型 2 驅動程式

OCI Oracle 驅動程式的 JAR 檔案是 `ojdbc14.jar`。請確定可透過 LD_LIBRARY_PATH 使用共用程式庫，並已設定 ORACLE_HOME 特性。請使用下列設定來配置連線池：

- **名稱**：稍後配置 JDBC 資源時，請使用此名稱。
- **資源類型**：指定適當值。
- **資料庫供應商**：`Oracle`
- **資料來源類別名稱**：指定下列其中一項：

  ```
  oracle.jdbc.pool.OracleDataSource
  oracle.jdbc.xa.client.OracleXADataSource
  ```

- **特性**：
  - **`user`** - 予以適當設定。
  - **`password`** - 予以適當設定。
  - **`URL`** - 使用下列語法指定完整的資料庫 URL：

    `jdbc:oracle:oci:[`*user*`/`*password*`]@`*host*`[:`*port*`]/`*service*

    例如：

    ```
    jdbc:oracle:oci:@localhost:1521:customer_db
    ```

  - **`xa-driver-does-not-support-non-tx-operations`** - 設定爲值 `true`。選擇性：唯有當非 XA 與 XA 連線都是從相同的連線池擷取時才需要。可能會降低效能。

    除了設定此特性外，建立兩個連線池也是可行的替代方案，也就是讓非 XA 連線與 XA 連線分別使用一個連線池。

# IBM Informix 類型 4 驅動程式

請使用下列設定來配置連線池：

- **名稱**：稍後配置 JDBC 資源時，請使用此名稱。
- **資源類型**：指定適當值。
- **資料庫供應商**：`Informix`
- **資料來源類別名稱**：指定下列其中一項：

```
com.informix.jdbcx.IfxDataSource
com.informix.jdbcx.IfxXADataSource
```

- **特性**：
  - **serverName** - 指定 Informix 資料庫伺服器名稱。
  - **portNumber** - 指定資料庫伺服器的連接埠號碼。
  - **user** - 予以適當設定。
  - **password** - 予以適當設定。
  - **databaseName** - 予以適當設定。這是選擇性特性。
  - **IfxIFXHost** - 指定資料庫伺服器的主機名稱或 IP 位址。

# CloudScape 5.1 類型 4 驅動程式

CloudScape 驅動程式的 JAR 檔案是 `db2j.jar`、`db2jtools.jar`、`db2jcview.jar`、`jh.jar`、`db2jcc.jar` 和 `db2jnet.jar`。請使用下列設定來配置連線池：

- **名稱**：稍後配置 JDBC 資源時，請使用此名稱。
- **資源類型**：指定適當值。
- **資料庫供應商**：`Cloudscape`
- **資料來源類別名稱**：`com.ibm.db2.jcc.DB2DataSource`
- **特性**：
  - **user** - 予以適當設定。
  - **password** - 予以適當設定。
  - **databaseName** - 予以適當設定。

◆ ◆ ◆ 第 4 章

# 配置 Java 訊息服務資源

Application Server 透過將 Sun Java System Message Queue (原來稱爲 Sun ONE Message Queue) 軟體整合到 Application Server 中，實作了 Java 訊息服務 (JMS) API。對於基本的 JMS API 管理作業，請使用 Application Server 管理主控台。對於進階作業 (包括管理訊息佇列叢集)，請使用 *MQ-as-install* `/imq/bin` 目錄中提供的工具。如需有關管理 Message Queue 的詳細資訊，請參閱「Message Queue Administration Guide」。

本章描述如何爲使用 Java 訊息服務 (JMS) API 的應用程式配置資源。它包含以下小節：

## JMS 資源

Java 訊息服務 (JMS) API 使用兩種管理物件：

- 允許應用程式以程式化方式建立其他 JMS 物件的連線工廠物件。
- 用作訊息儲存庫的目標。

這些物件是以管理方式建立的，而建立物件的方式則特定於每個 JMS 實作。在 Application Server 中執行以下作業：

- 透過建立連線工廠資源來建立連線工廠
- 透過建立兩個物件來建立目標：
  - 實體目標
  - 參考實體目標的目標資源

JMS 應用程式使用 JNDI API 來存取連線工廠和目標資源。通常，JMS 應用程式至少使用一個連線工廠和一個目標。若要瞭解應建立的資源，請研究應用程式或向應用程式開發者洽詢。

連線工廠分爲三種類型：

- `QueueConnectionFactory` 物件，用於點對點通訊
- `TopicConnectionFactory` 物件，用於出版訂閱通訊

- `ConnectionFactory` 物件，用於點對點通訊和發佈訂閱通訊；建議將這些物件用於新的應用程式

有兩種類型的目標：

- `Queue` 物件，用於點對點通訊
- `Topic` 物件，用於出版訂閱通訊

「*Java EE 5* **教學課程**」中有關 JMS 的章節提供了有關此兩類通訊和 JMS 其他方面的詳細資訊 (請參閱 `http://java.sun.com/javaee/5/docs/tutorial/doc/index.html`)。

建立資源的次序並不重要。

對於 J2EE 應用程式，請在 Application Server 部署描述元中指定連線工廠和目標資源，如下所示：

- 在 `resource-ref` 或 `mdb-connection-factory` 元素中指定連線工廠 JNDI 名稱。
- 在訊息驅動 Bean 的 `ejb` 元素和 `message-destination` 元素中，指定目標資源 JNDI 名稱。
- 在 `message-destination-link` 元素中指定實體目標名稱，該元素在企業 Bean 部署描述元的 `message-driven` 元素或 `message-destination-ref` 元素內。此外，還應在 `message-destination` 元素中指定該實體目標名稱。(`message-destination-ref` 元素替代了在新的應用程式中已停用的 `resource-env-ref` 元素。) 在 Application Server 部署描述元的 `message-destination` 元素中，將實體目標名稱與目標資源名稱連結起來。

# JMS 資源和連接器資源的關係

Application Server 透過使用名爲 `jmsra` 的系統資源配接卡實作 JMS。使用者建立 JMS 資源時，Application Server會自動建立連接器資源，這些連接器資源將顯示在管理主控台樹狀結構檢視的 [連接器] 節點下。

對於使用者建立的每個 JMS 連線工廠，Application Server 均會建立連接器連線池和連接器資源。對於使用者建立的每個 JMS 目標，Application Server 均會建立管理物件資源。使用者刪除 JMS 資源時，Application Server 會自動刪除連接器資源。

您可以使用管理主控台的 [連接器] 節點 (而非 [JMS 資源] 節點) 爲 JMS 系統資源配接卡建立連接器資源。請參閱第 7 章，以取得詳細資訊。

# JMS 連線工廠

JMS 連線工廠爲允許應用程式以程式化方式建立其他 JMS 物件的物件。這些受管理物件會實作 `ConnectionFactory`、`QueueConnectionFactory` 和 `TopicConnectionFactory` 介面。您可以使用 Application Server 管理主控台來建立、編輯或刪除 JMS 連線工廠。建立新的 JMS 連線工廠時，會同時建立該工廠的連接器連線池及連接器資源。

若要使用指令行公用程式管理 JMS 連線工廠，請使用 `create-jms-resource`、`list-jms-resources` 或 `delete-jms-resource` 指令。

# JMS 目標資源

JMS 目標可做爲訊息的儲存庫。您可以使用管理主控台來建立、修改或刪除 JMS 目標資源。若要建立新的 JMS 目標資源，請選取 [資源] > [JMS 資源] > [目標資源]。您可以在 [目標資源] 頁面上，指定以下內容：

- 資源的 JNDI 名稱。建議的作法是使用 JMS 資源的命名子環境前綴 jms/。例如：jms/Queue。
- 資源類型可以是 `javax.jms.Topic` 或 `javax.jms.Queue`。
- 目標資源的其他特性。如需有關所有這些設定和其他特性的詳細資訊，請參閱管理主控台線上說明。

若要使用指令行公用程式管理 JMS 目標，請使用 `create-jms-resource` 或 `delete-jms-resource` 指令。

---

**提示 –** 若要指定 `asadmin create-jms-resource` 指令的 `addresslist` 特性 (格式爲 `host:mqport,host2:mqport,host3:mqport`)，請使用 `\\` 退出「:」。例如，`host1\\:mqport,host2\\:mqport,host3\\:mpqport`。

如需有關使用退出字元的更多資訊，請參閱「`asadmin`(8) 線上手冊」。

---

# JMS 實體目標

若要進行生產，務必建立實體目標。但是，在開發和測試期間，不需要執行此步驟。應用程式首次存取目標資源時，Message Queue 會自動建立目標資源的 Name 特性指定的實體目標。該實體目標是暫時的，並且將在 Message Queue 配置特性指定的時間段後過期。

若要從管理主控台建立實體目標，請選取 [配置] > [實體目標]。在 [建立實體目標] 頁面上，指定實體目標的名稱並選擇目標類型，此類型可以是主題或佇列。如需有關 [實體目標] 頁面上的欄位和特性的詳細資訊，請參閱管理主控台線上說明。

若要進行生產，務必建立實體目標。但是，在開發和測試期間，不需要執行此步驟。應用程式首次存取目標資源時，Message Queue 會自動建立目標資源的 Name 特性指定的實體目標。該實體目標是暫時的，並且將在 Message Queue 配置特性指定的時間段後過期。

若要使用指令行公用程式來管理 JMS 實體目標，請使用 `create-jmsdest`、`flush-jmsdest` 或 `delete-jmsdest` 指令。

# 配置 JMS 提供者特性

使用管理主控台的 [JMS 服務] 頁面配置所有 JMS 連線都將使用的特性。在管理主控台中，選取 [配置] > [Java 訊息服務]。在 [JMS 服務] 頁面上，您可以控制下列一般 JMS 設定。

- 選取 [啓動逾時] 間隔，此項目指出 Application Server 需等候 JMS 服務啓動多久，才中斷啓動程序。
- 選取 [JMS 服務] 類型，以決定您要在本機還是遠端主機上管理 JMS 服務。
- 指定 [啓動引數] 以自訂 JMS 服務啓動。
- 連線遺失時，請選取 [重新連線] 核取方塊指定 JMS 服務是否要嘗試重新連線至訊息伺服器 (或 `AddressList` 中的位址清單)。
- 以秒爲單位指定 [重新連線間隔]。此屬性適用於對 `AddressList` 中每個位址的連線嘗試，以及對該清單中連續位址的連線嘗試。如果該間隔太短，則代理程式將沒有時間恢復。如果該間隔太長，則重新連線會變得遲緩，以至於讓人無法接受。
- 指定嘗試重新連線的次數。在這個欄位中，鍵入用戶端執行階段嘗試連線到清單中的下一個位址前，用戶端對 `AddressList` 中的每個位址嘗試連線 (或重新連線) 的次數。
- 選擇預設的 JMS 主機。
- 在 [位址清單運作方式] 下拉式清單中，選擇是按照 `AddressList` 中的位址順序 (`priority`)，還是按照隨機順序 (`random`) 嘗試連線。
- 在 [位址清單循環] 欄位中，鍵入 JMS 服務建立 (或重新建立) 連線時，在 `AddressList` 中反覆運算的次數。
- 若要使用非預設方案或服務，請在 [MQ 方案] 和 [MQ 服務] 欄位中，鍵入 Message Queue 位址方案名稱和 Message Queue 連線服務名稱。

所有這些特性的值也都能在執行階段更新。不過，只有在更新特性後所建立的連線工廠才會取得更新過的值。現有的連線工廠將繼續保有原來的特性值。另外，要讓絕大部份的值生效，必須重新啓動 Application Server。預設的 JMS 主機是唯一無須重新啓動 Application Server 就能更新的特性。

若要使用指令行公用程式來管理 JMS 提供者，請使用 set 或 `jms-ping` 指令。

## 存取遠端伺服器

將提供者和主機變更爲遠端系統會使所有 JMS 應用程式都在遠端伺服器上執行。若要在使用本機伺服器的同時使用一個或多個遠端伺服器，請使用 AddressList 特性建立連線工廠資源從而建立存取遠端伺服器的連線。

# 外來的 JMS 提供者

Generic Resource Adapter 1.5 for JMS 爲 Java EE Connector 1.5 資源配接卡，可包裝外部 JMS 提供者 (例如 IBM Websphere MQ、Tibco EMS、Sonic MQ 等) 的 JMS 用戶端程式庫，因此可將任何 JMS 提供者與 Java EE 1.4 Application Server (例如 Sun Java System Application Server) 整合。該配接卡是 `.rar` 歸檔，可藉由 Java EE 1.4 Application Server 的管理工具加以部署和配置。

## 配置 JMS 的通用資源配接卡

Application Server 的管理工具可用來部署和配置 JMS 的通用資源配接卡。本節說明如何以 Sun Java System Application Server 來配置 JMS 的通用資源配接卡。

整體而言，可對資源配接卡進行配置以顯示 JMS 提供者是否支援 XA。另外也能指出可以用何種可能的模式與 JMS 提供者進行整合。資源配接卡支援兩種整合模式。第一種模式是以 JNDI 做爲整合方式。在此情況下，受管理物件是在 JMS 提供者的 JNDI 樹狀結構中設定，並由通用資源配接卡進行查找以供使用。如果該模式並不適合用於整合，還可以使用 JMS 管理物件 javabean 類別的 Java 反射做爲整合模式。

您可以使用管理主控台或指令行配置資源配接卡。方法和配置其他資源配接卡均相同。

### 配置通用資源配接卡

部署資源配接卡之前，Application Server 應該能夠使用 JMS 用戶端程式庫。對某些 JMS 提供者而言，用戶端程式庫也能同時包含本機程式庫。在這種情況下，Application Server JVM 也應能夠使用這些原生程式庫。

1. 以部署連接器模組的方式來部署通用資源配接卡。
2. 建立連接器連線池
3. 建立連接器資源。
4. 建立管理物件資源。
5. 在 Application Server 中，對安全性策略進行以下變更：
   - 修改 `sjsas_home/domains/domain1/config/server.policy` **以增加** `java.util.logging.LoggingPermission "control"`

- 修改 sjsas_home/lib/appclient/client.policy 以增加 permission javax.security.auth.PrivateCredentialPermission "javax.resource.spi.security.PasswordCredential ^ \"^\"","read":

# 資源配接卡特性

下表列出在建立資源配接卡時會用到的特性。

| 特性名稱 | 有效值 | 預設值 | 說明 |
|---|---|---|---|
| ProviderIntegrationMode | javabean/jndi | javabean | 決定資源配接卡和 JMS 用戶端之間的整合模式。 |
| ConnectionFactoryClassName | Application Server 類別路徑中可供使用的類別名稱，例如：<br>com.sun.messaging.ConnectionFactory | 無 | JMS 用戶端的 javax.jms.ConnectionFactory 實作的類別名稱。只有當 ProviderIntegrationMode 爲 javabean 時才使用。 |
| QueueConnectionFactoryClassName | Application Server 類別路徑中可供使用的類別名稱，例如：<br>com.sun.messaging.QueueConnectionFactory | 無 | JMS 用戶端的 javax.jms.QueueConnectionFactory 實作的類別名稱。只有當 ProviderIntegrationMode 爲 javabean 時才使用。 |
| TopicConnectionFactoryClassName | Application Server 類別路徑中可供使用的類別名稱，例如：<br>com.sun.messaging.TopicConnectionFactory | 無 | JMS 用戶端的 javax.jms.TopicConnectionFactory 實作的類別名稱。只有將 ProviderIntegrationMode 指定爲 javabean 時才使用。 |
| XAConnectionFactoryClassName | Application Server 類別路徑中可供使用的類別名稱，例如：<br>com.sun.messaging.XAConnectionFactory | 無 | JMS 用戶端的 javax.jms.ConnectionFactory 實作的類別名稱。只有將 ProviderIntegrationMode 指定爲 javabean 時才使用。 |
| XAQueueConnectionFactoryClassName | Application Server 類別路徑中可供使用的類別名稱，例如：<br>com.sun.messaging.XAQueueConnectionFactory | 無 | JMS 用戶端的 javax.jms.XAQueueConnectionFactory 實作的類別名稱。只有將 ProviderIntegrationMode 指定爲 javabean 時才使用。 |

| 特性名稱 | 有效值 | 預設值 | 說明 |
|---|---|---|---|
| XATopicConnection FactoryClassName | Application Server 類別路徑中可供使用的類別名稱，例如：<br>com.sun.messaging. XATopicConnectionFactory | 無 | JMS 用戶端的 javax.jms.XATopicConnectionFactory 實作的類別名稱。只有當 ProviderIntegrationMode 爲 javabean 時才使用。 |
| TopicClassName | Application Server 類別路徑中可供使用的類別名稱，例如：<br>com.sun.messaging.Topic | 無 | JMS 用戶端的 javax.jms.Topic 實作的類別名稱。只有當 ProviderIntegrationMode 爲 javabean 時才使用。 |
| QueueClassName | 應用程式伺服器類別路徑中可供使用的類別名稱，例如：<br>com.sun.messaging.Queue | 無 | JMS 用戶端的 javax.jms.Queue 實作的類別名稱。只有將 ProviderIntegrationMode 指定爲 javabean 時才使用。 |
| SupportsXA | True/false | FALSE | 指定 JMS 用戶端是否支援 XA。 |
| ConnectionFactory Properties | 以逗號分隔的「名稱-值」對 | 無 | 指定 JMS 用戶端的 javabean 特性名稱以及 ConnectionFactory 的值。只有當 ProviderIntegrationMode 爲 javabean 時才需要。 |
| JndiProperties | 以逗號分隔的「名稱-值」對 | 無 | 指定用來連線至 JMS 提供者之 JNDI 的 JNDI 提供者特性。只有當 ProviderIntegrationMode 爲 jndi 時才使用。 |
| CommonSetter MethodName | 方法名稱 | 無 | 指定某些 JMS 供應商在設定其受管理物件的特性時，所使用的一般 setter 方法名稱。只有當 ProviderIntegrationMode 爲 javabean 時才使用。如果是 Sun Java System Message Queue，則此特性稱爲 setProperty。 |
| UserName | JMS 使用者名稱 | 無 | 與 JMS 提供者連線時採用的使用者名稱。 |
| Password | JMS 使用者的密碼 | 無 | 與 JMS 提供者連線時採用的密碼。 |

| 特性名稱 | 有效值 | 預設值 | 說明 |
| --- | --- | --- | --- |
| RMPolicy | ProviderManaged 或 OnePerPhysicalConnection | ProviderManaged | 作業事件管理員使用 XAResource 上的 isSameRM 方法，判斷由兩個 XAResources 所呈現的資源管理員實例是否相同。當 RMPolicy 設定爲 ProviderManaged (預設值) 時，由 JMS 提供者負責決定 RMPolicy，通用資源配接卡中的 XAResource 包裝程式僅將 isSameRM 呼叫委託給訊息佇列提供者的 XA 資源實作。這應該非常適用於大部份的訊息佇列產品。<br><br>一些 XAResource 實作 (例如 IBM MQ 系列) 的每個實體連線都必須有一個資源管理員，因此在單一作業事件中，如果由相同的佇列管理員進行內送和外送通訊時，則可能會發生問題。當 RMPolicy 設定爲 OnePerPhysicalConnection 時，通用資源配接卡中 XAResource 包裝程式實作的 isSameRM 會在委託至包裝物件前，檢查這兩個 XAResource 是否使用相同的實體連線。 |

## ManagedConnectionFactory 特性

ManagedConnectionFactory 特性是在建立 connector-connection-pool 時所指定。建立資源配接卡時指定的所有特性，都可在 ManagedConnectionFactory 中遭到置換。其他僅由 ManagedConnectionFactory 提供的特性如下所示。

| 特性名稱 | 有效值 | 預設值 | 說明 |
| --- | --- | --- | --- |
| ClientId | 有效的用戶端 ID | 無 | 由 JMS 1.1 規格所指定的 ClientID。 |
| ConnectionFactory JndiName | JNDI 名稱 | 無 | 連線工廠的 JNDI 名稱，而此連線工廠連結於 JMS 提供者的 JNDI 樹狀結構。管理員應提供 JMS 提供者本身的所有連線工廠特性 (除了 clientID 之外)。只有當 ProviderIntegratinMode 爲 jndi 時，才使用此特性名稱。 |
| ConnectionValidation Enabled | true/false | FALSE | 若設定爲 true，資源配接卡會使用異常偵聽程式來擷取任何連線異常，並將 CONNECTION_ERROR_OCCURED 事件傳送至 Application Server。 |

# 受管理物件資源特性

本小節所說明的特性會在建立受管理物件資源時指定。所有資源配接卡特性都可在受管理資源物件中置換。其他只有管理物件資源才可提供的特性如下所示。

| 特性名稱 | 有效值 | 預設值 | 說明 |
|---|---|---|---|
| DestinationJndiName | JNDI 名稱 | 無 | 目標的 JNDI 名稱，而此目標連結於 JMS 提供者的 JNDI 樹狀結構。 管理員應提供 JMS 提供者本身的所有特性。只有當 ProviderIntegrationMode 爲 jndi 時，才使用此特性名稱。 |
| DestinationProperties | 以逗號分隔的「名稱-值」對 | 無 | 指定 JMS 用戶端的 javabean 特性名稱以及目標值。只有當 ProviderIntegrationMode 爲 javabean 時才需要。 |

# 啓動規格特性

本小節所述的特性在 MDB 的 Sun 特定部署描述元中，被指定爲 activation-config-properties。所有資源配接卡特性都可在 Activation Spec 中置換。其他只有 ActivationSpec 才可提供的特性如下所示。

| 特性名稱 | 有效值 | 預設值 | 說明 |
|---|---|---|---|
| MaxPoolSize | 整數 | 8 | 資源配接卡爲同步傳遞訊息，可在內部建立的伺服器階段作業池最大大小。這必須等於 MDB 物件最大的池大小。 |
| MaxWaitTime | 整數 | 3 | 資源配接卡自其內部池取得伺服器階段作業時，會等候此特性所指定的秒數。如果超過此限制，訊息傳遞就會失敗。 |
| SubscriptionDurability | 長期或非長期 | 非長期 | 由 JMS 1.1 規格所指定的 SubscriptionDurability。 |
| SubscriptionName | | 無 | 由 JMS 1.1 規格所指定的 SubscriptionName。 |
| MessageSelector | 有效的訊息選擇器 | 無 | 由 JMS 1.1 規格所指定的 MessageSelector。 |

| 特性名稱 | 有效值 | 預設值 | 說明 |
|---|---|---|---|
| ClientID | 有效的用戶端 ID | 無 | 由 JMS 1.1 規格所指定的 ClientID。 |
| ConnectionFactoryJndiName | 有效的 JNDI 名稱 | 無 | 在 JMS 提供者中建立的連線工廠 JNDI 名稱。資源配接卡會使用此連線工廠來建立連線，以接收訊息。只有將 ProviderIntegrationMode 配置爲 jndi 時才使用。 |
| DestinationJndiName | 有效的 JNDI 名稱 | 無 | 在 JMS 提供者中建立的目標 JNDI 名稱。資源配接卡會使用此目標來建立連線，以接收訊息。只有將 ProviderIntegrationMode 配置爲 jndi 時才使用。 |
| DestinationType | javax.jms.Queue 或 javax.jms.Topic | Null | MDB 會偵聽的目標類型。 |
| DestinationProperties | 以逗號分隔的「名稱-值」對 | 無 | 指定 JMS 用戶端的 javabean 特性名稱以及目標值。只有當 ProviderIntegrationMode 爲 javabean 時才需要。 |
| RedeliveryAttempts | 整數 | | 當訊息在 MDB 中造成執行階段異常時，訊息要傳遞的次數。 |
| RedeliveryInterval | 時間 (以秒爲單位) | | 當訊息在 MDB 中造成執行階段異常時，重複傳遞的間隔。 |
| SendBadMessagesToDMD | true/false | False | 指出在超過嘗試傳遞的次數上限時，資源配接卡是否應傳送訊息至停用的訊息目標。 |
| DeadMessageDestinationJndiName | 有效的 JNDI 名稱。 | 無 | 在 JMS 提供者中建立的目標 JNDI 名稱。這是已停用訊息之標的目標。只有當 ProviderIntegrationMode 爲 jndi 時才使用。 |
| DeadMessageDestinationClassName | 目標物件的類別名稱。 | 無 | 只有當 ProviderIntegrationMode 爲 javabean 時才使用。 |
| DeadMessageDestinationProperties | 以逗號分隔的「名稱-值」對 | 無 | 指定 JMS 用戶端的 javabean 特性名稱以及目標值。只有當 ProviderIntegrationMode 爲 javabean 時才需要。 |
| ReconnectAttempts | 整數 | | 當異常偵聽程式擷取到連線錯誤時，嘗試重新連線的次數。 |

| 特性名稱 | 有效值 | 預設值 | 說明 |
| --- | --- | --- | --- |
| ReconnectInterval | 時間 (以秒爲單位) | | 重新連線的時間間隔。 |

◆ ◆ ◆ 第 5 章

# 5 配置 JavaMail 資源

Application Server 包括 JavaMail API。JavaMail API 是一組用於建立郵件系統模型的抽象 API。API 提供了一個獨立於平台和協定的架構來建置郵件和訊息傳送應用程式。JavaMail API 提供讀取與傳送電子郵件的功能。服務提供者可實作特定協定。您可以使用 JavaMail API 爲您的應用程式增加電子郵件功能。JavaMail 可讓 Java 應用程式存取您的網路或網際網路上，能使用網際網路郵件存取通訊協定 (IMAP) 與簡易郵件傳輸協定 (SMTP) 之郵件伺服器。它不提供郵件伺服器功能；因此您必須能夠存取郵件伺服器才能使用 JavaMail。

JavaMail API 可做爲 Java 平台選擇性套裝軟體實作，還可做爲 J2EE 平台的一部分使用。

Application Server 包含 JavaMail API 以及 JavaMail 服務提供者，使應用程式元件可以透過網際網路傳送電子郵件通知，以及從 IMAP 和 POP3 郵件伺服器讀取電子郵件。

如需瞭解有關 JavaMail API 的更多資訊，請造訪 JavaMail 網站，網址是 `http://java.sun.com/products/javamail/`。

本小節包含下列主題：

## 建立 JavaMail 階段作業

若要配置 JavaMail 以便能在 Application Server 中使用，請在 Application Server 管理主控台中建立郵件階段作業。這樣可讓伺服器端元件與應用程式使用 JNDI (利用您爲它們指定的階段作業特性) 存取 JavaMail 服務。建立郵件階段作業時，您可以在管理主控台中指定郵件主機、傳輸與儲存協定，以及預設郵件使用者，這樣使用 JavaMail 的元件就不需要設定這些特性。經常使用電子郵件功能的應用程式可從中獲益，因爲 Application Server 會建立單一階段作業物件，並透過 JNDI 提供給任何需要它的元件。

若要使用管理主控台建立 JavaMail 階段作業，請選取 [資源] —> [JavaMail 階段作業]。指定 JavaMail 設定，如下所示：

- JNDI 名稱：郵件階段作業的唯一名稱。對於 JavaMail 資源，請使用子環境前綴 mail/ 來命名。例如：`mail/MySession`。

- 郵件主機：預設郵件伺服器的主機名稱。如果未提供協定特定的主機特性，Store 和 Transport 物件的連線方法將使用該值。名稱必須可以解析爲實際的主機名稱。
- 預設使用者：連線至郵件伺服器時要提供的使用者名稱。如果未提供協定特定的使用者名稱特性，Store 和 Transport 物件的連線方法使用該值。
- 預設傳回位址：預設使用者的電子郵件地址，格式如下：*username@host.domain*。
- 說明：提供元件的描述性敘述。
- 階段作業：如果您不希望此時啓用郵件階段作業，請取消選取 [啓用] 核取方塊。

此外，只有重新配置郵件提供者以使用非預設儲存或傳輸協定時，才需要定義以下進階設定：

- 儲存協定：定義要使用的儲存物件通訊方法。依預設，儲存協定是 `imap`。
- 儲存協定類別：提供可實作想要之儲存協定的儲存通訊方法類別。依預設，儲存協定類別是 `com.sun.mail.imap.IMAPStore`。
- 傳輸協定：指定傳輸通訊方法。依預設，傳輸協定是 `smtp`。
- 傳輸協定類別：定義傳輸類別的通訊方法。依預設，傳輸協定類別是 `com.sun.mail.smtp.SMTPTransport`。
- 除錯：選取此核取方塊可爲此郵件階段作業啓用額外的除錯輸出功能，包括協定追蹤。如果將 JavaMail 記錄層級設爲 FINE 或更詳細，則系統會產生除錯輸出，並且該輸出會包含在系統記錄檔中。
- 其他特性：建立應用程式所需的特性，例如協定特定的主機或使用者名稱特性。JavaMail API 文件列出了可用的特性。

◆ ◆ ◆ 第 6 章

# 6 JNDI 資源

Java Naming and Directory Interface (JNDI) 是一種應用程式程式設計介面 (API)，可用來存取各種不同的命名和目錄服務。Java EE 元件透過呼叫 JNDI 查詢方法來尋找物件。

JNDI 是 Java 命名和目錄介面 API 的首字母縮略。透過對此 API 進行呼叫，應用程式可以尋找資源和其他程式物件。資源是提供與系統 (如資料庫伺服器和郵件傳送系統) 的連線的程式物件。(JDBC 資源有時被稱爲資料源。)每個資源物件都是由專屬的易懂名稱識別，稱爲 JNDI 名稱。Application Server 包含的命名和目錄服務將資源物件及其 JNDI 名稱連結在一起。若要建立新資源，需要將新的名稱-物件連結輸入到 JNDI 中。

本小節包含以下主題：



## J2EE 命名服務

JNDI 名稱是易懂的的物件名稱。這些名稱透過 J2EE 伺服器提供的命名和目錄服務連結到其物件。由於 J2EE 元件透過 JNDI API 存取此服務，因此物件通常使用其 JNDI 名稱。例如，PointBase 資料庫的 JNDI 名稱爲 `jdbc/Pointbase`。當 Application Server 啓動時，會從配置檔案中讀取資訊，並自動將 JNDI 資料庫名稱增加到名稱空間。

需要 Java EE 應用程式用戶端、企業 Bean 與 Web 元件來存取 JNDI 命名環境。

應用程式元件的命名環境是一種機制，使用它可以在部署或組譯期間自訂應用程式元件的企業邏輯。使用應用程式元件的環境即可對應用程式元件進行自訂，而無需存取或變更應用程式元件的源代碼。

Java EE 容器實作應用程式元件的環境，並將該環境做爲 JNDI 命名環境提供給應用程式元件實例。應用程式元件的環境的使用方式如下：

- 應用程式元件的商業方法使用 JNDI 介面存取該環境。應用程式元件提供者在部署描述元中宣告應用程式元件需要其執行階段環境提供的所有環境項目。
- 容器提供儲存應用程式元件環境的 JNDI 命名環境的實作。容器還提供了部署程式可以用於建立和管理每個應用程式元件的環境的工具。
- 部署程式使用容器提供的工具，可以初始化應用程式元件的部署描述元中宣告的環境項目。部署程式可以設定和修改環境項目的值。
- 容器使環境命名環境在執行階段可用於應用程式元件實例。應用程式元件的實例使用 JNDI 介面獲取環境項目的值。

每個應用程式元件定義了其本身的環境項目集。一個應用程式元件在同一容器內的所有實例共用相同的環境項目。不允許應用程式元件實例在執行階段修改環境。

# 命名參考與連結資訊

資源參照是部署描述元中的元素，可以爲資源識別元件的編碼名稱。更具體地說，編碼名稱參考資源的連線工廠。在下一小節給出的範例中，資源參照名稱爲 `jdbc/SavingsAccountDB`。

資源的 JNDI 名稱與資源參照的名稱不同。使用此命名方法，您需要在進行部署之前先對映這兩個名稱，但此方法也用於分離元件與資源。由於具有此分離功能，因此如果元件在以後需要存取其他資源，則無需變更名稱。這一靈活性使您可以更輕鬆地從預先存在的元件編譯 J2EE 應用程式。

下表列出了 Application Server 所使用的 J2EE 資源的 JNDI 查找及相關參照。

**表 6–1** JNDI 查找及相關參照

| JNDI 查找名稱 | 相關參照 |
|---|---|
| `java:comp/env` | 應用程式環境項目 |
| `java:comp/env/jdbc` | JDBC DataSource 資源管理程式連線 Factory |
| `java:comp/env/ejb` | EJB 參照 |
| `java:comp/UserTransaction` | UserTransaction 參照 |
| `java:comp/env/mail` | JavaMail 階段作業連線 Factory |
| `java:comp/env/url` | URL 連線 Factory |
| `java:comp/env/jms` | JMS 連線 Factory 與目標 |
| `java:comp/ORB` | 應用程式元件共用的 ORB 實例 |

# 使用自訂資源

自訂資源存取本機 JNDI 儲存庫，外部資源存取外部 JNDI 儲存庫。這兩種類型的資源都需要使用者指定的工廠類別元素、JNDI 名稱屬性等。在本小節中，我們將討論如何爲 J2EE 資源配置 JNDI 連線工廠資源，以及如何存取這些資源。

在 Application Server 中，您可以建立、刪除和列示資源以及 `list-jndi-entities`。

# 使用外部 JNDI 儲存庫和資源

通常，Application Server 上執行的應用程式需要存取儲存在外部 JNDI 儲存庫中的資源。例如，一般的 Java 物件可能會以 Java 模式儲存在 LDAP 伺服器中。外部 JNDI 資源元素允許使用者配置此類外部資源儲存庫。外部 JNDI 工廠必須實作 `javax.naming.spi.InitialContextFactory` 介面。

使用外部 JNDI 資源的範例爲：

```
<resources>
 <!-- external-jndi-resource element specifies how to access J2EE resources
 -- stored in an external JNDI repository. The following example
 -- illustrates how to access a java object stored in LDAP.
 -- factory-class element specifies the JNDI InitialContext factory that
 -- needs to be used to access the resource factory. property element
 -- corresponds to the environment applicable to the external JNDI context
 -- and jndi-lookup-name refers to the JNDI name to lookup to fetch the
 -- designated (in this case the java) object.
 -->
  <external-jndi-resource jndi-name="test/myBean"
      jndi-lookup-name="cn=myBean"
      res-type="test.myBean"
      factory-class="com.sun.jndi.ldap.LdapCtxFactory">
    <property name="PROVIDER-URL" value="ldap://ldapserver:389/o=myObjects" />
    <property name="SECURITY_AUTHENTICATION" value="simple" />
    <property name="SECURITY_PRINCIPAL", value="cn=joeSmith, o=Engineering" />
    <property name="SECURITY_CREDENTIALS" value="changeit" />
  </external-jndi-resource>
</resources>
```

◆ ◆ ◆ 第 7 章

# 7 連接器資源

本章描述如何配置用於存取企業資訊系統 (EIS) 的連接器。本章包括下列小節：



## 連接器簡介

連接器模組也稱爲資源配接卡，是允許應用程式與企業資訊系統 (EIS) 進行互動式操作的 Java EE 元件。EIS 軟體包含各種類型的系統：包括企業資源計劃 (ERP)、主機作業事件處理和非關聯式資料庫。安裝連接器模組和其他 Java EE 模組類似，只需部署此連接器模組即可。

連接器連線池是針對特定 EIS 的一組可重複使用的連線。若要建立連接器連線池，請指定與池關聯的連接器模組 (資源配接卡)。

連接器資源是爲應用程式提供到 EIS 的連線的程式物件。若要建立連接器資源，請指定其 JNDI 名稱及其關聯的連線池。多個連接器資源可以指定單一連線池。應用程式可透過查找資源的 JNDI 名稱定位資源。(如需有關 JNDI 的更多資訊，請參閱 [JNDI 名稱和資源] 部分)。EIS 的連接器資源的 JNDI 名稱通常位於 `java:comp/env/`*eis-specific* 子環境中。

Application Server 使用連接器模組 (資源配接卡) 實作 JMS。請參閱「JMS 資源和連接器資源的關係」一節。

# 管理連接器連線池

若要建立、編輯和刪除「連接器連線池」，請在「管理主控台」 中，按一下 [資源] —> [連接器連線池]。請參閱管理主控台線上說明，以取得管理連接器連線池的詳細說明。

## ▼ 設定 EIS 存取

1. **部署 (安裝) 連接器。請參閱管理主控台線上說明，以取得部署連接器的詳細說明。**
2. **為連接器建立連線池。**
3. **建立與連線池關聯的連接器資源。**

# 管理連接器資源

若要建立、編輯和刪除「連接器連線池」，請在「管理主控台」中，按一下 [資源] —> [連接器]。請參閱管理主控台線上說明，以取得管理連接器連線池的詳細說明。

# 管理受管理的物件資源

封裝在資源配接卡 (連接器模組) 中的受管理物件爲應用程式提供專用功能。例如，透過受管理物件可存取資源配接卡及其相關 EIS 的專屬剖析器。可以管理該物件，亦即，管理員可配置該物件。若要配置物件，請在 [建立或編輯管理物件資源] 頁面上增加名稱－值特性對。建立受管理物件資源時，請將受管理物件與 JNDI 名稱相關聯。

若要建立、編輯和刪除「連接器連線池」，請在「管理主控台」中，按一下 [資源] —> [管理物件資源]。請參閱管理主控台線上說明，以取得管理連接器連線池的詳細說明。

◆ ◆ ◆ 第 8 章

# 8 J2EE 容器

J2EE 容器爲 J2EE 應用程式元件提供執行階段支援。J2EE 應用程式元件使用容器的協定和方法存取伺服器提供的其他應用程式元件和服務。Application Server 提供應用程式用戶端容器、applet 容器、Web 容器和EJB 容器。如需有關顯示容器的圖解，請參閱第 29 頁的 「Application Server 架構」。

本章說明下列容器：



## Web 容器

Web 容器是容納 Web 應用程式的 J2EE 容器。Web 容器爲開發者提供執行 servlet 和 JavaServer Pages (JSP 檔案) 的環境，以擴充 Web 伺服器的功能。

## EJB 容器

企業 Bean (EJB 元件) 是包含商務邏輯的 Java 程式設計語言伺服器元件。EJB 容器提供對企業 Bean 的本機和遠端存取。

企業 Bean 分爲三種類型：階段作業 Bean、實體 Bean 和訊息驅動 Bean。階段作業 Bean 表示暫態物件和程序，並且通常由單一用戶端使用。實體 Bean 表示持續性資料，通常維護在資料庫中。訊息驅動 Bean 用於將訊息非同步傳送到應用程式模組和服務中。

容器負責建立企業 Bean、將企業 Bean 連結至命名服務以使其他應用程式元件可以存取企業 Bean、確定僅授權的用戶端才能存取企業 Bean 的方法、將 Bean 的狀態儲存到永久性儲存裝置中、快取 Bean 的狀態，以及在必要時啓動或鈍化 Bean。

◆ ◆ ◆ 第 9 章

# 9 配置安全性

安全性實質上就是資料保護：如何在儲存或傳輸資料時防止對資料進行未授權的存取或損毀。Application Server 具有基於 Java EE 標準的動態可延伸安全性架構。它內置的安全性功能包括密碼學、認證與授權以及公開金鑰基礎架構。Application Server 是基於 Java 安全性模型建置的，該安全性模型使用了一個沙箱，應用程式可以在沙箱中安全地執行，而不會給系統或使用者帶來潛在的危險。本節討論以下主題：



## 瞭解應用程式和系統安全性

從廣泛意義上講，應用程式安全性有兩種：

- 在**程式安全性**中，開發者編寫的應用程式碼負責處理安全性事務。做爲管理員，您對此機制沒有任何控制權。由於程式化安全性將安全性配置硬編碼到應用程式中而不是透過 J2EE 容器對其進行管理，因此一般不提倡使用這種程式化安全性。
- 以**宣告性安全性**來說， 容器 (Application Server) 透過應用程式的部署描述元處理安全性。您可以透過直接編輯部署描述元或使用 `deploytool` 等工具來控制宣告性安全性。由於可以在完成應用程式開發之後變更部署描述元，因此宣告性安全性具有更大靈活性。

除了應用程式安全性以外，還有影響 Application Server 系統中所有應用程式的**系統安全性**。

程式安全性受應用程式開發者的控制，因此本文件不對其進行討論；宣告性安全性受應用程式開發者的控制要少一些，本文件中只偶爾涉及到宣告性安全性。本文件主要針對系統管理員，因此主要討論系統安全性。

# 管理安全性的工具

Application Server 提供了以下用於管理安全性的工具：

- 管理主控台，一種基於瀏覽器的工具，用於配置整個伺服器的安全性，管理使用者、群組和範圍以及執行系統範圍內的其他安全性作業。如需有關管理主控台的一般介紹，請參閱第 31 頁的 「管理工具」。如需安全性作業的簡介，請參閱管理主控台線上說明。
- `asadmin`，指令行工具，可以執行管理主控台能夠執行的許多作業。您還可以使用 `asadmin` 執行使用管理主控台無法執行的某些作業。您可以從指令提示符號或程序檔中執行 `asadmin` 指令，以自動執行重複的作業。如需有關 `asadmin` 的一般簡介，請參閱第 31 頁的 「管理工具」。

Java 2 Platform, Standard Edition (J2SE) 提供了兩個用於管理安全性的工具：

- `keytool`，指令行公用程式，用於管理數位憑證和鍵對。使用 `keytool` 可以管理 `certificate` 範圍內的使用者。
- `policytool`，圖形化公用程式，用於管理系統範圍內的 Java 安全策略。做爲管理員，您很少會用到 `policytool`。

如需有關使用 `keytool`、`policytool` 和其他 Java 安全性工具的更多資訊，請參閱位於以下位置的「Java 2 SDK Tools and Utilities」：`http://java.sun.com/j2se/1.4.2/docs/tooldocs/tools.html#security`。

在企業設定檔中，還可以使用另外兩個實作 Network Security Services (NSS) 的工具管理安全性。如需有關 NSS 的更多資訊，請至 `http://www.mozilla.org/projects/security/pki/nss/`。用於管理安全性的工具包括：

- `certutil`，指令行公用程式，用於管理憑證和金鑰資料庫。
- `pk12util`，指令行公用程式，用於以 PKCS12 格式在憑證/金鑰資料庫和檔案之間匯入和匯出金鑰及憑證。

如需有關使用 `certutil`、`pk12util` 和其他 NSS 安全性工具的更多資訊，請參閱位於以下位置的「NSS Security Tools」：`http://www.mozilla.org/projects/security/pki/nss/tools`。

# 管理密碼安全性

在 Application Server 中，包含特定網域規格的 domain.xml 檔案最初包含純文字形式的 Sun Java System Message Queue 代理程式密碼。domain.xml 檔案中包含此密碼的元素爲 jms-host 元素的 admin-password 屬性。由於在安裝期間不能變更此密碼，因此它不會對安全性產生很大的影響。

但是，您可以使用管理主控台增加使用者和資源，並爲這些使用者和資源指定密碼。部分密碼將以純文字形式寫入 domain.xml 檔案，例如用於存取資料庫的密碼。將這些純文字形式的密碼保存在 domain.xml 檔案中可能會破壞安全性。您可以對 domain.xml 中的任何密碼進行加密，包括 admin-password 屬性或資料庫密碼。下列主題中包含有關安全性密碼的管理說明：



## 對 domain.xml 檔案中的密碼進行加密

若要對 domain.xml 檔案中的密碼進行加密，請依照下列步驟執行：

1. 在 domain.xml 檔案常駐的目錄 (依預設，此目錄爲 *domain-dir*/config) 中，執行以下 asadmin 指令：

   ```
   asadmin create-password-alias --user admin alias-name
   ```

   例如，

   ```
   asadmin create-password-alias --user admin jms-password
   ```

   螢幕將顯示輸入密碼提示 (在本範例中爲 admin)。請參閱 create-password-alias、list-password-aliases 和 delete-password-alias 指令的線上手冊，以取得更多資訊。

2. 移除並替代 domain.xml 中的密碼。使用 asadmin set 指令可以完成此作業。使用 set 指令實現此目的的範例如下：

   ```
   asadmin set --user admin server.jms-service.jms-host.
   default_JMS_host.admin-password='${ALIAS=jms-password}'
   ```

   **備註 –** 將別名密碼以單引號括住，如範例所示。

3. 重新啓動相關網域的 Application Server。

# 保護具有編碼密碼的檔案

某些檔案包含需要使用檔案系統許可權進行保護的編碼密碼。這些檔案包括：

- *domain-dir*/master-password

  此檔案包含編碼主密碼，並且應使用檔案系統許可權 600 對其進行保護。
- 若要使用 --passwordfile 引數將已建立的密碼檔案當成引數傳送給 asadmin，應使用檔案系統許可權 600 對密碼檔案進行保護。

# 變更主密碼

主密碼 (MP) 是全局性的共用密碼。它從不用於認證，也從不會在網路上傳輸。此密碼是整體安全性的重點；使用者可以選擇在需要時手動輸入此密碼，也可以將其隱藏在檔案中。它是系統中最敏感的資料。使用者可以移除此檔案，以強制系統提示您輸入主密碼。主密碼變更時，會重新儲存在主密碼金鑰庫中，也就是 Java JCEKS 類型金鑰庫。

若要變更主密碼，請執行下列步驟：

1. 停止網域的 Application Server。使用 asadmin change-master-password 指令會提示輸入舊密碼和新密碼，然後重新加密所有附屬項目。例如：

```
asadmin change-master-password>
Please enter the master password>
Please enter the new master password>
Please enter the the new master password again>
```

2. 重新啓動 Application Server。

**注意 –** 此時，不能啓動正在執行的伺服器實例並且不能重新啓動正在執行的伺服器實例，除非已變更這些實例所對映的節點代理程式上的 SMP。如果在變更伺服器實例的 SMP 之前重新啓動了該伺服器實例，它將無法啓動。

3. 一次停止一個節點代理程式及其相關伺服器。再次執行 asadmin change-master-password 指令，然後重新啓動節點代理程式及其相關伺服器。
4. 繼續對下一個節點代理程式執行這些步驟，直到對所有節點代理程式均已執行這些步驟。這樣可以完成輪替變更。

# 使用主密碼和金鑰庫

主密碼是安全金鑰庫的密碼。建立新的應用程式伺服器網域之後，即會產生新的自我簽署憑證，並將其儲存在使用主密碼鎖定的相關金鑰庫中。如果主密碼並非預設，start-domain 指令會提示您輸入主密碼。輸入正確的主密碼之後，網域就會啓動。

建立與網域相關聯的節點代理程式之後，節點代理程式就會將資料與網域進行同步化。執行此項作業的同時，還會將金鑰庫同步化。任何由此節點代理程式控制的伺服器實例都必須開啓金鑰庫。由於此金鑰庫基本上和網域建立程序所建立的金鑰庫相同，因此只能由相同的主密碼開啓。但是主密碼本身永遠都不會同步化；意即在同步化期間，並不會將主密碼傳送到節點代理程式，但它必須可供節點代理程式在本機使用。這就是在建立和/或啓動節點代理程式時會提示您輸入主密碼的原因，而您所輸入的密碼，必須和您在建立/啓動網域時所輸入的密碼相同。若網域的主密碼有所變更，則必須執行相同的步驟，在每個和此網域相關聯的節點代理程式上變更主密碼。

## 變更管理員密碼

第 93 頁的 「管理密碼安全性」中論述了如何加密管理員密碼。強烈建議您加密管理員密碼。若要在加密管理員密碼之前變更管理員密碼，請使用 `asadmin set` 指令。使用 `set` 指令實現此目的的範例如下：

```
asadmin set --user admin server.jms-service.jms-host.default_JMS_host.admin-password=new_pwd
```

請參閱管理主控台線上說明，取得使用管理主控台變更管理密碼的說明。

# 關於認證與授權

認證與授權是應用程式伺服器安全性的核心概念。以下主題論述了與認證和授權相關的內容：



## 認證實體

**認證**是實體 (使用者、應用程式或元件) 用於確定其他實體是否是其宣告的實體的方法。實體使用**安全性憑證**對其自身進行認證。憑證可以是使用者名稱和密碼、數位證書或其他憑證。

通常，認證表示使用者使用使用者名稱和密碼登入某個應用程式；也可以指 EJB 從伺服器請求資源時，提供安全性憑證。通常，伺服器或應用程式要求使用者端進行認證；另外，使用者端也可以要求伺服器對其自身進行認證。如果認證是雙向的，則稱爲相互認證。

當實體嘗試存取受保護的資源時，Application Server 將使用爲該資源配置的認證機制來確定是否授予存取權。例如，使用者可以在 Web 瀏覽器中輸入使用者名稱和密碼，如

果應用程式順利完成了對這些憑證的驗證，則表示該使用者已通過認證。在此階段作業的剩餘時間內，該使用者將始終與這個經過認證的安全性身份相關聯。

Application Server 支援四種認證類型。應用程式在其部署描述元中指定所使用的認證類型。

**表 9–1** Application Server 認證方法

| 認證方法 | 通訊協定 | 說明 | 使用者憑證加密 |
|---|---|---|---|
| BASIC | HTTP (SSL 選取性) | 使用伺服器的內建快顯式登入對話方塊。 | 無，除非使用 SSL。 |
| FORM | HTTP (SSL 選取性) | 應用程式提供它自己的自訂登入頁面和錯誤頁面。 | 無，除非使用 SSL。 |
| CLIENT-CERT | HTTPS (基於 SSL 的 HTTP) | 伺服器使用公開金鑰證書認證使用者端。 | SSL |

### 驗證單次登入

單次登入可以讓一個虛擬伺服器實例中的多個應用程式共用使用者認證狀態。使用單次登入，登入到一個應用程式的使用者也會隱式登入到需要相同認證資訊的其他應用程式。

單次登入基於群組。其部署描述元定義相同的**群組**並使用相同認證方法 (BASIC、FORM、CLIENT-CERT) 的所有 Web 應用程式，均共用單次登入。

對於爲 Application Server 定義的虛擬伺服器，依預設已啓用單次登入。

## 對使用者進行授權

使用者通過認證後，**授權**層級將決定該使用者可以執行哪些作業。使用者獲得的授權以其**角色**爲依據。例如，人力資源應用程式可以授權管理者檢視所有員工的個人資訊，但每個員工僅能檢視自己的個人資訊。如需有關角色的更多資訊，請參閱第 97 頁的「瞭解使用者、群組、角色和範圍」。

## 指定 JACC 提供者

JACC (Java 容器授權合約) 屬於 Java EE 規格，它定義可插接式授權提供者介面。這可以讓管理員設置協力廠商外掛程式模組以執行授權。

依預設，Application Server 提供符合 JACC 規格並基於檔案的簡單授權引擎。還可以指定其他協力廠商 JACC 提供者。

JACC 提供者使用 Java 認證與授權服務 (JAAS) API。JAAS 可以讓服務對使用者進行認證並強制對使用者進行存取控制。JAAS 實作了 Java 技術版本的標準可插接式認證模組 (PAM) 框架。

## 稽核認證與授權決策

Application Server 可以透過**稽核模組**提供對所有認證與授權決策的稽核追蹤。Application Server 提供預設的稽核模組，還提供了自訂稽核模組的功能。

## 配置訊息安全性

**訊息安全性**可以讓伺服器在訊息層執行 Web 服務呼叫和回應的點對點認證。Application Server 使用 SOAP 層上的訊息安全性提供者實作訊息安全性。訊息安全性提供者提供以下資訊，例如請求訊息和回應訊息所需的認證類型。受支援的認證類型包括：

- 寄件者認證，包括使用者名稱密碼認證。
- 特性認證，包括 XML 數位簽名。

此發行版本包括兩個訊息安全性提供者。可以爲 SOAP 層的認證配置訊息安全性提供者。可以配置的提供者包括 `ClientProvider` 和 `ServerProvider`。

訊息層安全性支援以 (可插接式) 認證模組的形式整合在 Application Server 及其用戶端容器中。依預設，Application Server 中的訊息層安全性處於停用狀態。

您可以爲整個 Application Server 或者爲特定的應用程式或方法配置訊息層安全性。第 10 章中論述了如何配置 Application Server 層級的訊息安全性。**「開發者指南」**討論如何在應用程式層級上配置訊息安全性。

# 瞭解使用者、群組、角色和範圍

Application Server 對以下實體強制執行其認證與授權策略：

- 第 98 頁的「使用者」：*Application Server* **中定義的**個人身份。通常，使用者是指個人、一個軟體元件 (例如企業 Bean)，甚至是一種服務。經過認證的使用者有時被稱爲**主體**。使用者有時被稱爲**個人**。
- 第 98 頁的「群組」：*Application Server* **中定義的**一組使用者，依循一般特性進行分類。
- 第 98 頁的「角色」：**由應用程式定義的**命名授權層級。可以將角色比作開鎖的鑰匙。許多人都可以有此鑰匙的複製鑰匙。鎖不關心誰要造訪，而只關心使用的鑰匙是否正確。
- 第 99 頁的「範圍」：包含使用者和群組資訊及其相關安全性憑證的儲存庫。範圍也稱爲**安全策略網域**。

**備註 –** 儘管使用者和群組是爲整個 Application Server 指定的，但是每個應用程式都需要定義自己的角色。封裝和部署應用程式時，應用程式會指定使用者/群組和角色之間的對映，如下圖所示。

# 使用者

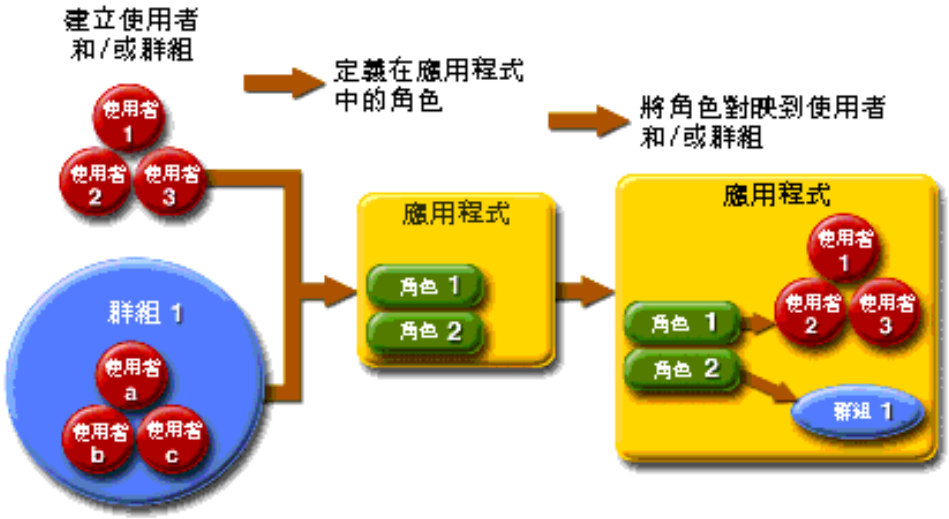


**圖 9–1** 角色對映

**使用者**是已在 Application Server 中定義的個人 (或應用程式) 身份。使用者可以與群組關聯。Application Server 認證服務可以管理多個範圍中的使用者。

# 群組

*J2EE* **群組** (或簡稱群組) 是依循一般特性 (例如職務或用戶設定檔) 進行分類的使用者類別。例如，假定電子商務應用程式的使用者屬於 `customer` 群組，但是大用戶可以屬於 `preferred` 群組。將使用者進行群組分類可以簡化有大量使用者時的存取控制。

# 角色

**角色**定義使用者可以存取哪些應用程式以及每個應用程式的哪些部分，以及使用者可以執行的作業。亦即，角色決定了使用者的授權層級。

例如，假定在人事應用程式中，所有員工均可以存取電話號碼和電子郵件位址但只有管理者才能存取薪水資訊。該應用程式至少可以定義兩個角色：employee 和 manager；僅允許 manager 角色的使用者檢視薪水資訊。

角色與使用者群組的不同之處在於，角色在應用程式中定義功能，而使用者群組是以某一方式相關的一組使用者。例如，假定在人事應用程式中有 full-time、part-time 和 on-leave 幾個群組，但所有這些群組中的使用者仍是 employee 角色。

角色是在應用程式部署描述元中定義的。相反，群組是針對整個伺服器和範圍而定義的。應用程式開發者或部署者在每個應用程式的部署描述元中將角色對映至一個或多個群組。

## 範圍

**範圍** (也稱爲**安全策略網域**或**安全網域**)，是伺服器定義和強制執行一般安全策略的範圍。在實際應用中，範圍是伺服器儲存使用者和群組資訊的儲存庫。

Application Server 預先配置了三個範圍：file (初始預設範圍)、certificate 和 admin-realm。您還可以設定 ldap、 JDBC、solaris 範圍或自訂範圍。應用程式可以在其部署描述元中指定要使用的範圍。如果應用程式不指定範圍，Application Server 將使用其預設範圍。

在 file 範圍中，伺服器將使用者憑證儲存在本地名爲 keyfile 的檔案中。您可以使用管理主控台來管理 file 範圍中的使用者。

在 certificate 範圍中，伺服器將使用者憑證儲存在憑證資料庫中。使用 certificate 範圍時，伺服器結合使用憑證和 HTTPS 通訊協定認證 Web 用戶端。如需有關憑證的更多資訊，請參閱第 101 頁的 「證書和 SSL 簡介」。

admin-realm 也是一個 FileRealm，它將管理員使用者憑證儲存在本地名爲 admin-keyfile 的檔案中。您可以使用管理主控台管理此範圍中的使用者，其方法與您管理 file 範圍中的使用者的方法相同。

在 ldap 範圍中，伺服器將從簡易資料存取協定 (LDAP) 伺服器 (例如 Sun Java System Directory Server) 中取得使用者憑證。LDAP 是一種可以讓任何人在網路 (無論是公共網際網路還是企業內部網路) 中尋找組織、個人和其他資源 (例如檔案和裝置) 的協定。如需有關管理 ldap 範圍中的使用者和群組的資訊，請參閱您的 LDAP 伺服器文件。

在 JDBC 範圍中，伺服器會從資料庫取得使用者憑證。Application Server 會使用資料庫資訊，以及配置檔的已啓用 JDBC 範圍選項。

在 solaris 範圍中，伺服器將從 Solaris 作業系統中取得使用者憑證。Solaris 9 OS 和更高版本支援此範圍。如需有關管理 solaris 範圍中的使用者和群組的資訊，請參閱您的 Solaris 文件。

自訂範圍是使用者憑證的任何其他儲存庫，例如關聯式資料庫或協力廠商元件。如需更多資訊，請參閱管理主控台線上說明。

## ▼ 配置 Java EE 應用程式的 JDBC 範圍

Application Server 可讓您在 JDBC 範圍中指定使用者的憑證，而不是在連線池中指定。使用 JDBC 範圍取代連線池，可防止其他應用程式瀏覽使用者憑證的資料庫表格。使用者的憑證是使用者的名稱和密碼。

---

**備註 –** 依預設，不支援在 JDBC 範圍中儲存純文字的密碼。一般情況下不應該儲存純文字密碼。

---

1. **建立資料庫表格，在其中儲存範圍的使用者憑證。**

   建立資料庫表格的方法，需視所使用的資料庫而定。

2. **將使用者憑證增加到在 步驟 1 中建立的資料庫表格。**

   將使用者的憑證增加到資料庫表格的方法，需視所使用的資料庫而定。

3. **建立 JDBC 範圍。**

   使用管理主控台 GUI 來達成此目的。如需有關建立 JDBC 範圍的說明，請參閱管理主控台 GUI 的線上說明。

4. **將在步驟 3 中建立的範圍指定為應用程式的範圍。**

   若要指定範圍，請針對應用程式修改適當的部署描述元：

   - **對於企業歸檔 (EAR) 檔案中的企業應用程式，請修改 `sun-application.xml` 檔案。**
   - **對於 Web 應用程式歸檔 (WAR) 檔案中的 Web 應用程式，請修改 `web.xml` 檔案。**
   - **對於 EJB JAR 檔案中的企業 Bean，請修改 `sun-ejb-jar.xml` 檔案。**

   如需有關如何指定範圍的更多資訊，請參閱「Sun Java System Application Server 9.1 Developer's Guide」中的「How to Set a Realm for an Application or Module」。

5. **將安全性角色指定給範圍中的使用者。**

   若要將安全性角色指定給使用者，請將 `security-role-mapping` 元素增加到在 步驟 4 中所修改的部署描述元。

   下列範例顯示了 `security-role-mapping` 元素，此元素會將安全性角色 `Employee` 指定給使用者 `Calvin`。

```
<security-role-mapping>
    <role-name>Employee</role-name>
    <principal-name>Calvin</principal-name>
  </security-role-mapping>
```

# 證書和 SSL 簡介

本節論述以下主題：



## 關於數位證書

**數位憑證** (或簡稱憑證) 是在網際網路上唯一識別人員和資源的電子檔案。證書還可以使兩個實體之間能夠進行安全、機密的通訊。

證書有很多種類型，例如個人證書 (由個人使用) 和伺服器證書 (用於透過安全套接字層 [SSL] 技術在伺服器和使用者端之間建立安全階段作業)。如需有關 SSL 的更多資訊，請參閱第 102 頁的 「關於安全套接字層」。

憑證以**公開金鑰密碼學**爲基礎，公開金鑰密碼學使用數位**金鑰** (很長的數位) 對資訊進行**加密**或編碼，從而使資訊只能被指定的接收者讀取。然後，接收者對資訊進行**解密** (解碼)，即可讀取該資訊。

一個金鑰對包含一個公開金鑰和一個私密金鑰。所有者對公開金鑰進行發放並使任何人都可以使用該公開金鑰。但是所有者永遠不會發放私密金鑰；私密金鑰始終是保密的。由於金鑰與數學相關，因此使用了金鑰對中的一個金鑰進行加密的資料只能透過金鑰對中的另一個金鑰進行解密。

證書就好像一本護照：它可以識別持有者並提供其他重要資訊。憑證由稱爲**憑證授權單位** (CA) 的可信任的協力廠商發行。CA 類似於護照申領辦公室：它將驗證憑證持有者的身份並對憑證進行簽署，以使他人無法僞造或竄改憑證。CA 對證書進行簽名之後，持有者可以提供該證書做爲身份證明並建立加密的機密通訊。

最重要的是，憑證會將所有者的公開金鑰連結至所有者身份。與護照將照片連結至其持有者的個人資訊類似，證書將公開金鑰連結至有關其所有者的資訊。

除了公開金鑰以外，證書通常還包括以下資訊：

- 持有者的名稱和其他標識，例如使用憑證之 Web 伺服器的 URL 或個人的電子郵件地址。
- 發行證書的 CA 的名稱。
- 過期日期。

數位憑證受 x.509 格式的技術規格約束。爲了驗證`certificate`範圍中某個使用者的身份，認證服務會將 X.509 憑證的共用名稱欄位當成主要名稱，對 X.509 憑證進行驗證。

### 關於證書鏈

Web 瀏覽器已預先配置了一組瀏覽器自動信任的**根** CA 憑證。來自其他憑證授權單位的所有憑證都必須附帶**憑證鏈**，以驗證這些憑證是否有效。憑證鏈是由連續 CA 憑證發行的憑證序列，最終以根 CA 憑證結束。

憑證最初產生時是**自簽名**憑證。自簽名證書是其發行者 (簽名者) 與主旨 (其公開金鑰由該證書進行認證的實體) 相同的證書。如果所有者向 CA 傳送證書簽名請求 (CSR)，然後輸入回應，自簽名證書將被證書鏈取代。鏈的底部是由 CA 發行的、用於認證主旨的公開金鑰證書 (回覆)。鏈中的下一個證書是認證 CA 公開金鑰的證書。通常，這是一個自簽名證書 (即，來自 CA、用於認證其自身公開金鑰的證書)，並且是鏈中的最後一個證書。

在其他情況下，CA 可能會傳回一個證書鏈。在此情況下，鏈的底部證書是相同的 (由 CA 簽名的證書，用於認證金鑰項目的公開金鑰)，但是鏈中的第二個證書是由其他 CA 簽名的證書，用於認證您向其傳送了 CSR 的 CA 的公開金鑰。然後，鏈中的下一個憑證是用於認證第二個 CA 金鑰的憑證，依此類推，直至到達自簽名的**根**憑證。因此，鏈中的每個證書 (第一個證書之後的證書) 都需要認證鏈中前一個證書的簽名者的公開金鑰。

# 關於安全套接字層

**安全套接字層** (SSL) 是用於確定網際網路通訊和作業事件保護的最常見標準。Web 應用程式使用 HTTPS (基於 SSL 的 HTTP)，HTTPS 使用數位證書來確保在伺服器和使用者端之間進行安全、機密的通訊。在 SSL 連線中，使用者端和伺服器在傳送資料之前都要對資料進行加密，然後由接受者對其進行解密。

Web 瀏覽器 (用戶端) 需要與某個安全站點建立連線時，則會發生 *SSL* **交換：**

- 瀏覽器將透過網路傳送請求安全階段作業的訊息 (通常請求爲以 `https` 開頭的 URL，而非 `http` 開頭的 URL)。
- 伺服器透過傳送其證書 (包括公開金鑰) 進行回應。
- 瀏覽器將驗證伺服器憑證是否有效，並驗證簽署該憑證的 CA，其憑證是否位於瀏覽器的資料庫中，並且是可信任的 CA。它還驗證 CA 證書是否已過期。
- 如果憑證有效，瀏覽器將產生一個一次性的、唯一的**階段作業金鑰**，並使用伺服器的公開金鑰對該階段作業進行加密。然後，瀏覽器將把加密的階段作業金鑰傳送給伺服器，這樣伺服器和瀏覽器都擁有該階段作業金鑰副本。
- 伺服器可以使用其私密金鑰對訊息進行解密，然後恢復該階段作業金鑰。

交握之後，即表示使用者端已驗證了網站的身份，並且只有該使用者端和 Web 伺服器擁有該階段作業金鑰副本。從現在開始，使用者端和伺服器便可以使用該階段作業金鑰對彼此間的所有通訊進行加密。這樣就確保了使用者端和伺服器之間的通訊的安全。

SSL 標準的最新版本稱爲 TLS (傳輸層安全性)。Application Server 支援安全套接字層 (SSL) 3.0 和傳輸層安全性 (TLS) 1.0 加密協定。

若要使用 SSL，Application Server 必須擁有接受安全連線的每個外部介面或 IP 位址的憑證。只有安裝了數位證書之後，大多數 Web 伺服器的 HTTPS 服務才能夠執行。請使用第 106 頁的「使用 keytool 公用程式產生憑證」中說明的程序來設定 Web 伺服器可以用於 SSL 的數位憑證。

## 關於加密算法

**加密算法**是用於加密或解密的加密演算法。SSL 和 TLS 協定支援用於伺服器和使用者端彼此進行認證、傳輸證書和建立階段作業金鑰的各種加密算法。

某些加密算法比其他加密算法更強大且更安全。使用者端和伺服器可以支援不同的密碼組。從 SSL3 和 TLS 協定中選取加密算法。在安全連線期間，使用者端和伺服器同意在通訊中使用它們均已啓用的最強大的加密算法，因此通常需要啓用所有加密算法。

## 使用基於名稱的虛擬主機

對安全應用程式使用基於名稱的虛擬主機可能會帶來問題。這是 SSL 協定本身的設計限制。必須先進行 SSL 交握 (使用者端瀏覽器在此時接受伺服器證書)，然後才能存取 HTTP 請求。這樣，在認證之前就無法確定包含虛擬主機名稱的請求資訊，因此也不能將多個證書指定給單一 IP 位址。

如果單一 IP 位址上的所有虛擬主機均需要對照同一證書進行認證，則增加多個虛擬主機將不會影響伺服器上正常的 SSL 作業。但是請注意，大多數瀏覽器會將伺服器的網域名稱與憑證中列出的網域名稱 (如果有，且主要適用於官方 CA 簽署憑證) 進行對照。如果網域名稱不匹配，這些瀏覽器將顯示警告。通常在生產環境中，只將基於位址的虛擬主機與 SSL 配合使用。

# 關於防火牆

**防火牆**控制兩個或多個網路之間的資料流，並管理網路之間的連結。防火牆可能包含硬體和軟體元素。本節描述了一些一般防火牆架構及其配置。此處的資訊主要針對 Application Server。如需有關特定防火牆技術的詳細資訊，請參閱防火牆供應商提供的文件。

通常，需要對防火牆進行配置，以便使用者端存取所需的 TCP/IP 連接埠。例如，如果 HTTP 偵聽程式正在連接埠 8080 上作業，則將防火牆配置爲僅允許處理連接埠 8080 上的 HTTP 請求。同樣地，如果爲連接埠 8181 設定了 HTTPS 請求，則必須將防火牆配置爲允許處理連接埠 8181 上的 HTTPS 請求。

如果需要從網際網路對 EJB 模組進行直接的 RMI-IIOP (全稱爲 Remote Method Invocations over Internet Inter-ORB Protocol [通過網際網路 ORB 交換協定的遠端方法呼叫]) 存取，則還需要開啓 RMI-IIOP 偵聽程式連接埠，但強烈建議您不要這樣做，因爲這樣可能會破壞安全性。

在雙防火牆架構中，您必須將外部防火牆配置爲允許處理 HTTP 和 HTTPS 作業事件。您必須將內部防火牆配置爲允許 HTTP 伺服器外掛程式與防火牆後面的 Application Server 進行通訊。

# 關於證書檔案

安裝 Application Server 時會產生一個適用於內部測試的 JSSE (Java Secure Socket Extension) 或 NSS (Network Security Services) 格式的數位憑證。依預設，Application Server 將其憑證資訊儲存在 *domain-dir* `/config` 目錄下的憑證資料庫中：

- **金鑰庫檔案** `key3.db` 包含 Application Server 的憑證 (包括其私密金鑰)。金鑰庫檔案受密碼保護。可使用 `asadmin change-master-password` 指令變更該密碼。如需有關 `certutil` 的更多資訊，請參閱第 108 頁的 「使用 `certutil` 公用程式」。

  每個金鑰存放區項目都有專屬別名。安裝後，Application Server 金鑰庫會有一個別名爲 `s1as` 的項目。

- **信任庫檔案** `cert8.db` 包含 Application Server 的可信任憑證 (包括其他實體的公開金鑰)。對於受信任的憑證，伺服器已確認憑證中的公開金鑰屬於憑證的所有者。受信任的證書通常包括那些憑證授權單位 (CA) 的證書。

  在開發者設定檔的伺服器端，Application Server 使用 JSSE 格式，該格式使用 `keytool` 管理憑證和金鑰庫。在叢集和企業設定檔的伺服器端，Application Server 使用 NSS 格式，該格式使用 `certutil` 管理儲存私密金鑰和憑證的 NSS 資料庫。在兩種設定檔中，用戶端 (應用程式用戶端或獨立用戶端) 均使用 JSSE 格式。

  依預設，已將 Application Server 配置爲具有金鑰庫和信任庫，它們將與範例應用程式配合使用並可用於開發目的。若要用於生產目的，您可能需要變更憑證別名、將其他憑證增加至信任庫或變更金鑰庫檔案和信任庫檔案的名稱和/或位置。

## 變更憑證檔案的位置

供開發使用的金鑰庫檔案和信任庫檔案儲存在 *domain-dir*`/config` 目錄中。

使用管理主控台以增加或修改憑證檔案之新位置的欄位值。

`-Dcom.sun.appserv.nss.db=${com.sun.aas.instanceRoot}/`*NSS-database-directory*

其中 *NSS-database-directory* 是 NSS 資料庫的位置。

# 使用 Java 安全套接字延伸 (JSSE) 工具

使用 `keytool` 可以設定和使用 JSSE (Java 安全套接字延伸) 數位憑證。在開發者設定檔的伺服器端，Application Server 使用 JSSE 格式管理憑證和金鑰庫。在所有設定檔中，用戶端 (應用程式用戶端或獨立用戶端) 均使用 JSSE 格式。

J2SE SDK 附帶有 `keytool`，可以讓管理員管理公開/私密金鑰對和關聯憑證。還可以讓使用者快取正與其通訊的另一方的公開金鑰 (以證書形式)。

若要執行 `keytool`，必須配置 shell 環境，以使 J2SE `/bin` 目錄位於路徑中，或者指令行中必須存在該工具的完整路徑。如需有關 `keytool` 的更多資訊，請參閱位於以下位置的

keytool 文件：http://java.sun.com/j2se/1.5.0/docs/tooldocs/solaris/keytool.html。

# 使用 keytool 公用程式

以下範例說明了與使用 JSSE 工具處理憑證相關的用法：

- 使用 RSA 金鑰演算法在類型爲 JKS 的金鑰庫中建立自我簽署憑證。RSA 是 RSA Data Security, Inc. 開發的公開金鑰加密技術。RSA 的首字母縮略分別代表該技術的三位發明者：Rivest、Shamir 和 Adelman。

  ```
  keytool  -genkey -noprompt -trustcacerts -keyalg RSA -alias ${cert.alias}
  -dname  ${dn.name} -keypass ${key.pass} -keystore ${keystore.file}
  -storepass ${keystore.pass}
  ```

  第 106 頁的「使用 keytool 公用程式產生憑證」中列出了建立憑證的另一個範例。
- 使用預設金鑰演算法在類型爲 JKS 的金鑰庫中建立自我簽署憑證。

  ```
  keytool -genkey -noprompt -trustcacerts -alias ${cert.alias} -dname
  ${dn.name} -keypass ${key.pass} -keystore ${keystore.file} -storepass
  ${keystore.pass}
  ```

  第 107 頁的「使用 keytool 公用程式簽署數位憑證」中列出了簽署憑證的範例。
- 顯示類型爲 JKS 的金鑰庫中的可用憑證。

  ```
  keytool -list -v  -keystore ${keystore.file} -storepass ${keystore.pass}
  ```
- 顯示類型爲 JKS 的金鑰庫中的憑證資訊。

  ```
  keytool -list -v  -alias ${cert.alias} -keystore ${keystore.file}
  -storepass ${keystore.pass}
  ```
- 將 RFC/文字格式的憑證匯入 JKS 庫中。憑證通常會以網際網路 RFC (註釋請求) 1421 標準定義的可列印編碼格式 (而非憑證的二進位編碼) 加以儲存。此憑證格式 (也稱爲 *Base 64* **編碼**) 便於透過電子郵件或某些其他機制，將憑證匯出至其他應用程式。

  ```
  keytool -import -noprompt -trustcacerts -alias ${cert.alias} -file
  ${cert.file} -keystore ${keystore.file} -storepass ${keystore.pass}
  ```
- 從類型爲 JKS 的金鑰庫中以 PKCS7 格式匯出憑證。公開金鑰加密標準 #7 (加密訊息語法標準) 定義的回覆格式除了包含已核發的憑證外，還包含支援憑證鏈。

  ```
  keytool -export -noprompt -alias ${cert.alias} -file ${cert.file}
  -keystore ${keystore.file} -storepass ${keystore.pass}
  ```
- 從類型爲 JKS 的金鑰庫中以 RFC/文字格式匯出憑證。

```
keytool -export -noprompt -rfc -alias ${cert.alias} -file
${cert.file} -keystore ${keystore.file} -storepass ${keystore.pass}
```

- 從類型爲 JKS 的金鑰庫中刪除憑證。

```
keytool -delete -noprompt -alias ${cert.alias}  -keystore ${keystore.file}
-storepass ${keystore.pass}
```

第 107 頁的 「使用 keytool 公用程式刪除憑證」中提供了從金鑰庫中刪除憑證的另一個範例。

# 使用 keytool 公用程式產生憑證

使用 keytool 產生、匯入和匯出憑證。依預設，keytool 將在其執行目錄中建立金鑰庫檔案。

1. 變更至要執行憑證的目錄。

   始終在包含金鑰庫檔案和信任庫檔案的目錄中產生憑證，依預設，該目錄爲 *domain-dir*/config。如需有關變更這些檔案位置的資訊，請參閱第 104 頁的 「變更憑證檔案的位置」。

2. 輸入以下 keytool 指令，以在金鑰庫檔案 keystore.jks 中產生憑證：

   ```
   keytool -genkey -alias keyAlias-keyalg RSA
    -keypass changeit
    -storepass changeit
   -keystore keystore.jks
   ```

   使用任何專屬名稱做爲 *keyAlias*。如果您已變更金鑰庫或私密金鑰密碼的預設值，請使用新密碼取代以上指令中的 changeit。預設金鑰密碼別名爲「s1as」。

   螢幕上將顯示提示，要求您提供姓名、組織和其他資訊，keytool 將使用這些資訊產生憑證。

3. 輸入以下 keytool 指令，以將產生的憑證匯出至檔案 server.cer (或 client.cer [如果您願意])：

   ```
   keytool -export -alias keyAlias-storepass changeit
    -file server.cer
    -keystore keystore.jks
   ```

4. 如需由憑證授權單位簽署的憑證，請參閱第 107 頁的 「使用 keytool 公用程式簽署數位憑證」。

5. 若要建立信任庫檔案 cacerts.jks，並將憑證增加至該信任庫，請輸入以下 keytool 指令：

   ```
   keytool -import -v -trustcacerts
   -alias keyAlias
   ```

```
 -file server.cer
-keystore cacerts.jks
 -keypass changeit
```

6. 如果您已變更金鑰庫或私密金鑰密碼的預設值，請使用新密碼取代以上指令中的 `changeit`。

   工具將顯示有關證書的資訊並提示您是否要信任該證書。

7. 鍵入 `yes`，然後按下 Enter 鍵。

   `keytool` 便會顯示如下資訊：

```
Certificate was added to keystore
[Saving cacerts.jks]
```

8. 重新啓動 Application Server。

# 使用 keytool 公用程式簽署數位憑證

建立數位憑證之後，所有者必須簽署該憑證以防僞造。電子商務站點或那些身份認證對其很重要的站點可以從知名的憑證授權機構 (CA) 購買憑證。如果無需考量認證 (例如當私密安全通訊可以滿足全部需求時)，則可節省獲取 CA 憑證所花費的時間和費用並使用自我簽署憑證。

1. 請依照 CA 網站上的說明產生憑證金鑰對。
2. 下載產生的證書金鑰對。

   將憑證儲存在包含金鑰庫檔案和信任庫檔案的目錄中，依預設，該目錄爲 *domain-dir*`/config`。請參閱第 104 頁的「變更憑證檔案的位置」。

3. 在 shell 中，變更至包含憑證的目錄。
4. 使用 `keytool` 將憑證匯入本機金鑰庫和本機信任庫 (如有必要)。

```
keytool -import -v -trustcacerts
-alias keyAlias
 -file server.cer
-keystore cacerts.jks
 -keypass changeit
-storepass changeit
```

   如果金鑰庫密碼或私密金鑰密碼不是預設密碼，請使用新密碼取代以上指令中的 `changeit`。

5. 重新啓動 Application Server。

# 使用 keytool 公用程式刪除憑證

若要刪除現有憑證，請使用 `keytool -delete` 指令，例如：

```
keytool -delete
 -alias keyAlias
 -keystore keystore-name
 -storepass password
```

# 使用 Network Security Services (NSS) 工具

在叢集和企業設定檔的伺服器端，使用 Network Security Services (NSS) 數位憑證管理儲存私密金鑰和憑證的資料庫。對於用戶端 (應用程式用戶端或獨立用戶端)，使用第 104 頁的 「使用 Java 安全套接字延伸 (JSSE) 工具」中所述的 JSSE 格式。

使用 Network Security Services (NSS) 管理安全性的工具包括：

- `certutil`，指令行公用程式，用於管理憑證和金鑰資料庫。第 108 頁的 「使用 certutil 公用程式」中提供了使用 `certutil` 公用程式的一些範例。
- `pk12util`，指令行公用程式，用於以 PKCS12 格式在憑證/金鑰資料庫和檔案之間匯入和匯出金鑰及憑證。第 110 頁的 「使用 pk12util 公用程式匯入和匯出憑證」中提供了使用 `pk12util` 公用程式的一些範例。
- `modutil`，指令行公用程式，用於管理 `secmod.db` 檔案或硬體記號中的 PKCS #11 模組資訊。第 111 頁的 「使用 modutil 增加和刪除 PKCS11 模組」中提供了使用 `modutil` 公用程式的一些範例。

這些工具位於 *as-install*`/lib/` 目錄中。以下環境變數用於指向 NSS 安全性工具的位置：

- `LD_LIBRARY_PATH =${`*as-install*`}/lib`
- `${`*os.nss.path*`}`

在這些範例中，憑證一般名稱 (CN) 是指用戶端或伺服器的名稱。CN 還用於在 SSL 交換過程中比較憑證名稱和產生憑證的主機名稱。如果憑證名稱和主機名稱不符，則在 SSL 交換過程中會產生警告或異常。在某些範例中，爲方便起見，使用憑證一般名稱 `CN=localhost`，以便所有使用者均可以使用該憑證，而不必使用其眞實主機名稱建立新憑證。

以下小節中的範例說明了與使用 NSS 工具處理憑證相關的用法：

- 第 108 頁的 「使用 certutil 公用程式」
- 第 110 頁的 「使用 pk12util 公用程式匯入和匯出憑證」
- 第 111 頁的 「使用 modutil 增加和刪除 PKCS11 模組」

## 使用 certutil 公用程式

執行 `certutil` 之前，請確定 `LD_LIBRARY_PATH` 指向執行此公用程式所需之程式庫的位置。這個位置可以從 `asenv.conf` (整個產品的配置檔案) 中 `AS_NSS_LIB` 的值加以識別。

憑證資料庫工具 certutil 是 NSS 指令行公用程式，可建立和修改 Netscape Communicator cert8.db 和 key3.db 資料庫檔案。該公用程式還可以列出、產生、修改或刪除 cert8.db 檔案中的憑證，以及建立或變更密碼、產生新的公開和私密金鑰對、顯示金鑰資料庫的內容或刪除 key3.db 檔案中的金鑰對。

金鑰和憑證管理程序通常以在金鑰資料庫中建立金鑰開始，然後在憑證資料庫中產生和管理憑證。以下文件論述了如何使用 NSS (包括 certutil 公用程式的語法) 管理憑證和金鑰資料
庫：http://www.mozilla.org/projects/security/pki/nss/tools/certutil.html。

以下清單中的每個項目均提供了使用 NSS 和 JSSE 安全性工具建立和/或管理憑證的範例。

- 產生自我簽署的伺服器和用戶端憑證。在此範例中，CN 必須爲 hostname.domain.[com|org|net|...] 格式。

  在此範例中，目錄爲 *domain-dir*/config。serverseed.txt 和 clientseed.txt 檔案可以包含任何隨機文字。此隨機文字將用於產生金鑰對。

  ```
  certutil -S -n $SERVER_CERT_NAME -x -t "u,u,u"
   -s "CN=$HOSTNAME.$HOSTDOMAIN, OU=Java Software, O=Sun Microsystems Inc.,
      L=Santa Clara, ST=CA, C=US"
   -m 25001 -o  $CERT_DB_DIR/Server.crt
   -d $CERT_DB_DIR -f passfile &lt;$CERT_UTIL_DIR/serverseed.txt
  ```

  產生用戶端憑證。此憑證也是自我簽署憑證。

  ```
  certutil -S -n $CLIENT_CERT_NAME  -x -t "u,u,u"
   -s "CN=MyClient, OU=Java Software, O=Sun Microsystems Inc.,
      L=Santa Clara, ST=CA, C=US"
   -m 25002 -o  $CERT_DB_DIR/Client.crt
   -d $CERT_DB_DIR -f passfile &lt;$CERT_UTIL_DIR/clientseed.txt
  ```

- 驗證在之前的分項符號中產生的憑證。

  ```
  certutil -V -u V -n $SERVER_CERT_NAME -d  $CERT_DB_DIR
  certutil -V -u C -n $CLIENT_CERT_NAME -d  $CERT_DB_DIR
  ```

- 顯示可用憑證。

  ```
  certutil -L -d  $CERT_DB_DIR
  ```

- 將 RFC/文字格式的憑證匯入 NSS 憑證資料庫中。

  ```
  certutil -A -a -n ${cert.nickname} -t ${cert.trust.options}
  -f ${pass.file} -i ${cert.rfc.file}
  -d ${admin.domain.dir}/${admin.domain}/config
  ```

- 以 RFC 格式從 NSS 憑證資料庫中匯出憑證。

```
certutil -L -a -n ${cert.nickname} -f ${pass.file}
-d ${admin.domain.dir}/${admin.domain}/config > cert.rfc
```

- 從 NSS 憑證資料庫中刪除憑證。

```
certutil -D -n ${cert.nickname} -f ${pass.file}
-d ${admin.domain.dir}/${admin.domain}/config
```

- 將 NSS 資料庫中的憑證轉換爲 JKS 格式

```
certutil -L -a -n ${cert.nickname}
-d ${admin.domain.dir}/${admin.domain}/config > cert.rfc
keytool  -import -noprompt -trustcacerts -keystore ${keystore.file}
-storepass ${keystore.pass} -alias ${cert.alias} -file cert.rfc
```

# 使用 pk12util 公用程式匯入和匯出憑證

pk12util 指令行公用程式用於以 PKCS12 格式在憑證/金鑰資料庫和檔案之間匯入與匯出金鑰和憑證。PKCS12 爲公開金鑰加密標準 (PKCS) #12，個人資訊交換語法標準。如需有關 pk12util 公用程式的說明，請至http://www.mozilla.org/projects/security/pki/nss/tools/pk12util.html。

- 將 PKCS12 格式的憑證匯入 NSS 憑證資料庫中。

```
pk12util -i ${cert.pkcs12.file} -k ${certdb.pass.file}
-w ${cert.pass.file} -d ${admin.domain.dir}/${admin.domain}/config
```

- 將 PKCS12 格式的憑證匯入 NSS 憑證資料庫記號模組中。

```
pk12util -i ${cert.pkcs12.file} -h ${token.name} -k ${certdb.pass.file}
-w ${cert.pass.file} -d ${admin.domain.dir}/${admin.domain}/config
```

- 以 PKCS12 格式從 NSS 憑證資料庫中匯出憑證。

```
pk12util -o -n ${cert.nickname} -k ${pass.file} -w${cert.pass.file}
-d ${admin.domain.dir}/${admin.domain}/config
```

- 以 PKCS12 格式從 NSS 憑證資料庫記號模組中匯出憑證 (對硬體加速功能配置有用)。

```
pk12util -o -n ${cert.nickname} -h ${token.name} -k ${pass.file}
-w ${cert.pass.file} -d ${admin.domain.dir}/${admin.domain}/config
```

- 將 PKCS12 憑證轉換爲 JKS 格式 (需要 Java 來源)：

```
&lt;target name="convert-pkcs12-to-jks" depends="init-common">
   &lt;delete file="${jks.file}" failonerror="false"/>
   &lt;java classname="com.sun.enterprise.security.KeyTool">
      &lt;arg line="-pkcs12"/>
```

```
        &lt;arg line="-pkcsFile ${pkcs12.file}"/>
        &lt;arg line="-pkcsKeyStorePass ${pkcs12.pass}"/>
        &lt;arg line="-pkcsKeyPass ${pkcs12.pass}"/>
        &lt;arg line="-jksFile ${jks.file}"/>
        &lt;arg line="-jksKeyStorePass ${jks.pass}"/>
        &lt;classpath>
           &lt;pathelement path="${s1as.classpath}"/>
           &lt;pathelement path="${env.JAVA_HOME}/jre/lib/jsse.jar"/>
        &lt;/classpath>
     &lt;/java>
  &lt;/target>
```

## 使用 modutil 增加和刪除 PKCS11 模組

**安全性模組資料庫工具** `modutil` 是指令行公用程式，用於管理 `secmod.db` 檔案或硬體記號中的 PKCS #11 (加密記號介面標準) 模組資訊。您可以使用此工具來增加和刪除 PKCS #11 模組、變更密碼、設定預設值、列出模組內容、啓用或停用槽、啓用或停用 FIPS-140-1 規範遵循，以及爲加密作業指定預設提供者。此工具還可以建立 `key3.db`、`cert7.db` 和 `secmod.db` 安全性資料庫檔案。如需有關此工具的更多資訊，請至 `http://www.mozilla.org/projects/security/pki/nss/tools/modutil.html`。

- 增加新的 PKCS11 模組或記號。

  ```
  modutil -add ${token.module.name} -nocertdb -force -mechanisms RSA:DSA:RC4:DES
  -libfile ${SCA.lib.path} -dbdir ${admin.domain.dir}/${admin.domain}/config
  ```

- 從 NSS 庫中刪除 PKCS11 模組。

  ```
  modutil -delete ${token.module.name} -nocertdb -force -mechanisms RSA:DSA:RC4:DES
  -libfile ${SCA.lib.path} -dbdir ${admin.domain.dir}/${admin.domain}/config
  ```

- 列出 NSS 庫中的可用記號模組。

  ```
  modutil -list  -dbdir ${admin.domain.dir}/${admin.domain}/config
  ```

# 將 Application Server 與硬體加密加速器搭配使用

您可以使用硬體加速器記號來改善加密效能，同時提供安全金鑰儲存功能。另外，您也可以經由智慧卡，爲一般使用者提供行動式安全金鑰儲存。

Sun Java System Application Server 支援使用 PKCS#11 記號進行 SSL 或 TLS 通訊，並支援使用 Network Security Services (NSS) 工具管理金鑰和 PKCS#11 記號。本小節說明 Application Server 如何提供上述支援，同時帶您逐步執行相關配置的程序。

J2SE 5.0 PKCS#11 提供者可以和 Application Server 執行階段輕鬆整合。藉由這些提供者，您可以在 Application Server 中使用硬體加速器和其他 PKCS#11 記號，以達到快速效能，並保護 SSL 或 TLS 通訊中原有的私密金鑰。

本小節包含下列主題：



# 關於配置硬體加密加速器

Sun Java System Application Server 已通過 Sun Crypto Accelerator 1000 (SCA-1000) 和 SCA-4000 測試。

Application Server 和 J2SE 5.0 一同使用時，可以和 PKCS#11 記號進行通訊。和 Application Server 一起封裝的是 NSS PKCS#11 記號程式庫 (適用於 NSS 內部 PKCS#11 模組，通常稱爲 NSS 軟體記號) 以及 NSS 指令行管理工具。如需更多詳細資訊，請參閱第 108 頁的 「使用 Network Security Services (NSS) 工具」。

使用 NSS 工具根據 PKCS#11 記號和 J2SE PKCS#11 提供者建立金鑰和憑證，以便在執行階段存取記號金鑰和憑證。PKCS#11 提供者爲加密服務的提供者，可做爲原生 PKCS#11 程式庫外圍的包裝程式。PKCS#11 記號通常是指含原生 PKCS#11 介面的所有硬體和軟體記號。硬體記號是以實體裝置 (如硬體加速器和智慧卡) 實作的 PKCS#11 記號。軟體記號則是完全以軟體實作的 PKCS#11 記號。

---

**備註 –** 如果您在 J2SE 1.4.x 平台上執行 Application Server，則僅支援一個 PKCS#11 記號，即 NSS 軟體記號。

---

在 Microsoft Windows 環境下，請將 NSS 程式庫 *AS_NSS* 和 NSS 工具目錄 *AS_NSS_BIN* 的位置，增加到 PATH 環境變數中。爲簡化說明，本小節所描述的程序僅使用 UNIX 指令。必要時請將 UNIX 變數替代爲 Windows 變數。

硬體加密加速器的配置可分爲兩個主要程序：



# 配置 PKCS#11 記號

本小節說明如何以 NSS 安全性工具 `modutil` 配置 PKCS#11 記號。請使用下列程序來配置 PKCS#11 記號。

輸入下列指令 (全部置於同一行)：

```
modutil -dbdir AS_NSS_DB -nocertdb -force -add moduleName -libfile
 absolute_path_of_pkcs11_library -mechanisms list_of_security_mechanisms
```

其中，*AS_NSS_DB* 是 NSS 資料庫目錄 (和您使用 Domain Administration Server (DAS) 時的 `AS_DOMAIN_CONFIG` 相同)

例如，若要配置硬體加速器記號，請輸入下列內容 (全部置於同一行)：

```
modutil -dbdir AS_NSS_DB -nocertdb -force -add "Sun Crypto Accelerator" -libfile
 /opt/SUNWconn/crypto/lib/libpkcs11.so -mechanisms RSA:DSA:RC4:DES
```

此範例中的硬體加速器爲 SCA–1000 加密加速器。依預設，對應的 PKCS#11 程式庫位於 `/opt/SUNWconn/crypto/lib/libpkcs11.so`。

`mechanisms` 必須是記號上可供使用之加密機制的完整清單。如果僅使用數個可用的加密機制，請參閱第 115 頁的 「配置 J2SE 5.0 PKCS#11 提供者」。如要取得所有支援機制的完整清單，請參閱 NSS 安全性工具網站上的 `modutil` 文件，網址是 `http://www.mozilla.org/projects/security/pki/nss/tools`。

以下範例假設在記號安裝期間所指定的記號名稱爲 `mytoken`。

爲驗證硬體加速器的配置是否正確，請輸入下列指令：

```
modutil -list -dbdir AS_NSS_DB
```

標準輸出如以下所示：

```
Using database directory /var/opt/SUNWappserver/domains/domain1/config ...

Listing of PKCS#11 Modules
-----------------------------------------------------------
  1. NSS Internal PKCS#11 Module
         slots: 2 slots attached
        status: loaded

         slot: NSS Internal Cryptographic Services
        token: NSS Generic Crypto Services

         slot: NSS User Private Key and Certificate Services
        token: NSS Certificate DB

  2. Sun Crypto Accelerator
        library name: /opt/SUNWconn/crypto/lib/libpkcs11.so
         slots: 1 slot attached
        status: loaded

         slot: Sun Crypto Accelerator:mytoken
        token: mytoken
-----------------------------------------------------------
```

# 管理金鑰和憑證

本小節說明數種使用 `certutil` 和 `pk12util` 來建立和管理金鑰與憑證的常用程序。如需有關 `certutil` 和 `pk12util` 的詳細資訊，請參閱第 108 頁的 「使用 Network Security Services (NSS) 工具」以及 NSS 安全性工具網站上的文件，網址是 `http://www.mozilla.org/projects/security/pki/nss/tools`。

---

**備註 –** 藉由在 `java.security` 特性檔案 (位於 Java 執行階段的 *JAVA_HOME*`/jre/lib/security` 目錄中) 中配置 PKCS#11 提供者，您還可以使用 J2SE `keytool` 公用程式來管理金鑰和憑證。如需使用 `keytool` 的詳細資訊以及「Java PKCS#11 Reference Guide」，請參閱 `http://java.sun.com/j2se/1.5.0/docs/guide/security/p11guide.html`。

---

本小節說明下列主題：

- 第 114 頁的 「列出金鑰和憑證」
- 第 115 頁的 「使用私密金鑰和憑證」

## 列出金鑰和憑證

- 若要列出已配置 PKCS#11 記號中的金鑰和憑證，請執行下列指令：

  ```
  certutil -L -d AS_NSS_DB [-h tokenname]
  ```

  例如，若要列出預設 NSS 軟體記號的內容，請鍵入：

  ```
  certutil -L -d AS_NSS_DB
  ```

  標準輸出和下列內容類似：

  ```
  verisignc1g1                 T,c,c
  verisignc1g2                 T,c,c
  verisignc1g3                 T,c,c
  verisignc2g3                 T,c,c
  verisignsecureserver         T,c,c
  verisignc2g1                 T,c,c
  verisignc2g2                 T,c,c
  verisignc3g1                 T,c,c
  verisignc3g2                 T,c,c
  verisignc3g3                 T,c,c
  s1as                         u,u,u
  ```

  在以上的輸出中，記號名稱顯示在左欄，三個一組的可信任屬性顯示在右欄。以 Application Server 憑證而言，通常是 `T,c,c`。不同於 J2SE `java.security.KeyStore` API 僅包含一個信任層級，NSS 技術包含數個信任層級。Application Server 主要注重於第一種信任屬性，該屬性說明此記號使用 SSL 的方式。對於此屬性：

T 代表憑證授權單位 (CA) 是可信任的用戶端憑證核發者。
u 代表您可以使用憑證 (和金鑰) 來進行認證或簽署。
屬性組合 u,u,u 代表資料庫中有私密金鑰。

- 若要列出硬體記號 mytoken 的內容，請執行下列指令：

```
certutil -L -d AS_NSS_DB -h mytoken
```

會提示您輸入硬體記號的密碼。標準輸出和下列內容類似：

```
Enter Password or Pin for "mytoken":
mytoken:Server-Cert                                          &#9;u,u,u
```

## 使用私密金鑰和憑證

使用 certutil 來建立自我簽署的憑證，並匯入或匯出憑證。若要匯入或匯出私密金鑰，請使用 pk12util 公用程式。如需更多詳細資訊，請參閱第 108 頁的 「使用 Network Security Services (NSS) 工具」。

**注意 –** 在 Application Server 中，請勿直接使用 NSS 工具 certutil 和 modutil 來修改 NSS 密碼。否則 Application Server 中的安全性資料可能會毀壞。

# 配置 J2SE 5.0 PKCS#11 提供者

Application Server 需藉由 J2SE PKCS#11 提供者，在執行階段存取位於 PKCS#11 記號中的金鑰和憑證。依預設，Application Server 會爲 NSS 軟體記號配置 J2SE PKCS#11 提供者。本小節說明如何置換 J2SE PKCS#11 提供者的預設配置。

在 Application Server 上，會爲每個 PKCS#11 記號產生下列預設的 PKCS#11 配置參數。

- 預設 NSS 軟體記號的配置：

```
name=internal
library=${com.sun.enterprise.nss.softokenLib}
nssArgs="configdir='${com.sun.appserv.nss.db}'
 certPrefix='' keyPrefix='' secmod='secmod.db'"
slot=2
omitInitialize = true
```

- SCA 1000 硬體加速器的配置：

```
name=HW1000
library=/opt/SUNWconn/crypto/lib/libpkcs11.so
slotListIndex=0
omitInitialize=true
```

這些配置符合「Java PKCS#11 Reference Guide」所描述的語法。

**備註 –** 只要求名稱參數必須爲唯一。J2SE 5.0 有些舊版本僅支援字母數字式的字元。

您可以建立自訂的配置檔案，以置換預設的配置參數。例如，您可以在 SCA–1000 中，明確停用 RSA 密碼和 RSA 金鑰對產生器。如需有關停用 RSA 密碼和 RSA 金鑰對產生器的詳細資訊，請參閱 `http://www.mozilla.org/projects/security/pki/nss/tools`。

建立自訂的配置檔案：

1. 使用下列程式碼建立名爲 *as-install*`/mypkcs11.cfg` 的配置檔案，然後儲存該檔案。

```
name=HW1000
library=/opt/SUNWconn/crypto/lib/libpkcs11.so
slotListIndex=0
disabledMechanisms = {
&#9;CKM_RSA_PKCS
&#9;CKM_RSA_PKCS_KEY_PAIR_GEN
}
omitInitialize=true
```

2. 如有必要，請更新 NSS 資料庫。在這個案例中，請更新 NSS 資料庫，這樣才能停用 RSA。

   執行下列指令：

```
modutil -undefault "Sun Crypto Accelerator" -dbdir AS_NSS_DB -mechanisms RSA
```

   `mechanisms` 清單上的演算法名稱，與預設配置中的演算法名稱不同。如需 NSS 中有效 `mechanisms` 的清單，請參閱 NSS 安全性工具網站上的 `modutil` 文件，網址是 `http://www.mozilla.org/projects/security/pki/nss/tools`。

3. 如下所示進行變更，在適當的位置增加特性以更新伺服器：

```
&lt;property name="mytoken" value="&InstallDir;/mypkcs11.cfg"/>
```

   特性的位置可能是下列其中一處：

   - 若提供者適用於 DAS 或伺服器實例，請將特性增加到相關聯的 `<security-service>` 之下。
   - 若提供者適用於節點代理程式，請將特性增加到 `domain.xml` 檔案中，相關聯的 `<node-agent>` 元素之下。

4. 重新啓動 Application Server。

   自訂配置將在重新啓動後生效。

◆ ◆ ◆ 第 10 章

# 10 配置訊息安全性

本章中的某些內容假設您對安全性和 Web 服務概念已有基本的瞭解。本章說明如何在 Application Server 中爲 Web 服務配置訊息層安全性。本章包含以下主題：



## 訊息安全性概況

在**訊息安全性**中，安全性資訊會插入到訊息中，以使其透過網路層傳輸並附帶訊息到達訊息目標。訊息安全性與傳輸層安全性 (在「Java EE 5.0 Tutorial」中的「Security」一章中討論) 不同，因爲訊息安全性可用於將訊息保護機制與訊息傳輸機制分離，從而使訊息在傳輸後仍保持受保護狀態。

Web 服務安全性：SOAP 訊息安全性 (WS-Security) 爲互通 Web 服務安全性的國際標準，是由 Web 服務技術的所有主要提供者 (包括 Sun Microsystems) 在 OASIS 中協作開發的。WS-Security 是一種訊息安全性機制，它使用 XML 加密和 XML 數位簽名來確保經由 SOAP 傳送的 Web 服務訊息的安全。WS-Security 規格定義了多種安全性記號 (包括 X.509 證書、SAML 指定和使用者名稱/密碼記號) 的使用，以認證和加密 SOAP Web 服務訊息。

您可透過 http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-soap-message-security-1.0.pdf 檢視 WS-Security 規格。

# 瞭解 Application Server 中的訊息安全性

Application Server 在其 Web 服務用戶端和伺服器端容器中提供對 WS-Security 標準的整合式支援。整合此功能後，Web 服務安全性可由代表應用程式之 Application Server 的容器來強制執行，並且可以用於保護任何 Web 服務應用程式，無需變更應用程式的實作。Application Server 透過提供可將 SOAP 層訊息安全性提供者和訊息保護策略連結到容器和容器中部署的應用程式之功能，來達到此效果。

## 指定訊息安全性職責

在 Application Server 中，第 118 頁的 「系統管理員」 和 第 118 頁的 「應用程式部署人員」 角色應負起配置訊息安全性的主要責任。儘管通常其他的角色都不需要開發者且無需變更其實作，就可以確保現有應用程式的安全，但在某些情況下，第 119 頁的 「應用程式開發人員」 也可以發揮作用。以下小節定義了各種角色的職責：



### 系統管理員

系統管理員負責：

- 在 Application Server 中配置訊息安全性提供者。
- 管理使用者資料庫。
- 管理金鑰儲存區和信任儲存區檔案。
- 在使用加密並執行 1.5.0 版之前的 Java SDK 時，配置 Java 加密擴展 (JCE) 提供者。
- 安裝範例伺服器。僅在 `xms` 範例應用程式用於說明如何使用訊息層 Web 服務安全性時，才執行此作業。

系統管理員使用管理主控台來管理伺服器安全性設定，並使用指令行工具來管理憑證資料庫。在 Platform Edition 中，憑證和私密金鑰均儲存在金鑰庫中，並透過 `keytool` 進行管理。Standard Edition 和 Enterprise Edition 將憑證和私密金鑰儲存在 NSS 資料庫中，並使用 `certutil` 對其進行管理。本文件主要適用於系統管理員。如需有關訊息安全性作業的簡介，請參閱第 122 頁的 「配置 Application Server 以實現訊息安全性」。

### 應用程式部署人員

應用程式部署人員負責：

- 如果上游角色 (開發者或組譯者) 尚未指定任何所需的特定於應用程式的訊息保護策略，則 (在應用程式組譯時) 指定這些策略。
- 修改特定於 Sun 的部署描述元，以向 Web 服務端點和服務參照指定特定於應用程式的訊息保護策略資訊 (即 message-security-binding 元素)。

### 應用程式開發人員

應用程式開發者可以啓用訊息安全性，但並不是必須要這樣做。訊息安全性可以由系統管理員設定，以便確保所有 Web 服務的安全；如果連結到應用程式的提供者或保護策略不同於連結到容器的提供者或保護策略，訊息安全性則由應用程式部署人員設定。

應用程式開發者或組譯者負責：

- 確定應用程式是否需要特定於應用程式的訊息保護策略。如果需要，則透過與應用程式部署人員交流，確保在應用程式組譯時指定所需策略。

## 關於安全性記號和安全性機制

WS-Security 規格爲使用安全性記號來認證和加密 SOAP Web 服務訊息提供了可延伸機制。與 Application Server 一起安裝的 SOAP 層訊息安全性提供者可用於採用使用者名稱/密碼和 X509 憑證安全性記號，以認證和加密 SOAP Web 服務訊息。採用其他安全性記號 (包括 SAML 指定) 的其他提供者將與 Application Server 的後續發行版本一起安裝。

### 關於使用者名稱記號

Application Server 使用 SOAP 訊息中的**使用者名稱記號**來建立訊息**傳送者**的認證身份。包含使用者名稱記號 (在內嵌式密碼中) 的訊息接收者透過確認傳送者是否知道使用者的秘密 (即密碼)，來驗證訊息傳送者是否經過授權成爲使用者 (在記號中識別)。

使用使用者名稱記號時，必須在 Application Server 中配置有效的使用者資料庫。

### 關於數位簽名

Application Server 使用 XML 數位簽名將認證身份連結到訊息**內容**。用戶端使用數位簽名來建立呼叫者身份，其方法與使用傳輸層安全性時用基本認證或 SSL 用戶端證書認證建立呼叫者身份的方法相似。訊息接收者將驗證數位簽名以認證訊息內容的來源 (可能與訊息傳送者不同)

使用數位簽名時，必須在 Application Server 中配置有效的金鑰庫和信任庫檔案。如需有關該主題的更多資訊，請參閱第 104 頁的 「關於證書檔案」。

### 關於加密

加密的目的是將資料修改爲只有目標讀者才能理解的形式。加密程序透過用加密元素替換原始內容來完成。與公開金鑰加密指出的那樣，可以使用加密來建立能夠閱讀訊息的一方或多方身份。

使用加密時，必須先安裝支援加密的 JCE 提供者。如需有關該主題的更多資訊，請參閱第 124 頁的 「配置 JCE 提供者」。

## 關於訊息保護策略

訊息保護策略是針對請求訊息處理和回應訊息處理而定義的，並根據對源和/或收信人認證的需求來進行表示。來源認證策略代表一個需求，即必須在訊息中建立已傳送訊息或已定義訊息內容的實體之身份，以使其可以由訊息收件者進行認證。接收者認證策略代表一個需求，即必須傳送訊息，以使可接收訊息的實體之身份可 由訊息傳送者建立。提供者套用特定的訊息安全性機制，以使訊息保護策略在 SOAP Web 服務訊息的上下文中實現。請求和回應訊息保護策略是在將提供者配置到容器時定義的。特定於應用程式的訊息保護策略 (細緻程度爲 Web 服務連接埠或作業) 也可以在應用程式或應用程式用戶端的特定於 Sun 的部署描述元中進行配置。在任何情況下，如果定義了訊息保護策略，則用戶端的請求和回應訊息保護策略必須與伺服器的請求和回應訊息保護策略相符 (即二者等效)。如需有關針對各應用程式定義訊息保護策略的更多資訊，請參閱「*Developers Guide*」中的「*Securing Applications*」一章。

# 訊息安全性術語字彙表

以下內容描述本文件中所使用的術語。第 122 頁的 「配置 Application Server 以實現訊息安全性」中也討論了這些概念。

- 認證層

  **認證層**是必須執行認證處理的訊息層。Application Server 在 SOAP 層強制執行 Web 服務訊息安全性。

- 認證提供者

  在 Application Server 的此發行版本中，Application Server 呼叫**認證提供者**來處理 SOAP 訊息層安全性。

  - **用戶端提供者**用於建立 (透過簽名或使用者名稱/密碼) 請求訊息的來源身份和/或保護 (透過加密) 請求訊息，從而使這些訊息僅可由其目標接收者檢視。用戶端提供者還可以將其容器建立爲接收的回應之授權收信人 (透過成功地解密)，並驗證回應中的密碼或簽名以認證與回應相關聯的源身份。在 Application Server 中配置的用戶端提供者可用來保護由做爲其他服務的用戶端的伺服器端元件 (即 Servlet 和 EJB 元件) 所傳送的請求訊息和所接收的回應訊息。
  - **伺服器端提供者**用於將其容器建立爲所接收請求的授權接收者 (透過成功地解密)，並驗證請求中的密碼或簽名以認證與該請求相關的來源身份。伺服器端提供者還將建立 (透過簽名或使用者名稱/密碼) 回應訊息的源身份和/或保護 (透過加密) 回應訊息，以使這些訊息只能由其目標收信人檢視。**伺服器端提供者**僅可由伺服器端容器呼叫。

- 預設伺服器提供者

  **預設伺服器提供者**用於爲尚未連結特定伺服器提供者的任何應用程式識別要呼叫的伺服器提供者。**預設伺服器提供者**有時被稱爲**預設提供者**。

- 預設用戶端提供者

  **預設用戶端提供者**用於爲尚未連結特定用戶端提供者的任何應用程式識別要呼叫的用戶端提供者。

- 請求策略

  **請求策略**定義與認證提供者執行的請求處理相關的認證策略需求。按照訊息寄件者的順序表示這些策略，從而使內容之後出現的加密請求意味著訊息收件者將在驗證簽名之前先要對訊息進行解密。

- 回應策略

  **回應策略**定義與認證提供者執行的回應處理相關的認證策略需求。按照訊息寄件者的順序表示這些策略，從而使內容之後出現的加密請求意味著訊息收件者將在驗證簽名之前先要對訊息進行解密。

# 確保 Web 服務的安全

透過將 SOAP 層訊息安全性提供者和訊息保護策略連結到部署有應用程式的容器或連結到應用程式提供的 Web 服務端點，來確保在 Application Server 中部署的 Web 服務的安全。透過將 SOAP 層訊息安全性提供者和訊息保護策略連結到用戶端容器或連結到由用戶端應用程式宣告的可攜式服務參照，從而在 Application Server 的用戶端容器中配置 SOAP 層訊息安全性功能。

安裝 Application Server 後，將在 Application Server 的用戶端和伺服器端容器中配置 SOAP 層訊息安全性提供者，其中這些提供者可由容器或者由容器中部署的個別應用程式或用戶端進行連結使用。在安裝程序中，這些提供者配置簡單的訊息保護策略，即如果連結到容器或者連結到容器中的應用程式或用戶端，則將導致所有請求和回應訊息中內容的源均由 XML 數位簽名進行認證。

可以採用 Application Server 的管理介面，從而連結現有提供者以供 Application Server 的伺服器端容器使用，修改由提供者強制執行的訊息保護策略，或使用替代的訊息保護策略來建立新的提供者配置。您可以在應用程式用戶端容器的 SOAP 訊息層安全性配置上執行類似的管理作業，如第 127 頁的「啓用應用程式用戶端的訊息安全性」中所定義。

依預設，Application Server 中的訊息層安全性處於停用狀態。若要配置 Application Server 的訊息層安全性，請執行第 122 頁的「配置 Application Server 以實現訊息安全性」中列出的步驟。如果您要將 Web 服務安全性用於保護在 Application Server 上部署的所有 Web 服務應用程式，請執行第 126 頁的「啓用訊息安全性的提供者」中的步驟。

完成以上步驟 (可能需要重新啓動 Application Server) 後，Web 服務安全性將套用於在 Application Server 上部署的所有 Web 服務應用程式。

## 配置特定於應用程式的 Web 服務安全性

透過在應用程式的特定於 Sun 的部署描述元中定義 message-security-binding 元素，來配置特定於應用程式的 Web 服務安全性功能 (在應用程式組譯時)。這些

message-security-binding 元素用於將特定提供者或訊息保護策略與 Web 服務端點或服務參照相關聯，並符合要求以使其套用到相應端點或參照的服務的特定連接埠或方法。

如需定義特定應用程式訊息防護策略的更多資訊，請參閱 「Sun Java System Application Server 9.1 Developer's Guide」中的第 5 章「Securing Applications」。

# 確保範例應用程式的安全

Application Server 隨附名爲 `xms` 的範例應用程式。`xms` 應用程式具有簡單的 Web 服務功能，可由 J2EE EJB 端點和 Java Servlet 端點共同實作。這兩個端點共用同一個服務端點介面。服務端點介面定義了單一作業 (`sayHello`)，此作業可取得字串引數，並傳回由呼叫引數加前置的 `Hello` 所組成的 `String`。

`xms` 範例應用程式用於說明如何使用 Application Server 的 WS-Security 功能來確保現有 Web 服務應用程式的安全。範例隨附的指令說明了如何啓用 Application Server 的 WS-Security 功能，以將其用於確保 `xms` 應用程式的安全。範例還說明了 WS-Security 功能與應用程式的直接連結 (如第 121 頁的 「配置特定於應用程式的 Web 服務安全性」中所述)。

`xms` 範例應用程式安裝在以下目錄中：*as-install*`/samples/webservices/security/ejb/apps/xms/`。

如需有關編譯、封裝和執行 `xms` 範例應用程式的資訊，請參閱「Developers' Guide」中的「Securing Applications」一章。

# 配置 Application Server 以實現訊息安全性



## 請求策略配置和回應策略配置的動作

下表列出了訊息保護策略配置，以及由該配置的 WS-Security SOAP 訊息安全性提供者所執行的最終訊息安全性作業。

表 10–1 訊息保護策略與 WS-Security SOAP 訊息安全性作業對映

| 訊息保護策略 | 最終的 WS-Security SOAP 訊息保護作業 |
|---|---|
| auth-source="sender" | 此訊息包含 `wsse:Security` 標頭，此標頭包含 `wsse:UsernameToken` (帶有密碼)。 |
| auth-source="content" | 對 SOAP 訊息內文的內容進行簽名。此訊息包含 `wsse:Security` 標頭，此標頭包含以 `ds:Signature`。 |
| auth-source="sender"<br>auth-recipient="before-content"<br>或<br>auth-recipient="after-content" | SOAP 訊息內文的內容已加密，並由最終的 `xend:EncryptedData` 替代。此訊息包含 `wsse:Security` 標頭，此標頭包含 `wsse:UsernameToken` (帶有密碼) 和 `xenc:EncryptedKey`。`xenc:EncryptedKey` 包含用於加密 SOAP 訊息內文的金鑰。此金鑰在收信人的公開金鑰中加密。 |
| auth-source="content"<br>auth-recipient="before-content" | SOAP 訊息內文的內容已加密，並由最終的 `xend:EncryptedData` 替代。`xenc:EncryptedData` 已簽名。此訊息包含 `wsse:Security` 標頭，此標頭包含 `xenc:EncryptedKey` 和 `ds:Signature`。`xenc:EncryptedKey` 包含用於加密 SOAP 訊息內文的金鑰。此金鑰在收信人的公開金鑰中加密。 |
| auth-source="content"<br>auth-recipient="after-content" | SOAP 訊息內文的內容已簽名和加密，並由最終的 `xend:EncryptedData` 替代。此訊息包含 `wsse:Security` 標頭，此標頭包含 `xenc:EncryptedKey` 和 `ds:Signature`。`xenc:EncryptedKey` 包含用於加密 SOAP 訊息內文的金鑰。此金鑰在收信人的公開金鑰中加密。 |
| auth-recipient="before-content"<br>或<br>auth-recipient="after-content" | SOAP 訊息內文的內容已加密，並由最終的 `xend:EncryptedData` 替代。此訊息包含 `wsse:Security` 標頭，此標頭包含 `xenc:EncryptedKey`。`xenc:EncryptedKey` 包含用於加密 SOAP 訊息內文的金鑰。此金鑰在收信人的公開金鑰中加密。 |
| 未指定策略。 | 模組未執行任何安全性作業。 |

# 配置其他安全性功能

Application Server 使用 SOAP 處理層中整合的訊息安全性提供者來實作訊息安全性。訊息安全性提供者取決於 Application Server 的其他安全性功能。

1. 如果使用的 Java SDK 版本早於 1.5.0，而且使用了加密技術，請配置 JCE 提供者。
2. 第 124 頁的 「配置 JCE 提供者」中討論了如何配置 JCE 提供者。
3. 如果使用了使用者名稱記號，請在必要時配置使用者資料庫。使用使用者名稱/密碼記號時，必須配置適當的範圍，並爲該範圍配置適當的使用者資料庫。

4. 管理證書和私密金鑰 (如有必要)。

### 完成之後

如果已將 Application Server 的功能配置爲供訊息安全性提供者使用，則可能會啓用隨 Application Server 一起安裝的提供者，如第 126 頁的「啓用訊息安全性的提供者」中所述。

## 配置 JCE 提供者

J2SE 1.4.x 中包含的 Java 加密延伸 (JCE) 提供者不支援RSA 加密。由於由 WS-Security 定義的 XML 加密通常是基於 RSA 加密，因此，若要使用 WS-Security 來加密 SOAP 訊息，您必須下載並安裝支援 RSA 加密的 JCE 提供者。

---

**備註 –** RSA 是 RSA Data Security, Inc. 開發的公開金鑰加密技術。RSA 的首字母縮略分別代表該技術的三位發明者：Rivest、Shamir 和 Adelman。

---

如果您是在 Java SDK 1.5 版中執行 Application Server，則 JCE 提供者已進行正確配置。如果您是在 Java SDK 1.4.x 版中執行 Application Server，請執行以下步驟，以靜態方式將 JCE 提供者增加爲 JDK 環境的一部分：

1. 下載並安裝 JCE 提供者 JAR (Java 歸檔) 檔案。

   透過以下 URL 可以獲得支援 RSA 加密的 JCE 提供者之清單：`http://java.sun.com/products/jce/javase_providers.html`。

2. 將 JCE 提供者 JAR 檔案複製到 *java-home*`/jre/lib/ext/`。

3. 停止 Application Server。

   如果未停止 Application Server 而後來又在該程序中將其重新啓動，則 Application Server 將無法識別 JCE 提供者。

4. 在任何一種文字編輯器中編輯 *java-home*`/jre/lib/security/java.security` 特性檔案。將剛下載的 JCE 提供者增加至此檔案。

   `java.security` 檔案包含用於增加該提供者的詳細說明。通常，您需要在具有類似特性的某個位置處增加一行，其格式如下：

   ```
   security.provider.n=provider-class-name
   ```

   在此範例中，*n* 是 Application Server 計算安全性提供者時所使用的喜好設定順序。將剛才增加的 JCE 提供者的 *n* 設定爲 2。

   例如，如果下載的是「The Legion of the Bouncy Castle JCE」提供者，則應增加以下行。

   ```
   security.provider.2=org.bouncycastle.jce.provider.
      BouncyCastleProvider
   ```

確定將 Sun 安全性提供者保持在最高的喜好設定順序，其值爲 1。

```
security.provider.1=sun.security.provider.Sun
```

將其他安全性提供者的層級調低，從而使每個層級上只有一個安全性提供者。

以下是提供必要 JCE 提供者並將現有提供者保持在正確位置的 `java.security` 檔案範例。

```
security.provider.1=sun.security.provider.Sun
security.provider.2=org.bouncycastle.jce.provider.
   BouncyCastleProvider
security.provider.3=com.sun.net.ssl.internal.ssl.Provider
security.provider.4=com.sun.rsajca.Provider
security.provider.5=com.sun.crypto.provider.SunJCE
security.provider.6=sun.security.jgss.SunProvider
```

5. 儲存並關閉檔案。
6. 重新啓動 Application Server。

# 訊息安全性設定

爲使用訊息安全性而對 Application Server 設定的大多數步驟都可以透過使用管理主控台、`asadmin` 指令行工具或透過手動編輯系統檔案來完成。通常，不建議編輯系統檔案，因爲它可能會做出無意識的變更而使 Application Server 無法正常工作，因此，如有可能，建議優先選擇使用管理主控台來配置 Application Server，然後選擇使用 `asadmin` 工具指令。僅在無管理主控台或等效的 `asadmin` 時，才手動編輯系統檔案。

訊息層安全性支援以 (可插接式) 認證模組的形式整合在 Application Server 及其用戶端容器中。依預設，Application Server 中的訊息層安全性處於停用狀態。以下小節提供用於啓用、建立、編輯和刪除訊息安全性配置和提供者的詳細資訊。


- 第 126 頁的 「啓用訊息安全性的提供者」
- 第 126 頁的 「配置訊息安全性提供者」
- 第 127 頁的 「建立訊息安全性提供者」
- 第 127 頁的 「啓用應用程式用戶端的訊息安全性」
- 第 127 頁的 「設定應用程式用戶端配置的請求策略和回應策略」
- 第 128 頁的 「詳細資訊」


大多數情況下，在執行以上管理作業後應重新啓動 Application Server。此步驟尤其適用於作業完成後，您要將管理變更的效果套用至 Application Server 上已部署應用程式中的情況。

# 啓用訊息安全性的提供者

若要爲在 Application Server 中部署的 Web 服務端點啓用訊息安全性，則必須指定要在伺服器端依預設使用的提供者。若要啓用訊息安全性的預設提供者，您還需要啓用由 Application Server 中部署的 Web 服務之用戶端所使用的提供者。第 127 頁的 「啓用應用程式用戶端的訊息安全性」中討論了有關啓用用戶端使用的提供者之資訊。

若要啓用源自已部署端點的 Web 服務呼叫之訊息安全性，您必須指定預設用戶端提供者。如果已爲 Application Server 啓用了預設用戶端提供者，則您必須確保在 Application Server 中部署的端點所呼叫的所有服務均配置爲與訊息層安全性相容。

使用指令行公用程式：

- 若要指定預設伺服器提供者，請使用：

```
asadmin  set --user  admin-user --port admin-port
server-config.security-service.message-security-config.SOAP.
default_provider=ServerProvider
```

- 若要指定預設用戶端提供者，請使用：

```
asadmin  set --user  admin-user --port admin-port
server-config.security-service.message-security-config.SOAP.
default_client_provider=ClientProvider
```

# 配置訊息安全性提供者

通常應重新配置提供者以修改其訊息保護策略 (儘管提供者類型、實作類別和提供者特定的配置特性可能也需要修改)。

使用指令行公用程式來設定回應策略，以 `response` 取代下列指令中的 `request`。

- 若要將請求策略增加到用戶端並設定認證來源，請使用：

```
asadmin  set --user  admin-user --port admin-port
server-config.security-service.message-security-config.SOAP.
provider-config.ClientProvider.request-policy.auth_source=
sender | content
```

- 若要將請求策略增加到伺服器並設定認證來源，請使用：

```
asadmin  set --user  admin-user --port admin-port
server-config.security-service.message-security-config.SOAP.
provider-config.ServerProvider.request-policy.auth_source=
sender | content
```

- 若要將請求策略增加到用戶端並設定認證收信人，請使用：

```
asadmin  set --user  admin-user --port admin-port
server-config.security-service.message-security-config.SOAP.
provider-config.ClientProvider.request-policy.auth_recipient=
before-content | after-content
```

- 若要將請求策略增加到伺服器並設定認證收信人，請使用：

```
asadmin  set --user  admin-user --port admin-port
server-config.security-service.message-security-config.SOAP.
provider-config.ServerProvider.request-policy.auth_recipient=
before-content | after-content
```

# 建立訊息安全性提供者

若要使用管理主控台配置現有的提供者，請選取 [配置] 節點 > 要配置的實例 > [安全性] 節點 > [訊息安全性] 節點 > [SOAP] 節點 > [提供者] 標籤。

如需有關建立訊息安全性提供者的詳細說明，請參閱管理主控台線上說明。

# 啓用應用程式用戶端的訊息安全性

必須配置用戶端提供者的訊息保護策略，以使其等效於將與其進行互動式操作的伺服器端提供者的訊息保護策略。在安裝 Application Server 時已配置 (但未啓用) 的提供者符合此情況。

若要啓用用戶端應用程式的訊息安全性，請修改特定於 Application Server 的應用程式用戶端容器配置。

# 設定應用程式用戶端配置的請求策略和回應策略

**請求策略和回應策略**定義與認證提供者執行的請求處理和回應處理相關的認證策略需求。按照訊息寄件者的順序表示這些策略，從而使內容之後出現的加密請求意味著訊息收件者將在驗證簽名之前先要對訊息進行解密。

若要獲得訊息安全性，必須同時在伺服器和用戶端中啓用請求策略和回應策略。在用戶端和伺服器中配置策略時，請確定應用程式層級訊息連結之請求/回應保護的用戶端策略與伺服器策略相匹配。

若要設定應用程式用戶端配置的請求策略，請依循第 127 頁的 「啓用應用程式用戶端的訊息安全性」中的說明，修改特定於 Application Server 的應用程式用戶端容器配置。在應用程式用戶端配置檔案中，增加以下所示的 `request-policy` 和 `response-policy` 元素，以設定請求策略。

提供的其他代碼可用做參照。其他代碼在安裝中可能略有不同。請勿變更這些代碼。

```
<client-container>
  <target-server name="your-host" address="your-host"
      port="your-port"/>
  <log-service file="" level="WARNING"/>
  <message-security-config auth-layer="SOAP"
      default-client-provider="ClientProvider">
    <provider-config
        class-name="com.sun.enterprise.security.jauth.ClientAuthModule"
        provider-id="ClientProvider" provider-type="client">
      <request-policy auth-source="sender | content"
        auth-recipient="after-content | before-content"/>
      <response-policy auth-source="sender | content"
        auth-recipient="after-content | before-content"/>
       <property name="security.config"
           value="as-install/lib/appclient/wss-client-config.xml"/>
    </provider-config>
  </message-security-config>
</client-container>
```

`auth-source` 的有效值包括 `sender` 和 `content`。`auth-recipient` 的有效值包括 `before-content` 和 `after-content`。第 122 頁的「請求策略配置和回應策略配置的動作」中提供了用於說明這些值的各種組合結果的表。

如果不想指定請求策略或回應策略，請將該元素保留爲空白，例如：

```
<response-policy/>
```

# 詳細資訊

- 您可透過以下 URL 檢視 Java 2 Standard Edition 的安全性討論：`http://java.sun.com/j2se/1.4.2/docs/guide/security/index.html`。
- 您可透過以下 URL 檢視「*Java EE 5.0 Tutorial*」中的「*Security*」一章：`http://java.sun.com/javaee/5/docs/tutorial/doc/index.html`。
- 「Administration Guide」中的「Configuring Security」一章。
- 「Developer's Guide」中的「Securing Applications」一章。
- 您可透過以下 URL 檢視「Oasis Web Services Security: SOAP Message Security (WS-Security)」規格：http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-soap-message-security-1.0.pdf。
- 您可透過以下 URL 檢視 OASIS Web Services Security Username Token Profile 1.0：http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-username-token-profile-1.0.pdf。
- 您可透過以下 URL 檢視 OASIS Web Services Security X.509 Certificate Token Profile 1.0：http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-x509-token-profile-1.0.pdf。

- 您可透過以下 URL 檢視「XML-Signature Syntax and Processing」文件：http://www.w3.org/TR/xmldsig-core/。
- 您可透過以下 URL 檢視「XML Encryption Syntax and Processing」文件：http://www.w3.org/TR/xmlenc-core/。

◆ ◆ ◆ 第 11 章

# 11 配置診斷服務

診斷服務可讓您更容易地查看和控制伺服器及其應用程式的執行階段效能，藉此診斷及隔離發生的錯誤。

本章包括下列小節：



## 診斷架構是什麼？

Application Server 診斷架構是一種監視架構，其可定義並實作一組在 Application Server 標準生命週期中執行的服務。使用診斷服務後，即可針對執行中伺服器及伺服器部署之應用程式產生的診斷資料，加以定義、建立、收集和存取。

## 診斷服務架構

診斷服務會報告 Application Server 實例的配置詳細資訊。診斷 Application Server 問題 (例如異常、效能問題和其他未預期的結果) 是很有用的。在管理主控台診斷服務中，您可以：

- **計算總和檢查**：收集 `appserver_install_dir/ lib`、`appserver_install_dir/etc` 和 `appserver_install_dir/bin` 目錄下選擇性 Application Server 二進位檔的總和檢查
- **驗證配置**：擷取配置檔案，例如 `domain.xml`、`server.policy`。
- **擷取安裝記錄**：特定安裝的詳細資訊，例如 Application Server 版本編號，或修補程式 ID 和安裝時所產生的記錄檔內容。當同一台機器中有多個安裝時，可使用安裝目錄的絕對路徑來決定收集哪個安裝程式記錄檔。會從 DAS 的安裝資料夾以及節點代理程式複製 `config/asenv.conf` 的內容。

  只會針對檔案型安裝來收集特定安裝的詳細資訊。
- **擷取系統資訊**：預設會收集下列系統資訊：

  - 網路設定
  - 作業系統詳細資訊
  - 硬體資訊

  Application Server 的平台版本上沒有使用原生碼收集的資料。
- **擷取應用程式部署描述元**：如 `ejb-jar.xml`、`sun-ejb-jar.xml`、`web.xml`、`sun-web.xml` 等部署描述元
- 記錄層級：
- 記錄項目：

  所產生診斷報告中需包含的記錄項目數。

# 產生診斷報告

依據您在「管理主控台」的 [Application Server 診斷] 標籤中所設定的喜好設定，來產生診斷報告。機密資料出現在「機密特性」表格所列出的產生報告中。

◆ ◆ ◆ 第 12 章

# 12 作業事件

透過將一個或多個步驟納入不可分的工作單元，作業事件可確保資料的完整性和一致性。本章包括下列小節：



## 關於作業事件



### 何為作業事件？

作業事件是應用程式中一系列嚴密的動作，所有動作必須成功完成，否則每個動作中的所有變更會被撤消。例如，將資金從支票帳戶轉入儲蓄帳戶是一項作業事件，步驟如下：

1. 檢查支票帳戶是否有足夠的資金來支付此轉帳操作。
2. 如果支票帳號中有足夠的資金，則將該筆資金記入此帳號的借方。
3. 將這些資金記入儲蓄帳戶的貸方。
4. 將此次轉帳記錄到支票帳戶記錄中。
5. 將此次轉帳記錄到儲蓄帳戶記錄中。

如果這些步驟的任何一個步驟失敗，則必須撤消在前面的步驟中所做的所有變更，而且支票帳戶和儲蓄帳戶的狀態必須與它們在作業事件開始之前的狀態相同。該事件稱爲**回復**。如果所有步驟均成功完成，則該作業事件處於**已確定**狀態。作業事件以確定或轉返狀態結束。

另請參閱：

- 第 134 頁的 「J2EE 技術中的作業事件」
- 第 135 頁的 「配置作業事件」


## J2EE 技術中的作業事件

J2EE 技術中的作業事件處理包括以下五個參與者：

- 作業事件管理員
- Application Server
- 資源管理員
- 資源配接卡
- 使用者應用程式

透過實作不同的 API 和功能，每個實體均有助於提高作業事件處理的可靠性，如下所述：

- 作業事件管理員提供支援作業事件分隔、作業事件資源管理、同步化及作業事件上下文傳遞所需的服務和管理功能。
- Application Server 提供支援應用程式執行階段環境 (包括作業事件狀態管理) 所需的基礎架構。
- 資源管理員 (透過資源配接卡) 提供應用程式對資源的存取權。資源管理員加入分散式作業事件的方法，是實作由作業事件管理員使用的作業事件資源介面，以針對作業事件關聯、作業事件完成以及回復工作進行通知。 關聯式資料庫伺服器便是這樣一個資源管理員。
- 資源配接卡是一個系統層級的軟體程式庫，應用程式伺服器或用戶端可使用該程式庫連線到資源管理員。不同資源管理員通常有專屬的資源配接卡。它可以做爲程式庫，在使用它的用戶端位址空間中使用。JDBC 驅動程式便是此類資源配接卡的一個範例。
- 開發用於應用程式伺服器環境的作業事件使用者應用程式使用 JNDI 來查找作業事件資料源及作業事件管理員 (可選)。應用程式可以使用企業 Bean 的宣告性作業事件屬性設定或明確的程式化作業事件分隔。

另請參閱：


- 第 133 頁的 「何爲作業事件？」
- 第 135 頁的 「配置作業事件」


## 特定資料庫的解決方法

Application Server 藉由回復下列 JDBC 驅動程式的實作，提供一些已知問題的解決方法。除非明確停用，否則會使用這些解決方法。

- Oracle thin 驅動程式 - 不論輸入旗標爲何，`XAResource.recover` 方法都會重複傳回有問題的相同 Xids 集。根據 XA 規格，作業事件管理程式最初使用 TMSTARTSCAN 呼叫此方法，然後使用 TMNOFLAGS 重複呼叫，直到不傳回 Xids 爲止。`XAResource.commit` 方法也有一些問題。

  若要停用 Application Server 解決方法，請將 `oracle-xa-recovery-workaround` 特性值設定爲 `false`。如需有關如何設定特性的詳細資訊，請參閱第 135 頁的 「配置 Application Server 如何從作業事件中回復 」。

**備註 –** 這些解決方法並不意味能支援任何特定的 JDBC 驅動程式。

# 有關作業事件的管理主控台作業

Application Server 根據管理主控台中的設定來處理作業事件。

## 配置作業事件

本小節說明如何配置以下作業事件設定：


- 第 135 頁的 「配置 Application Server 如何從作業事件中回復 」
- 第 136 頁的 「設定作業事件逾時值」
- 第 137 頁的 「設定作業事件記錄的位置」
- 第 137 頁的 「設定關鍵點間隔」


如需有關作業事件的其他資訊，請參閱以下小節：


- 第 133 頁的 「何爲作業事件？」
- 第 134 頁的 「J2EE 技術中的作業事件」


### ▼ 配置 Application Server 如何從作業事件中回復

由於伺服器當機或資源管理員當機，作業事件可能未完成。完成這些中斷的作業事件並將其從故障中恢復至關重要。Application Server 可在伺服器啓動時從這些故障中回復並完成作業事件。

執行恢復作業時，如果無法訪問某些資源，則伺服器重新啓動作業可能被延遲，因爲伺服器正在嘗試恢復作業事件。

如果作業事件跨伺服器進行，啓動此作業事件的伺服器會連絡其他伺服器以獲得作業事件的結果。如果無法訪問其他伺服器，則該作業事件將使用 [啓發式決策] 欄位來確定結果。

1 **在樹形元件中，選取 [配置] 節點。**

2 選取要配置的實例：

- 若要配置特定的實例，請選取該實例的配置節點。例如，對於預設實例 server，請選取 [server-config] 節點。
- 若要配置所有實例的預設設定，請選取 [default-config] 節點。

3 選取 [作業事件服務] 節點。

4 若要回復未完成的作業事件，請在 [重新啓動時] 欄位中核取 [回復]。

5 在 [重試逾時] 欄位中，設定 Application Server 嘗試連線無法存取的伺服器的時間 (以秒為單位)。預設值為 10 分鐘 (600 秒)。

6 在 [啓發式決策] 欄位中，為作業事件中無法訪問的伺服器設定策略。
除非有充分的理由將 [啓發式決策] 欄位設定爲 [確定]，否則將其保留設定爲 [轉返]。確定不確定的作業事件會破壞應用程式的資料完整性。

7 設定任何需要的特性。
按一下 [增加特性] 按鈕，在 [名稱] 和 [值] 欄位中鍵入值，並核取 [名稱] 左側的方塊來啓動該特性。

8 按一下 [儲存]。

9 重新啓動 Application Server。

## ▼ 設定作業事件逾時值

依預設，伺服器不會使作業事件逾時。也就是說，伺服器無限期地等待作業事件完成。在爲作業事件設定了逾時值後，如果作業事件在配置的時間內未完成，則 Application Server 將回復該作業事件。

1 在樹形元件中，選取 [配置] 節點。

2 選取要配置的實例：

- 若要配置特定的實例，請選取該實例的配置節點。例如，對於預設實例 server，請選取 [server-config] 節點。
- 若要配置所有實例的預設設定，請選取 [default-config] 節點。

3 選取 [作業事件服務] 節點。

4. **在 [作業事件逾時] 欄位中，輸入作業事件逾時之前等待的秒數。**
   作業事件逾時的預設值爲 0 秒。這將停用作業事件逾時。
5. **按一下 [儲存]。**
6. **重新啓動 Application Server。**

## ▼ 設定作業事件記錄的位置

爲了保持被呼叫資源的資料完整性，同時爲了能夠從故障中恢復，作業事件記錄將記錄有關每個作業事件的資訊。作業事件記錄儲存在 [作業事件記錄位置] 欄位所指定目錄的 `tx` 子目錄中。這些記錄無法進行人爲讀取。

1. **在樹形元件中，選取 [配置]s 節點。**
2. **選取要配置的實例：**
   - **若要配置特定的實例，請選取該實例的配置節點。例如，對於預設實例 `server`，請選取 [`server-config`] 節點。**
   - **若要配置所有實例的預設設定，請選取 [`default-config`] 節點。**
3. **選取 [作業事件服務] 節點。**
4. **在 [作業事件記錄位置] 欄位中輸入作業事件記錄的位置。**
   將建立 `tx` 子目錄，而且作業事件記錄儲存在該目錄下。

   預設值爲 `${com.sun.aas.instanceRoot}/logs`。`${com.sun.aas.instanceRoot}` 變數是實例的名稱，該變數在啓動 Application Server 實例時設定。若要查看 `${com.sun.aas.instanceRoot}` 的值，請按一下 [實際值]。
5. **按一下 [儲存]。**
6. **重新啓動 Application Server。**

## ▼ 設定關鍵點間隔

關鍵點作業可壓縮作業事件記錄檔。關鍵點間隔是指記錄上關鍵點作業之間的作業事件數目。關鍵點作業可以減小作業事件記錄檔的大小。關鍵點間隔數越大 (例如，2048)，作業事件記錄檔也越大，但關鍵點作業較少，效能可能更佳。關鍵點間隔越小 (例如，256)，記錄檔也越小，而同時由於關鍵點作業較爲頻繁，效能會略微降低。

1. **在樹形元件中，選取 [配置]s 節點。**

2. 選取要配置的實例：
   - 若要配置特定的實例，請選取該實例的配置節點。例如，對於預設實例 `server`，請選取 [`server-config`] 節點。
   - 若要配置所有實例的預設設定，請選取 [`default-config`] 節點。
3. 選取 [作業事件服務] 節點。
4. 在 [關鍵點間隔] 欄位中，輸入關鍵點作業之間的作業事件數目。
   預設值爲 2048。
5. 按一下 [儲存]。
6. 重新啓動 Application Server。

◆ ◆ ◆ 第 13 章

# 13 配置 HTTP 服務

HTTP 服務是 Application Server 的元件，提供用於部署 Web 應用程式和使 HTTP 用戶端能夠存取所部署 Web 應用程式的功能。這些工具通過兩種相關物件提供，即虛擬伺服器和 HTTP 偵聽程式。

本章說明以下主題：



## 虛擬伺服器

虛擬伺服器 (有時稱爲虛擬主機) 是一種物件，允許同一實體伺服器託管多個網際網路網域名稱。同一個實體伺服器上託管的所有虛擬伺服器共用該實體伺服器的網際網路通訊協定 (IP) 位址。虛擬伺服器將某個伺服器的網域名稱 (例如 `www.aaa.com`) 與執行 Application Server 的特定伺服器相關聯。

---

**備註 –** 請勿將網際網路網域與 Application Server 的管理網域混淆。

---

例如，假設您要在實體伺服器上託管以下網域：

```
www.aaa.com
www.bbb.com
www.ccc.com
```

同時假設 `www.aaa.com`、`www.bbb.com` 和 `www.ccc.com` 都分別具有與之關聯的 Web 模組 `web1`、`web2` 和 `web3`。

這意味著以下 URL 將全部由您的實體伺服器處理：

```
http://www.aaa.com:8080/web1
http://www.bbb.com:8080/web2
http://www.ccc.com:8080/web3
```

第一個 URL 將對映到虛擬主機 `www.aaa.com`，第二個 URL 將對映到虛擬主機 `www.bbb.com`，第三個 URL 將對映到虛擬主機 `www.ccc.com`。

另一方面，由於未在 `www.bbb.com` 註冊 `web3`，以下 URL 將導致 404 回覆碼：

```
http://www.bbb.com:8080/web3
```

若要使此對映有效，請確保 `www.aaa.com`、`www.bbb.com` 和 `www.ccc.com` 均可解析爲實體伺服器的 IP 位址。這些網域名稱需要在您的網路的 DNS 伺服器中註冊。此外，在 UNIX 系統上，應將這些網域增加到 `/etc/hosts` 檔案中 (如果 `/etc/nsswitch.conf` 檔案中的 `hosts` 設定包括 `files`)。

啓動 Application Server 後，將自動啓動以下虛擬伺服器：

- 名爲 `server` 的虛擬伺服器，用於託管所有使用者定義的 Web 模組
- 名爲 `__asadmin` 的虛擬伺服器，用於託管所有與管理相關的 Web 模組 (特別是管理主控台)。該伺服器是一個受限制的伺服器，您不能將 Web 模組部署到該虛擬伺服器上。

如果是在非生產環境中開發、測試和部署 Web 服務，通常只需要使用 `server` 虛擬伺服器。在生產環境中，其他虛擬伺服器可爲使用者和用戶提供主控功能，這樣，儘管只有一個實體伺服器，但每個使用者和用戶都好像有自己的 Web 伺服器一樣。

# HTTP 偵聽程式

每個虛擬伺服器都透過一個或多個 HTTP 偵聽程式，來提供伺服器與用戶端之間的連線。每個 HTTP 偵聽程式都是具有 IP 位址、連接埠號碼、伺服器名稱以及預設虛擬伺服器的偵聽插槽。

HTTP 偵聽程式必須具有唯一的連接埠號碼和 IP 位址組合。例如，透過將 IP 位址指定爲 0.0.0.0，HTTP 偵聽程式可以在機器的給定連接埠上偵聽所有已配置的 IP 位址。或者，HTTP 偵聽程式可以爲每個偵聽程式指定唯一的 IP 位址，但使用相同的連接埠。

由於 HTTP 偵聽程式是 IP 位址和連接埠號碼的組合，因此您可以擁有多個 IP 位址相同但連接埠號碼不同 (例如 1.1.1.1:8081 和 1.1.1.1:8082) 的 HTTP 偵聽程式，或 IP 位址不同但連接埠號碼相同 (例如 1.1.1.1:8081 和 1.2.3.4:8081) 的 HTTP 偵聽程式 (如果已將機器配置爲可以回應這些位址)。

不過，如果 HTTP 偵聽程式使用 0.0.0.0 IP 位址 (偵聽某個連接埠上的所有 IP 位址)，您便無法建立其他 IP 位址的 HTTP 偵聽程式 (偵聽特定 IP 位址的同一連接埠)。例如，如果 HTTP 偵聽程式使用 0.0.0.0:8080 (連接埠 8080 上的所有 IP 位址)，則其他 HTTP 偵聽程式不能使用 1.2.3.4:8080。

由於執行 Application Server 的系統通常只能存取一個 IP 位址，因此 HTTP 偵聽程式通常使用 0.0.0.0 IP 位址和不同的連接埠號碼，其中每個連接埠號碼用於不同目的。如果系統可以存取多個 IP 位址，則每個位址均可用於不同目的。

依預設，Application Server 啓動時，它具有以下 HTTP 偵聽程式：

- 兩個分別名爲 `http-listener-1` 和 `http-listener-2` 的 HTTP 偵聽程式，這兩個偵聽程式與名爲 `server` 的虛擬伺服器相關聯。名爲 `http-listener-1` 的偵聽程式未啓用安全性；而 `http-listener-2` 已啓用了安全性。
- 名爲 `admin-listener` 的 HTTP 偵聽程式，該偵聽程式與名爲 `__asadmin` 的虛擬伺服器相關聯。此偵聽程式已未啓用安全性。

所有這些偵聽程式均使用 IP 位址 0.0.0.0，以及在安裝 Application Server 期間指定爲 HTTP 伺服器連接埠號碼的連接埠號碼。如果 Application Server 使用預設連接埠號碼值，則 `http-listener-1` 使用連接埠 8080、`http-listener-2` 使用連接埠 8181、`admin-listener` 使用連接埠 48489。

每個 HTTP 偵聽程式均有一個預設虛擬伺服器。當請求 URL 的主機元件與 HTTP 偵聽程式關聯的所有虛擬伺服器 (在虛擬伺服器的 `http-listeners` 屬性中列出 HTTP 偵聽程式，即可將虛擬伺服器與該 HTTP 偵聽程式關聯起來) 均不相符時，HTTP 偵聽程式會將所有請求 URL 路由至預設虛擬伺服器。

此外，還應在 HTTP 偵聽程式中指定接收器執行緒的數目。接收器執行緒就是等待連線的執行緒。執行緒接受連線並將其放入佇列 (稱爲連線佇列) 中，在佇列中將由工作者執行緒接受這些連線。配置足夠多的接收器執行緒，以便在發生新的請求時總有一個可用，但又需要數目相當少，以免給系統造成太重負擔。在 Application Server 中，接收器與請求處理 (工作者) 執行緒之間沒有區別。每一個 HTTP 偵聽程式執行緒均負責接受與處理請求。因此，Application Server 預設配置中的 HTTP 偵聽程式會使用 50 個接收器執行緒。連線佇列既包括接收器執行緒剛剛接受的新連線，又包括持續作用連線管理子系統管理的永久性連線。

一組請求處理執行緒將從連線佇列中擷取內送的 HTTP 請求並對其進行處理。這些執行緒將剖析 HTTP 標頭，選取適當的虛擬伺服器並透過請求處理引擎處理請求。當沒有更多要處理的請求，但可以保持永久性連線 (透過使用 `HTTP/1.1` 或傳送 `Connection: keep-alive` 標頭) 時，請求處理執行緒會假定連線處於閒置狀態，並將連線傳送給持續作用連線管理子系統。

持續作用子系統會定期輪詢此類閒置連線，並將使用中的連線列入連線佇列中，以便將來進行處理。請求處理執行緒將再次從連線佇列中擷取連線並處理其請求。持續作用子系統是多執行緒的，可以管理大約數萬個連線。透過將大量連線分成較小的子集，使用有效的輪詢技術來確定哪些連線已就緒並具有請求，以及哪些連線由於處於閒置狀態的時間較長而被視爲已關閉 (超過允許的持續作用逾時的最大值)。

HTTP 偵聽程式的伺服器名稱即爲重新導向期間由伺服器傳送給用戶端的 URL 中顯示的主機名稱。此屬性會影響伺服器自動產生的 URL；但不會影響儲存在伺服器中目錄和檔案的 URL。如果伺服器使用一個別名，則該名稱應爲此別名。如果用戶端傳送 `Host:` 標頭，則在重新導向中該主機名稱將取代 HTTP 偵聽程式的伺服器名稱值。

要使用不同於原始請求中指定連接埠號的連接埠號碼，請指定重新導向連接埠。如果發生以下某種情況之一，則會進行**重新導向：**

- 如果用戶端嘗試存取已不存在於指定 URL 處的資源 (即該資源已移至其他位置)，伺服器將傳回一個指定的回應碼，並在回應的位置標頭中包含新的位置，從而將用戶端重新導向至新位置 (而不是傳回 404)。
- 如果用戶端嘗試透過常規 HTTP 連接埠存取受保護的資源 (例如 SSL)，則伺服器會將此請求重新導向至啓用了 SSL 的連接埠上。在此情況下，伺服器將在位置回應標頭中傳回一個新的 URL，其中的原始非安全連接埠將由啓用 SSL 的連接埠所取代。用戶端隨後會連線到這個新的 URL。

此外，還應指定是否爲 HTTP 偵聽程式啓用安全性以及使用哪種類型的安全性 (例如使用哪個 SSL 協定以及哪些密碼)。

若要存取部署在 Application Server 上的 Web 應用程式，請使用 URL `http://localhost:8080/` (或者，如果是安全應用程式，則使用 `https://localhost:8181/`) 和爲此 Web 應用程式指定的環境根目錄。若要存取管理主控台，請使用 URL `https://localhost:4848/` 或 `http://localhost:4848/asadmin/` (其預設環境根目錄)。

由於虛擬伺服器必須指定一個現有的 HTTP 偵聽程式，並且不能指定其他虛擬伺服器已使用的 HTTP 偵聽程式，因此在建立新虛擬伺服器之前，應至少建立一個 HTTP 偵聽程式。

◆ ◆ ◆ 第 14 章

# 14 管理 Web 服務

本章說明如何使用 Application Server 執行 Web 服務管理。管理主控台 和 `asadmin` 工具可讓您部署、測試和管理 Web 服務。您可以快速地視覺化、瞭解、監視與管理複雜的 Web 服務。您可以查看網域中部署的所有 Web 服務，就像查看 Java EE 應用程式和應用程式元件 (如 EJB) 一樣。

您也可以：

- 即時追蹤與繪製 Web 服務的回應時間和呼叫計數。
- 在達到邊界條件時產生警示，包括回應時間失敗和流量失敗。
- 檢視 XML 中的 Web 服務呼叫內容。
- 在執行階段使用 XSLT 變換訊息。

本章包含以下主題：



## Web 服務簡介

Web 服務是用戶端以 XML 型通訊協定 (如簡易物件存取協定 (SOAP)) 存取的應用程式，並且是透過網際網路協定 (如 HTTP) 傳送。用戶端可透過 Web 服務應用程式的介面和連結存取此應用程式。定義介面與連結時需採用 XML 工件，如 Web 服務定義語言 (WSDL) 檔案。

可延伸標記語言 (XML) 是全球資訊網協會 (W3C) 所開發的標準，並且是用來建立 Web 服務的基礎之一。XML 可讓 Web 服務與用戶端使用共用語言相互通訊。XML 是簡單、靈活、以文字爲基礎的標記語言。標記 XML 資料時需使用以角括弧包圍的標籤。標籤包含了所標記資料的意義。此類標記可讓不同的系統彼此之間輕鬆交換資料。

文件類型定義 (DTD) 或 XML 模式定義 (XSD) 則負責描述 XML 文件的結構。內容是對應 XML 文件中標籤的相關資訊、標籤順序等等。

XSLT 的全名是「可延伸樣式表語言轉換」，可將 XML 文件從一種格式轉換成另一種格式。

## Web 服務標準

簡易物件存取協定 (SOAP) 提供了 Web 服務的共用訊息傳送格式。SOAP 可讓彼此不熟悉的物件交換訊息。SOAP 使用以 XML 爲基礎的資料編碼格式和 HTTP 來傳輸訊息。SOAP 與程式設計語言和作業平台無關，而且不需要在其端點上使用任何特定技術。

通用描述、探索及整合 (UDDI) 提供了一種標準的方法，可用來註冊、解除註冊和查找 Web 服務。就像電話系統的電話簿一樣，UDDI 登錄可讓提供者註冊其服務，並允許請求者尋找服務。當請求者找到服務後，登錄對於請求者與提供者就沒有任何功能。

Web 服務描述語言 (WSDL) 定義了一種標準方法，可用來指定 Web 服務的詳細資訊。它是具有一般用途的 XML 模式，可指定 Web 服務介面、連結及其他部署詳細資訊。有了這類指定服務詳細資訊的標準方法，用戶端不必事先瞭解 Web 服務就能使用該服務。

ebXML (Electronic Business using eXtensible Markup Language) 是一組規格，可讓企業透過網際網路營業。OASIS (結構化資訊標準精進組織) 可控制 ebXML 規格。

## Java EE Web 服務標準

XML 處理的 Java API (JAXP) 是不受供應商限制的一組簡易 API，可用來剖析或處理 XML 文件。JAXP 可讓 Web 服務「插入」任何相容的 XML 剖析器。若未「插入」外部剖析器，JAXP 就會使用其本身的 XML 剖析器實作。

「XML 型遠端程序呼叫的 Java API (JAX-RPC)」使用 XML 型通訊協定處理主從式遠端程序呼叫。JAX-RPC 啓用以 SOAP 爲基礎的互通、可移植 Web 服務。開發者使用 JAX-RPC 程式設計模型，開發以 SOAP 爲基礎的 Web 服務端點，以及對應的 WSDL 描述和用戶端。以 JAX-RPC 爲基礎的 Web 服務可與非 Java 用戶端進行互動。同樣地，以 JAX-RPC 爲基礎的用戶端可與非 Java Web 服務進行互動。

XML 登錄的 Java API (JAXR) 是可存取商業登錄的 Java API，其彈性架構支援 UDDI 以及其他登錄規格 (如 ebXML)。JAXR 用戶端可以是獨立的 Java 應用程式，也可以是 J2EE 元件，並使用 JAXR 提供者所提供的 JAXR API 實作來存取商業登錄。JAXR 提供者包含兩個部分：特定登錄的 JAXR 提供者 (提供特定登錄的 API 實作) 及 JAXR 可插接式的提供者 (實作與登錄類型無關的 API 功能)。可插接式提供者不會讓用戶端看到特定登錄提供者的詳細資訊。

開發者可使用「內含適用 Java 附件 API 的 SOAP (SAAJ)」產生與使用符合 SOAP 1.1 規格與「包含附件備註之 SOAP」的訊息。SAAJ 提供一種抽象方法，用於處理內含附件的 SOAP 訊息。進階的開發者可使用 SAAJ 讓應用程式直接使用 SOAP 訊息。附件可能是完整的 XML 文件、XML 片段或 MIME 類型附件。此外，SAAJ 也允許開發者啓用其他 MIME 類型支援。JAX 技術 (如 JAX-RPC) 會在內部使用 SAAJ，讓開發者無需處理 SOAP 的複雜細節。SAAJ 的功能有：

- 同步處理請求與回應訊息傳送作業：用戶端傳送訊息，然後等待回應。
- 單向非同步訊息傳送作業：用戶端傳送訊息並繼續處理，而不等待回應。

# 部署和測試 Web 服務

Application Server 可讓您輕鬆部署與測試 Web 服務。

## 部署 Web 服務

以企業歸檔 (EAR) 部署 Web 服務，就像部署企業應用程式一樣。

也可由 POJO (一般舊 Java 物件) 實作 Web 服務。使用自動部署功能來部署 POJO Web 服務，方法是將其拖放到自動部署目錄中。Application Server 會自動產生適當的 Web XML 檔案並部署 Web 服務。

在「管理主控台」中，您可以在 [Application Server] > [Web 服務] | [一般] 下檢視已部署的 Web 服務清單。

## 檢視已部署的 Web 服務

若要藉由管理主控台來測試 Web 服務，請選取 [應用程式] > [Web 服務] > [*web-service-name*] | [一般]。管理主控台會顯示 Web 服務的屬性：

- 名稱：Web 服務的名稱。
- 端點位址 URI：Web 服務端點的 URI。
- 應用程式：按一下連結以顯示 Web 應用程式或企業應用程式的特性。
- WSDL：按一下連結以顯示 Web 服務的 WSDL 檔案。
- 模組名稱：Web 服務的 WAR 或 EAR 檔案的名稱。
- 對映檔案：按一下連結以顯示 Java WSDL 對映檔案。
- Webservices.xml：按一下連結以顯示 webservices.xml 檔案。
- 實作類型：SERVLET 或 EJB
- 實作類別名稱：
- 部署描述元：

## 測試 Web 服務

管理主控台 可讓您測試 Web 服務與診斷問題。您可以使用通用的測試 Servlet 來 ping 已部署的 Web 服務。會針對每個方法呼叫顯示 SOAP 訊息。

若要藉由管理主控台來測試 Web 服務，請選取 [應用程式] > [Web 服務] > [*web-service-name*] | [一般]，然後按一下 [測試] 按鈕。

## Web 服務安全性

SOAP 訊息層安全性支援是以 WS-Security 的 SAML 記號設定檔爲基礎。也提供 Web 服務的防竄改稽核。

# 使用 Web 服務登錄

**備註 –** Application Server 沒有內部登錄。若要將 Web 服務發佈到內部登錄，則必須在應用程式伺服器中下載和安裝登錄。若要將 Web 服務發佈到外部登錄，請指定外部登錄的位址。

## 增加登錄

在 [Application Server] > [Web 服務] | [登錄] 上，藉由管理主控台增加或移除 Web 服務登錄。使用此頁面建立登錄存取點 (RAP)。增加登錄時，請指定下列參數：

- JNDI 名稱：登錄的連線資源池 (JNDI) 名稱。此連接器資源的 JNDI 名稱是登錄的 JNDI 名稱。
- 選擇要增加的登錄類型：UDDI 3.0 或 ebXML。
- 發佈 URL 和查詢 URL：分別是發佈和查詢登錄的位址。格式爲：http://<主機名稱>/<登錄安裝的路徑>。
- 登錄的使用者名稱和密碼。

可使用下列步驟建立登錄 JNDI 名稱：

- 建立可與登錄通訊的資源配接卡。
- 在應用程式伺服器環境中，預先配置的 JAXR 資源配接卡會與 UDDI 通訊。您也可以下載 SOA 登錄資源配接卡模組。SOA 登錄是 Sun 特有的 ebXML 登錄。
- 使用資源配接卡建立連線資源池。
- 使用此連線池建立連接器資源。

## 將 Web 服務發佈至登錄

若要藉由管理主控台發佈 Web 服務，請選取 [應用程式] > [Web 服務] > [*web-service-name*] | [發佈]。

在 [發佈 Web 服務] 畫面中，選取一個以上的登錄以便發佈 Web 服務，然後按一下 [發佈]。若要發佈所有可用的登錄，請按一下 [全部增加] 按鈕。

輸入要在登錄的哪些類別中顯示此 Web 服務。使用逗號來分隔每個種類。您必須在所使用的登錄中定義種類。輸入此 Web 服務的描述。如果發佈到 UDDI 登錄，請輸入組織的名稱。

如果使用負載平衡器，請輸入負載平衡器主機名稱、連接埠號碼和 SSL 連接埠號碼。如果將 Web 服務發佈到外部登錄 (可透過網際網路在該處找到 WSDL)，這些選項會取代在 WSDL 中，針對其中一個負載平衡器所指定的主機名稱和連接埠號碼。

若要取消發佈 Web 服務，請在 [發佈 Web 服務] 畫面中，選取要從其中取消發佈 Web 服務的登錄，然後按一下 [取消發佈]。

# 以 XSLT 篩選轉換訊息

您可以將 XSLT 轉換規則套用到 Web 服務端點。如此可精準控制 Web 服務的請求和回應。您可以將多個 XSLT 規則套用到 Web 服務端點方法，也可以配置套用轉換的順序。所有 XSLT 檔案皆儲存在中央儲存庫的 `generated/xml/`*appOrModule* 目錄中。這些變換規則會與遠端伺服器實例同步。

您可以將變換規則套用到 SOAP 請求或回應。

若要藉由管理主控台增加要套用的變換規則至 Web 服務作業，請選取 [應用程式] > [Web 服務] > [*web-service-name*] | [變換]。按一下 [新增]。

此時會顯示此 Web 服務端點可以使用的變換規則清單?

瀏覽至包含變換規則的 XSLT 檔案的位置。所有產生的 XSLT 檔案皆儲存在 `generated/xml/`**應用程式或模組名稱**/目錄中。

如果爲一個 Web 服務端點增加多個變換規則，則會依增加變換規則的順序套用這些規則。

若要啓用變換規則，請在 [變換規則] 頁面上選取對應規則的核取方塊，然後按一下 [啓用]。若要停用規則，請按一下 [停用]。

若要移除變換規則，請在 [變換規則] 頁面上選取對應規則的核取方塊，然後按一下 [移除]。如此將從清單中移除變換規則。如果此變換規則已套用至 Web 服務端點，則會自動停用。不過此 XSLT 檔案仍會在檔案路徑位置中。其他 Web 服務端點可使用此 XSLT 檔案。

# 監視 Web 服務

管理主控台可追蹤及以圖形顯示 Web 服務的作業統計，並可顯示 Web 服務所傳送和接收的訊息。

若要啓用 Web 服務監視，請使用管理主控台，從中選取 [應用程式] > [Web服務] > [*web-service-name*] | [監視] | [配置]。

在 [監視配置] 頁面中，設定監視層級：

- 低 - 監視 Web 服務的回應時間、流量、請求總數及錯誤。此功能不會執行方法層級監視。
- 高 - 增加追蹤和監視每秒請求數、平均回應時間及流量屬性的訊息。
- 關閉 - 停用監視。

輸入 [訊息歷程記錄] 的值。預設值是 25。按一下 [重設] 按鈕可清除所有統計資料，並重新啓動正在執行的平均值計算。

## 檢視 Web 服務統計

Application Server9.1 提供了追蹤和以圖形顯示 Web 服務的作業統計功能。

在 [應用程式] > [Web 服務] > [*web-service-name*] | [監視] | [統計] 上檢視監視統計。可用的統計有：

- 所有成功或失敗作業的回應時間，以毫秒爲單位 (最大值、最小值、平均值)。
- 流量
- 請求的總數
- 錯誤的總數，包括發生錯誤之端點的 URI
- 認證失敗的總數
- 授權成功的總數

## 監視 Web 服務訊息

您也可以配置讓 Web 服務檢視 Web 服務端點的訊息 (預設值爲 25)。這些訊息會儲存在遠端伺服器實例的記憶體中。畫面會顯示 SOAP 請求、回應和 HTTP 標頭資訊的詳細資訊。

在 [應用程式] > [Web 服務] > [*web-service-name*] | [監視] | [訊息] 上監視 Web 服務訊息。

啓用時，會看到 Web 服務端點的最後幾則 (預設爲 25) 訊息。這些訊息會保留在遠端伺服器實例的記憶體中，包括 SOAP 請求、回應和 HTTP 標頭資訊的詳細資訊。

顯示 Web 服務所接收的訊息清單。顯示的訊息數視監視配置而定。

您也可以選取篩選器，僅檢視成功訊息或失敗訊息。

◆ ◆ ◆ 第 15 章

# 15 配置物件請求代理程式

本章描述如何配置物件請求代理程式 (ORB) 和 IIOP 偵聽程式。它包含以下小節：



## 物件請求代理程式簡介



### CORBA

Application Server 支援標準的協定集和格式集，可確保互通的功能。這些協定之間的協定是由 CORBA 定義的。

CORBA (共用物件請求代理程式架構) 模型以請求分散式物件服務或伺服器服務的用戶端爲基礎，透過明確定義的介面，以遠端方法請求形式發送物件請求。遠端方法請求傳送有關需要執行的作業的資訊，其中包括被呼叫方法的服務供應商的物件名稱 (稱爲物件參考) 和參數 (如果有)。CORBA 自動處理物件註冊、物件位置、物件啓動、請求非多工、錯誤處理、排列與作業派送等網路程式設計作業。

### 什麼是 ORB？

物件請求代理程式 (ORB) 是 CORBA 的中央元件。ORB 提供所需的基礎架構來識別並尋找物件、處理連線管理、傳送資料並請求通訊。

CORBA 物件之間從不直接進行通訊，該物件是透過遠端存根向在本機機器中執行的 ORB 發出請求。然後，本機 ORB 將請求發送至使用網際網路 Orb 交換協定 (縮寫爲 IIOP) 的另一台機器中的 ORB。然後，遠端 ORB 找到適當的物件、處理請求並傳回結果。

使用 RMI-IIOP，應用程式或物件可將 IIOP 用作遠端方法呼叫 (RMI) 協定。企業 Bean (EJB 模組) 的遠端用戶端透過 RMI-IIOP 與 Application Server 進行通訊。

## IIOP 偵聽程式

IIOP 偵聽程式是一個偵聽通訊端，接受來自企業 Bean 的遠端用戶端和其他 CORBA 型用戶端的內送連線。可以爲 Application Server 配置多個 IIOP 偵聽程式。爲每個偵聽程式指定一個連接埠號碼、一個網路位址和 (選擇性地) 多個安全性屬性。

# 配置 ORB

若要配置 ORB，請在「管理主控台」中，按一下 [配置] 標籤。按一下與要配置的實例相對應的 ORB 標籤。

# 管理 IIOP 偵聽程式

若要建立、編輯和刪除 IIOP 偵聽程式，請按一下「管理主控台」中的 [配置] 標籤。按一下與要配置的實例相對應的 ORB 標籤。選取 [IIOP 偵聽程式] 標籤。如需更多詳細說明，請參閱管理主控台線上說明。

◆ ◆ ◆ 第 16 章

# 16 執行緒池

Java 虛擬機器 (JVM) 一次可支援多個執行緒。爲了提昇效能，Application Server 可維護一個或多個執行緒池。可以將特定的執行緒池指定給連接器模組和 ORB。

一個執行緒池可以爲多個連接器模組和企業 Bean 提供服務。請求執行緒處理使用者對應用程式元件的請求。伺服器收到請求時，它會將該請求指定給執行緒池中的空閑執行緒。執行緒會執行用戶端請求，並傳回結果。例如，如果請求需要使用目前被佔用的系統資源，則此執行緒將等待，直至此資源可用時，才允許請求使用此資源。

指定爲來自應用程式的請求保留的最大執行緒數和最小執行緒數。可以在這兩個值之間，動態調整執行緒池。指定的最小執行緒池大小將通知伺服器爲應用程式請求至少分配該大小的保留執行緒數。該數目可以增加到所指定的最大執行緒池大小。

增加程序可用的執行緒數目，可讓程序同時回應更多的應用程式請求。

透過將 Application Server 執行緒分到不同的執行緒池中，避免在一個資源配接卡或應用程式佔用 Application Server 中的所有執行緒時，出現執行緒不足的情況。

本章包含以下主題：



## 配置執行緒池

若要使用管理主控台建立執行緒池，請移至 [配置] > [執行緒池] > [目前的池] > [新增]。

- 在 [執行緒池 ID] 欄位中輸入執行緒池的名稱。
- 在 [最小執行緒池大小] 欄位中，輸入爲此佇列中的請求提供服務的執行緒池中執行緒的最小數目。

  創設此執行緒池時將預先建立這些執行緒。
- 在 [最大執行緒池大小] 欄位中，輸入爲此佇列中的請求提供服務的執行緒池中執行緒的最大數目。

這是存在於此執行緒池中的執行緒數上限。

- 在 [閒置逾時] 欄位中輸入數值 (以秒爲單位)，超過此時間段之後將從池中移除閒置執行緒。
- 在 [工作佇列數] 欄位中，輸入此執行緒池所處理的工作佇列總數。
- 重新啓動 Application Server。

如需有關建立執行緒池的詳細資訊，請按一下管理主控台的「說明」。

您也可以使用 `asadmin` 指令 `create-threadpool`，從指令行刪除執行緒池。

若要使用管理主控台來編輯執行緒池的設定，請移至 [配置] > [執行緒池] > [目前的池]，並選取您要配置的池。修改所選取之執行緒池的值，儲存並重新啓動 Application Server。

如需有關編輯執行緒池的詳細資訊，請按一下管理主控台的 [說明]。

若要使用管理主控台刪除執行緒池，請移至 [配置] > [執行緒池] > [目前的池]。檢查要刪除的執行緒池名稱，然後按一下 [刪除]。

重新啓動應用程式伺服器。

您也可以使用 `asadmin` 指令 `delete-threadpool`，從指令行刪除執行緒池。

◆ ◆ ◆ 第 17 章

# 17 配置記錄

本章簡要說明如何配置記錄並檢視伺服器記錄。它包含以下小節：



## 關於記錄



### 記錄記錄

Application Server 使用 JSR 047 中指定的 Java EE 平台記錄 API。Application Server 記錄訊息記錄在伺服器記錄中，通常位於 *domain-dir*`/logs/server.log`。當記錄自動重建時，Application Server 會建立名爲 `server.log` 的全新空白檔案，並將舊檔案重新命名爲 `server.log_`*date*，其中 *date* 是檔案自動重建的日期和時間。

*domain-dir*`/logs` 目錄中除了包含伺服器記錄之外，還包含兩種其他類型的記錄。`access` 子目錄中包含 HTTP 服務存取記錄，`tx` 子目錄中包含作業事件服務記錄。如需有關這些記錄的資訊，請參閱第 135 頁的「配置作業事件」。

Application Server 的元件可產生記錄輸出。應用程式元件也可以產生記錄輸出。

應用程式元件可以使用 Apache Commons Logging Library 來記錄訊息。但是，建議採用平台標準 JSR 047 API 以獲得更好的記錄配置。

記錄檔的記錄遵循以下統一格式：

```
[#|yyyy-mm-ddThh:mm:ss.SSS-Z|Log Level|ProductName-Version|LoggerName|Key Value Pairs|Message|#]
```

例如：

```
[#|2006-10-21T13:25:53.852-0400|INFO|sun-appserver9.1|javax.enterprise.
system.core|_ThreadID=13;|CORE5004: Resource Deployed:
[cr:jms/DurableConnectionFactory].|#]
```

在此範例中，

- `[#` 和 `#]` 標示該記錄的開頭和結尾。
- 垂直線 (`|`) 用於分隔記錄欄位。
- `2006-10-21T13:25:53.852-0400` 指定日期和時間。
- *Log Level* 爲 `INFO`。此層級可以是以下任何值：`SEVERE`、`WARNING`、`INFO`、`CONFIG`、`FINE`、`FINER` 和 `FINEST`。
- *ProductName-Version* 爲 `sun-appserver9.1`。
- *LoggerName* 是一種階層式記錄程式名稱空間，用於識別記錄模組來源，在此例中爲 `javax.enterprise.system.core`。
- *Key Value Pairs* 爲鍵名和鍵值，通常爲執行緒 ID，例如 `_ThreadID=14;`。
- *Message* 是記錄訊息的文字。對於所有的 Application Server `SEVERE` 和 `WARNING` 訊息以及多種 `INFO`，它均以包含模組代碼和數值的訊息 ID 開頭 (在此範例中爲 `CORE5004`)。

以後的版本中，可能會變更或增強記錄檔的記錄格式。

# 記錄程式名稱空間階層結構

Application Server 爲它的每個模組都提供了記錄程式。下表依照字母順序，列出模組名稱和每個記錄程式的名稱空間，如管理主控台 [記錄層級] 頁面所示 (請參閱第 158 頁的「配置記錄層級」)。[記錄層級] 頁面中未顯示表中最後三個模組。

**表 17–1** Application Server 記錄程式名稱空間

| 模組名稱 | 名稱空間 |
|---|---|
| 管理 | `javax.enterprise.system.tools.admin` |
| 類別載入程式 | `javax.enterprise.system.core.classloading` |
| 配置 | `javax.enterprise.system.core.config` |
| 連接器 | `javax.enterprise.resource.resourceadapter` |
| CORBA | `javax.enterprise.resource.corba` |
| 部署 | `javax.enterprise.system.tools.deployment` |
| EJB 容器 | `javax.enterprise.system.container.ejb` |

表 17–1 Application Server 記錄程式名稱空間 (續)

| 模組名稱 | 名稱空間 |
|---|---|
| 群組管理服務 (僅限叢集和企業設定檔) | javax.ee.enterprise.system.gms |
| JavaMail | javax.enterprise.resource.javamail |
| JAXR | javax.enterprise.resource.webservices .registry |
| JAXRPC | javax.enterprise.resource.webservices.rpc |
| JAXWS | javax.enterprise.resource.webservices.javaws |
| JBI | com.sun.jbi |
| JMS | javax.enterprise.resource.jms |
| JTA | javax.enterprise.resource.jta |
| JTS | javax.enterprise.system.core.transaction |
| MDB 容器 | javax.enterprise.system.container.ejb.mdb |
| 命名 | javax.enterprise.system.core.naming |
| 持續性 | oracle.toplink.essentials、<br>javax.enterprise.resource.jdo、<br>javax.enterprise.system.container.cmp |
| 節點代理程式 (僅限叢集和企業設定檔) | javax.ee.enterprise.system.nodeagent |
| 根 | javax.enterprise |
| SAAJ | javax.enterprise.resource.webservices.saaj |
| 安全性 | javax.enterprise.system.core.security |
| 自我管理 | javax.enterprise.system.core.selfmanagement |
| 伺服器 | javax.enterprise.system |
| 同步化 (僅限叢集和企業設定檔) | javax.ee.enterprise.system.tools .synchronization |
| Util | javax.enterprise.system.util |
| 檢驗器 | javax.enterprise.system.tools.verifier |
| Web 容器 | javax.enterprise.system.container.web<br>org.apache.catalina org.apache.coyote<br>org.apache.jasper |

# 配置記錄

本小節包含下列主題：



## 配置一般記錄設定

若要使用管理主控台配置一般記錄設定：

- 如需有關開發者設定檔的資訊，請移至 [Application Server] → [記錄] → [一般]
- 如需有關叢集和企業設定檔的資訊，請移至 [配置] → [配置] → [記錄設定] → [一般]

在 [一般] 頁面上，依據您的需求，輸入適當值以自訂記錄。停止並重新啓動 Application Server。

如需設定各種配置參數的詳細資訊，請在管理主控台中按一下 [說明]。

若要以 `asadmin` 配置這些記錄設定，請使用 `get` 和 `set` 指令。

## 配置記錄層級

若要使用管理主控台配置記錄層級：

- 如需有關開發者設定檔的資訊，請移至 [Application Server] → [記錄] → [記錄層級]，
- 如需有關叢集和企業設定檔的資訊，請移至 [配置] → [配置] → [記錄] → [記錄設定] → [記錄層級]

爲此頁面上所列出的模組設定記錄層級。使用 [附加特性] 區域可以爲任何應用程式記錄程式配置記錄層級。如需有關模組記錄程式清單的資訊，請參閱第 156 頁的「記錄程式名稱空間階層結構」。

如需設定各種配置參數的詳細資訊，請在管理主控台中按一下 [說明]。

若要在 `asadmin` 中配置這些記錄設定，請使用 `get` 和 `set` 指令。

# 檢視伺服器記錄

若要檢視記錄檔：

- 在開發者設定檔中，移至 [應用程式伺服器] → [記錄] → [檢視記錄檔]。
- 在叢集和企業設定檔中，移至 [配置] → [配置] → [記錄程式設定] → [一般]，再按一下 [檢視記錄檔]。

使用 [搜尋條件] 區域所提供的選項，依據您的喜好設定顯示記錄結果。

- **實例名稱** — 從下拉式清單中選擇一個實例名稱，以檢視該伺服器實例的記錄。預設值爲目前伺服器實例。
- **記錄檔** — 從下拉式清單中選擇一個記錄檔名稱，以檢視該記錄的內容。預設爲 `server.log`。
- **時間戳記** — 若要檢視最新的訊息，請選取 [最新] (預設)。如果只檢視特定時間段內的訊息，請選取 [特定範圍] 並在顯示的 [從] 欄位和 [到] 欄位中鍵入日期和時間值。對於時間值，其語法必須採用以下格式 (*SSS* 表示毫秒)：

  **`hh:mm:ss.SSS`**

  例如：

  `17:10:00.000`

  如果 [從] 欄位中的時間值遲於 [到] 欄位中的時間值，將顯示錯誤訊息。
- **記錄層級** — 若要依照記錄層級篩選訊息，請從下拉式清單中選擇一種記錄層級。依預設，將顯示伺服器記錄中選取記錄層級和更嚴重記錄層級的所有訊息。選取標有 [不包含更高層級的訊息] 的核取方塊，可僅顯示所選層級的訊息。

  若要確定您要檢視的訊息都顯示在伺服器記錄中，請先在 [記錄層級] 頁面中設定適當的記錄層級。請參閱第 158 頁的 「配置記錄層級」。

  如果您選擇基於記錄層級篩選記錄訊息，則將只顯示符合指定篩選條件的訊息。不過，這種篩選不影響那些記錄到伺服器記錄中的訊息。

  將顯示伺服器記錄中最新的 40 個項目以及在 [記錄設定] 和 [記錄層級] 頁面中指定的設定。

  按一下 [時間戳記] 標頭旁邊的箭頭對這些訊息進行排序，使最新的訊息顯示在最後。

  若要檢視訊息的格式化版本，請按一下標示的連結

  `(details)`

  將顯示標有 [記錄項目詳細資訊] 的視窗，該視窗包含了訊息的格式化版本。

  在項目清單的末尾，按一下按鈕以檢視記錄檔中較早或較晚的項目。

按一下 [搜尋條件] 區域中的 [進階搜尋] 以進一步細化記錄檢視器的搜尋條件。使用如下所示的 [進階選項] 欄位：

- **記錄程式** — 若要依模組進行篩選，請從下拉式清單中選擇一個或多個名稱空間。按住 Shift 鍵並按一下或按住 Ctrl 鍵並按一下來選擇多個名稱空間。

  選取更高層級的名稱空間也就選取了該名稱空間下的所有名稱空間。例如，選取 `javax.enterprise.system` 的同時也就選取了該名稱空間下所有模組的記錄程式：`javax.enterprise.system.core`、`javax.enterprise.system.tools.admin` 等等。

- **自訂記錄程式** — 若要檢視特定於某個特定應用程式的記錄程式中的訊息，請在文字欄位中鍵入記錄程式名稱 (每行一個名稱)。如果應用程式具有多個模組，您可以檢視任何模組，也可以檢視所有模組。例如，假定應用程式具有使用以下名稱的記錄程式：

  ```
  com.mycompany.myapp.module1
  com.mycompany.myapp.module2
  com.mycompany.myapp.module3
  ```

  若要檢視應用程式中所有模組的訊息，請鍵入 `com.mycompany.myapp`。若要僅檢視 `module2` 的訊息，請鍵入 `com.mycompany.myapp.module2`。

  如果指定了一個或多個自訂記錄程式，則僅當您在 [記錄程式] 區域中明確指定 Application Server 模組的訊息後，才會顯示這些訊息。

- **名稱值對** — 若要檢視特定執行緒的輸出，請在文字欄位中鍵入該執行緒的鍵名和鍵值。鍵名爲 `_ThreadID`。例如：

  ```
  _ThreadID=13
  ```

  假定 `com.mycompany.myapp.module2` 在多個執行緒中執行。若要限制記錄檢視器使其只顯示單一執行緒的輸出，請在 [自訂記錄程式] 欄位中指定該模組的記錄程式，然後在此欄位中指定執行緒 ID。

- **顯示** — 若要一次檢視 40 條以上的訊息 (預設)，請從下拉式清單中選擇其他可用的值 (100、250 或 1000)。

  若要檢視堆疊追蹤，請取消選取 [限制過長訊息] 核取方塊。依預設，檢視器中不會顯示堆疊追蹤；若要檢視堆疊追蹤，請按一下訊息的 [(**詳細資訊**)] 連結。

  按一下 [基本搜尋] 可以隱藏 [進階選項] 區域。

◆ ◆ ◆ 第 18 章

# 18 監視元件和服務

本章包含有關使用 Application Server 管理主控台來監視元件的資訊。本章包括下列小節：



## 關於監視



### 監視 Application Server

使用監視功能可以觀察在 Application Server 的伺服器實例中所部署的各種元件和服務的執行階段狀態。利用有關執行階段元件和程式狀態的資訊，可以確定效能瓶頸以便進行調校、和容量規劃、預測故障、在發生故障時分析根本原因，以及確保一切執行正常。

啓用監視功能會因增加系統耗用而使效能降低。

也可以選擇在伺服器達到特定的效能臨界值時採取行動。如需更多資訊，請參閱第 19 章。

# 監視概述

若要監視 Application Server，請執行以下步驟：

1. 使用管理主控台或 asadmin 工具啓用對特定服務和元件的監視功能。
   如需有關此步驟的更多資訊，請參閱第 184 頁的 「啓用與停用監視」。
2. 使用管理主控台或 asadmin 工具檢視指定服務或元件的監視資料。
   如需有關此步驟的更多資訊，請參閱第 185 頁的 「檢視監視資料」。

# 關於可監視物件的樹狀結構

Application Server 使用樹狀結構追蹤可監視物件。由於監視物件的樹是動態的，因此在實例中增加、更新或移除元件時該樹會相應地發生變更。樹狀結構中的根物件是伺服器實例名稱 (例如 server)。(在 Platform Edition 中，僅允許使用一個伺服器實例。)

以下指令顯示了樹的頂層：

```
asadmin> list --user adminuser --monitor server
server.applications
server.http-service
server.connector-service
server.jms-service
server.jvm
server.orb
server.resources
server.thread-pools
```

以下小節描述了這些子樹：

- 第 162 頁的 「應用程式樹」
- 第 163 頁的 「HTTP 服務樹」
- 第 164 頁的 「資源樹」
- 第 164 頁的 「連接器服務樹」
- 第 164 頁的 「JMS 服務樹」
- 第 165 頁的 「ORB 樹」
- 第 165 頁的 「執行緒池樹」

## 應用程式樹

以下示意圖顯示了企業應用程式的各種元件的頂層節點和子節點。具有可用的監視統計資訊的節點標有星號 (*)。如需更多資訊，請參閱第 166 頁的 「EJB 容器統計資訊」。

**範例 18–1** 應用程式節點樹狀結構

```
applications
   |--- application1
   |    |--- ejb-module-1
   |    |        |--- ejb1 *
   |    |                 |--- cache (for entity/sfsb) *
   |    |                 |--- pool (for slsb/mdb/entity) *
   |    |                 |--- methods
   |    |                      |---method1 *
   |    |                      |---method2 *
   |    |                 |--- stateful-session-store (for sfsb)*
   |    |                 |--- timers (for s1sb/entity/mdb) *
   |    |--- web-module-1
   |    |        |--- virtual-server-1 *
   |    |                      |---servlet1  *
   |    |                      |---servlet2  *
   |--- standalone-web-module-1
   |    |        |----- virtual-server-2 *
   |    |                      |---servlet3 *
   |    |                      |---servlet4 *
   |    |        |----- virtual-server-3 *
   |    |                      |---servlet3 *(same servlet on different vs)
   |    |                      |---servlet5 *
   |--- standalone-ejb-module-1
   |    |        |--- ejb2 *
   |    |                 |--- cache (for entity/sfsb) *
   |    |                 |--- pool (for slsb/mdb/entity) *
   |    |                 |--- methods
   |    |                      |--- method1 *
   |    |                      |--- method2 *
   |--- application2
```

## HTTP 服務樹

以下示意圖顯示了 HTTP 服務的節點。具有可用的監視資訊的節點標有星號 (*)。請參閱第 171 頁的 「HTTP 服務統計資訊」。

**範例 18–2** HTTP 服務圖解 (開發者設定檔版本)

```
http-service
         |--- virtual-server-1
              |--- http-listener-1 *
              |--- http-listener-2 *
         |--- virtual-server-2
              |--- http-listener-1 *
              |--- http-listener-2 *
```

**範例 18–3** HTTP 服務圖解 (叢集和企業設定檔版本)

```
http-service *
             |---connection-queue *
             |---dns *
             |---file-cache *
             |---keep-alive *
             |---pwc-thread-pool *
             |---virtual-server-1*
             |        |--- request *
             |---virtual-server-2*
             |        |--- request *
```

## 資源樹

資源節點包含 JDBC 連線池和連接器連線池等池的可監視屬性。以下示意圖顯示了各種資源元件的頂層節點和子節點。具有可用的監視統計資訊的節點標有星號 (*)。請參閱第 171 頁的 「JDBC 連線池統計資訊」。

**範例 18–4** 資源示意圖

```
resources
        |---connection-pool1(either connector-connection-pool or jdbc)*
        |---connection-pool2(either connector-connection-pool or jdbc)*
```

## 連接器服務樹

連接器服務節點包含連接器連線池等池的可監視屬性。以下示意圖顯示了各種連接器服務元件的頂層節點和子節點。具有可用的監視統計資訊的節點標有星號 (*)。請參閱第 172 頁的 「JMS/連接器服務統計資訊」。

**範例 18–5** 連接器服務示意圖

```
connector-service
        |--- resource-adapter-1
        |        |-- connection-pools
        |        |        |-- pool-1 (All pool stats for this pool)
        |        |-- work-management (All work mgmt stats for this RA)
```

## JMS 服務樹

JMS 服務節點包含連接器連線池等池的可監視屬性。以下示意圖顯示了各種 JMS 服務元件的頂層節點和子節點。具有可用的監視統計資訊的節點標有星號 (*)。

**範例 18–6** JMS服務示意圖

```
jms-service
        |-- connection-factories [AKA conn. pools in the RA world]
        |        |-- connection-factory-1 (All CF stats for this CF)
        |-- work-management (All work mgmt stats for the MQ-RA)
```

### ORB 樹

ORB 節點包含連線管理員的可監視屬性。以下示意圖顯示了 ORB 元件的頂層節點和子節點。具有可用的監視統計資訊的節點標有星號 (*)。請參閱第 174 頁的 「ORB 中連線管理程式的統計資訊」。

**範例 18–7** ORB示意圖

```
orb
        |--- connection-managers
        |        |--- connection-manager-1 *
        |        |--- connection-manager-1 *
```

### 執行緒池樹

執行緒池節點包含連線管理程式的可監視屬性。以下示意圖顯示了 ORB 元件的頂層節點和子節點。具有可用的監視統計資訊的節點標有星號 (*)。請參閱第 174 頁的 「執行緒池統計資訊」。

**範例 18–8** 執行緒池示意圖

```
thread-pools
        |    |--- thread-pool-1 *
        |    |--- thread-pool-2 *
```

## 關於受監視的元件和服務的統計資訊

本小節描述可用的監視統計資訊：

- 第 179 頁的「生產 Web 容器 (PWC) 統計資訊」

## EJB 容器統計資訊

以下表格說明了 EJB 容器統計資訊：

- 表 18–1
- 表 18–2
- 表 18–3
- 表 18–4
- 表 18–5
- 表 18–6

下表說明 EJB 統計資訊。

**表 18–1** EJB 統計資訊

| 屬性名稱 | 資料類型 | 說明 |
|---|---|---|
| createcount | 計數統計資訊 | 呼叫 EJB 的 create 方法的次數。 |
| removecount | 計數統計資訊 | 呼叫 EJB 的 remove 方法的次數。 |
| pooledcount | 範圍統計資訊 | 處於匯集狀態的實體 Bean 的數目。 |
| readycount | 範圍統計資訊 | 處於就緒狀態的實體 Bean 的數目。 |
| messagecount | 計數統計資訊 | 訊息驅動 Bean 收到的訊息數。 |
| methodreadycount | 範圍統計資訊 | 處於 MethodReady 狀態的有狀態或無狀態階段作業 Bean 的數目。 |
| passivecount | 範圍統計資訊 | 處於 Passive 狀態的有狀態階段作業 Bean 的數目。 |

下表中列出了 EJB 方法呼叫的可用統計資訊。

**表 18–2** EJB 方法統計資訊

| 屬性名稱 | 資料類型 | 說明 |
|---|---|---|
| methodstatistic | 時間統計資訊 | 作業被呼叫的次數；呼叫期間所花費的總時間等資訊。 |

**表 18–2** EJB 方法統計資訊 *(續)*

| 屬性名稱 | 資料類型 | 說明 |
|---|---|---|
| totalnumerrors | 計數統計資訊 | 執行方法導致異常的次數。如果爲 EJB 容器啓用了監視功能，則此統計資訊是爲無狀態和有狀態階段作業 Bean 和實體 Bean 而收集的。 |
| totalnumsuccess | 計數統計資訊 | 方法成功執行的次數。如果爲 EJB 容器啓用了監視功能，則此統計資訊是爲無狀態和有狀態階段作業 Bean 和實體 Bean 收集的。 |
| executiontime | 計數統計資訊 | 上次成功/不成功嘗試執行方法作業所花費的時間 (ms)。如果在 EJB 容器中啓用了監視功能，則此統計資訊是爲無狀態和有狀態階段作業 Bean 和實體 Bean 而收集的。 |

下表中列出了 EJB 階段作業儲存的統計資訊。

**表 18–3** EJB 階段作業儲存統計資訊

| 屬性名稱 | 資料類型 | 說明 |
|---|---|---|
| currentSize | 範圍統計資訊 | 目前位於儲存中的鈍化階段作業數或檢查點階段作業數。 |
| activationCount | 計數統計資訊 | 從儲存中啓動的階段作業數。 |
| activationSuccessCount | 計數統計資訊 | 從儲存中成功啓動的階段作業數 |
| activationErrorCount | 計數統計資訊 | 上次成功/不成功嘗試執行方法作業所花費的時間 (ms)。如果在 EJB 容器中啓用了監視功能，則此統計資訊是爲無狀態和有狀態階段作業 Bean 和實體 Bean 而收集的。 |
| passivationCount | 計數統計資訊 | 使用此儲存鈍化 (取消啓動) 的階段作業數。 |
| passivationSuccessCount | 計數統計資訊 | 使用此儲存成功鈍化的階段作業數。 |
| passivationErrorCount | 計數統計資訊 | 無法使用此儲存鈍化的階段作業數。 |
| expiredSessionCount | 計數統計資訊 | 此儲存移除的過期階段作業數。 |

表 18–3 EJB階段作業儲存統計資訊 (續)

| 屬性名稱 | 資料類型 | 說明 |
|---|---|---|
| passivatedBeanSize | 計數統計資訊 | 由此儲存鈍化的總位元組數，包括總數、最小值和最大值。 |
| passivationTime | 計數統計資訊 | 將 Bean 鈍化到儲存所花費的時間，包括總時間值、最小值和最大值。 |
| checkpointCount (僅限企業設定檔) | 計數統計資訊 | 使用此儲存進行階段作業檢查點操作的階段作業數。 |
| checkpointSuccessCount (僅限企業設定檔) | 計數統計資訊 | 成功進行檢查點操作的階段作業數。 |
| checkpointErrorCount (僅限企業設定檔) | 計數統計資訊 | 無法進行檢查點操作的階段作業數。 |
| checkpointedBeanSize (僅限企業設定檔) | 值統計資訊 | 由該儲存進行檢查點操作的位元組總數。 |
| checkpointTime (僅限企業設定檔) | 時間統計資訊 | 通過檢查點操作將 Bean 放入儲存中所花費的時間。 |

下表中列出了 EJB 池的可用統計資訊。

表 18–4 EJB 池統計資訊

| 屬性名稱 | 資料類型 | 說明 |
|---|---|---|
| numbeansinpool | 限制範圍統計資訊 | 相關聯池中的 EJB 數，可提供有關池的變更方式的資訊。 |
| numthreadswaiting | 限制範圍統計資訊 | 等待自由 Bean 的執行緒數，指出請求可能擁塞。 |
| totalbeanscreated | 計數統計資訊 | 從開始收集資料以來相關聯池中所建立的 Bean的數目。 |
| totalbeansdestroyed | 計數統計資訊 | 自開始收集資料以來從相關聯池中銷毀的 Bean 的數目。 |
| jmsmaxmessagesload | 計數統計資訊 | 針對要服務的訊息驅動 Bean 而一次載入 JMS 階段作業的最大訊息數。預設值爲 1。僅套用於訊息驅動 Bean 的池。 |

下表中列出了 EJB 快取的可用統計資訊。

表 18–5 EJB 快取統計資訊

| 屬性名稱 | 資料類型 | 說明 |
|---|---|---|
| cachemisses | 限制範圍統計資訊 | 使用者請求未在快取中找到 Bean 的次數。 |
| cachehits | 限制範圍統計資訊 | 使用者請求找到快取記憶體中某個項目的次數。 |
| numbeansincache | 限制範圍統計資訊 | 快取記憶體中 Bean 的數目。這便是快取記憶體目前的大小。 |
| numpassivations | 計數統計資訊 | 鈍化 Bean 的數目。僅套用於有狀態階段作業 Bean。 |
| numpassivationerrors | 計數統計資訊 | 鈍化期間發生的錯誤數。僅套用於有狀態階段作業 Bean。 |
| numexpiredsessionsremoved | 計數統計資訊 | 清除執行緒所移除的過期階段作業數目。僅套用於有狀態階段作業 Bean。 |
| numpassivationsuccess | 計數統計資訊 | 鈍化成功完成的次數。僅套用於有狀態階段作業 Bean。 |

下表中列出了計時器的可用統計資訊。

表 18–6 計時器統計資訊

| 統計資訊 | 資料類型 | 說明 |
|---|---|---|
| numtimerscreated | 計數統計資訊 | 系統中建立的計時器的數目。 |
| numtimersdelivered | 計數統計資訊 | 系統所傳送的計時器的數目。 |
| numtimersremoved | 計數統計資訊 | 從系統中移除的計時器的數目。 |

## Web 容器統計資訊

物件樹狀結構中包含了 Web 容器，如第 162 頁的 「應用程式樹」所示。系統爲每個單獨的 Web 應用程式都顯示了 Web 容器統計資訊。Servlet 之 Web 容器的可用統計資訊列在表 18–7 中，Web 模組的可用統計資訊則顯示在表 18–8 中。

表 18–7 Web 容器 (Servlet) 統計資訊

| 統計資訊 | 單位 | 資料類型 | 注釋 |
|---|---|---|---|
| errorcount | 數目 | 計數統計資訊 | 回應碼大於或等於 400 的情況的累積次數。 |
| maxtime | 毫秒 | 計數統計資訊 | Web 容器等待請求的最長時間。 |

表 18–7 Web 容器 (Servlet) 統計資訊 (續)

| 統計資訊 | 單位 | 資料類型 | 注釋 |
|---|---|---|---|
| processingtime | 毫秒 | 計數統計資訊 | 處理每個請求所需時間的累積值。處理時間是總請求處理時間除以請求計數所得的平均值。 |
| requestcount | 數目 | 計數統計資訊 | 到目前爲止所處理的請求總數。 |

第 169 頁的 「Web 容器統計資訊」中列出了 Web 模組的可用統計資訊。

表 18–8 Web 容器 (Web 模組) 統計資訊

| 統計資訊 | 資料類型 | 注釋 |
|---|---|---|
| jspcount | 計數統計資訊 | 已載入 Web 模組的 JSP 頁面的數目。 |
| jspreloadcount | 計數統計資訊 | 已重新載入 Web 模組的 JSP 頁面的數目。 |
| sessionstotal | 計數統計資訊 | 已爲 Web 模組建立的階段作業總數。 |
| activesessionscurrent | 計數統計資訊 | Web 模組的目前處於使用中狀態的階段作業數。 |
| activesessionshigh | 計數統計資訊 | Web 模組的同時處於使用中狀態的階段作業最大數。 |
| rejectedsessionstotal | 計數統計資訊 | Web 模組的被拒絕的階段作業總數。是指由於允許處於使用中狀態的階段作業數已達到最大值而未被建立的階段作業數。 |
| expiredsessionstotal | 計數統計資訊 | Web 模組的已過期階段作業的總數。 |
| sessionsize | 平均範圍統計資訊 | Web 模組的階段作業大小。值可以爲大、小或平均值，或以位元組爲單位 (用於串列化階段作業)。 |
| sessionpersisttime | 平均範圍統計資訊 | 將 HTTP 階段作業狀態保持到 Web 模組的後端儲存中所花費的時間 (以 ms 爲單位，短、長或平均值)。 |
| cachedsessionscurrent | 計數統計資訊 | 快取在記憶體中用於 Web 模組的目前階段作業數。 |

表 18–8 Web 容器 (Web 模組) 統計資訊 (續)

| 統計資訊 | 資料類型 | 注釋 |
|---|---|---|
| passivatedsessionscurrent | 計數統計資訊 | Web 模組的目前已鈍化的階段作業數。 |

## HTTP 服務統計資訊

下表顯示了開發者設定檔之 HTTP 服務的可用統計資訊。如需其他設定檔的 HTTP 服務的統計資訊，請參閱第 179 頁的 「生產 Web 容器 (PWC) 統計資訊」。

表 18–9 HTTP 服務統計資訊 (開發者設定檔)

| 統計資訊 | 單位 | 資料類型 | 注釋 |
|---|---|---|---|
| bytesreceived | 位元組 | 計數統計資訊 | 每個請求處理器接收到的位元組累積值。 |
| bytessent | 位元組 | 計數統計資訊 | 每個請求處理器所傳送的位元組累積值。 |
| currentthreadcount | 數目 | 計數統計資訊 | 目前位於偵聽程式執行緒池中的處理執行緒數。 |
| currentthreadsbusy | 數目 | 計數統計資訊 | 處理請求的偵聽程式執行緒池中目前正在使用的請求處理執行緒的數目。 |
| errorcount | 數目 | 計數統計資訊 | 錯誤計數的累積值，錯誤計數是指回應碼大於或等於 400 這類情況發生的次數。 |
| maxsparethreads | 數目 | 計數統計資訊 | 可以存在的未使用回應處理執行緒的最大數目。 |
| minsparethreads | 數目 | 計數統計資訊 | 可以存在的未使用回應處理執行緒的最小數目。 |
| maxthreads | 數目 | 計數統計資訊 | 偵聽程式所建立的請求處理執行緒的最大數目。 |
| maxtime | 毫秒 | 計數統計資訊 | 處理執行緒的最長時間。 |
| processing-time | 毫秒 | 計數統計資訊 | 處理每個請求所花費時間的累積值。處理時間是總請求處理時間除以請求計數所得的平均值。 |
| request-count | 數目 | 計數統計資訊 | 到目前爲止所處理的請求總數。 |

## JDBC 連線池統計資訊

用於在執行階段監視 JDBC 資源，以測量效能並擷取資源使用情況。由於建立 JDBC 連線的成本很高並且常常會導致應用程式出現效能瓶頸問題，因此對 JDBC 連線池如何釋放和建立新連線以及正在等待從特定池中擷取連線的執行緒數的監視是至關重要的。

下表中列出了 JDBC 連線池的可用統計資訊。

**表 18–10** JDBC 連線池統計資訊

| 統計資訊 | 單位 | 資料類型 | 說明 |
|---|---|---|---|
| numconnfailedvalidation | 數目 | 計數統計資訊 | 從開始時間到上次取樣時間爲止在連線池中驗證失敗的連線總數。 |
| numconnused | 數目 | 範圍統計資訊 | 提供連線使用統計資訊。目前正使用的連線總數，以及有關使用過的連線的最大數目的資訊 (高水印)。 |
| numconnfree | 數目 | 計數統計資訊 | 上次取樣時池中的自由連線總數。 |
| numconntimedout | 數目 | 限制範圍統計資訊 | 開始時間與上次取樣時間之間池中的逾時連線總數。 |
| averageconnwaittime | 數目 | 計數統計資訊 | 指示嘗試與連接器連線池建立成功連線請求的平均等待時間。 |
| waitqueuelength | 數目 | 計數統計資訊 | 佇列中正在等待處理的連線請求數。 |
| connectionrequestwaittime | | 範圍統計資訊 | 連線請求的最長和最短等待時間。目前值表示上次由池處理的請求的等待時間。 |
| numconncreated | 毫秒 | 計數統計資訊 | 自上次重設以來建立的實體連線數。 |
| numconndestroyed | 數目 | 計數統計資訊 | 自上次重設以來已銷毀的實體連線數。 |
| numconnacquired | 數目 | 計數統計資訊 | 從池中獲取的邏輯連線數。 |
| numconnreleased | 數目 | 計數統計資訊 | 釋放到池中的邏輯連線數。 |

## JMS/連接器服務統計資訊

表 18–11 顯示了連接器連線池的可用統計資訊。表 18–12 顯示了連接器運作管理的統計資訊。

**表 18–11** 連接器連線池統計資訊

| 統計資訊 | 單位 | 資料類型 | 說明 |
|---|---|---|---|
| numconnfailedvalidation | 數目 | 計數統計資訊 | 從開始時間到上次取樣時間爲止在連線池中驗證失敗的連線總數。 |
| numconnused | 數目 | 範圍統計資訊 | 提供連線使用統計資訊。目前正使用的連線總數，以及有關使用過的連線的最大數目的資訊 (高水印)。 |
| numconnfree | 數目 | 範圍統計資訊 | 上次取樣時池中的自由連線總數。 |
| numconntimedout | 數目 | 計數統計資訊 | 開始時間與上次取樣時間之間池中的逾時連線總數。 |
| averageconnwaittime | 數目 | 計數統計資訊 | 連線由連線池處理之前的平均等待時間。 |
| waitqueuelenght | 數目 | 計數統計資訊 | 佇列中正在等待處理的連線請求數。 |
| connectionrequestwaittime | | 範圍統計資訊 | 連線請求的最長和最短等待時間。目前值表示上次由池處理的請求的等待時間。 |
| numconncreated | 毫秒 | 計數統計資訊 | 自上次重設以來建立的實體連線數。 |
| numconndestroyed | 數目 | 計數統計資訊 | 自上次重設以來已銷毀的實體連線數。 |
| numconnacquired | 數目 | 計數統計資訊 | 從池中獲取的邏輯連線數。 |
| numconnreleased | 數目 | 計數統計資訊 | 釋放到池中的邏輯連線數。 |

下表列出了連接器運作管理的可用統計資訊。

**表 18–12** 連接器工作管理統計資訊

| 統計資訊 | 資料類型 | 說明 |
|---|---|---|
| activeworkcount | 範圍統計資訊 | 由連接器執行的工作物件數。 |
| waitqueuelength | 範圍統計資訊 | 執行前在佇列中等待的工作物件數。 |
| workrequestwaittime | 範圍統計資訊 | 工作物件在被執行前所等待的最長和最短時間。 |
| submittedworkcount | 計數統計資訊 | 由連接器模組提交的工作物件數。 |
| rejectedworkcount | 計數統計資訊 | Application Server 拒絕的工作物件數。 |
| completedworkcount | 計數統計資訊 | 完成的工作物件數。 |

## ORB 中連線管理程式的統計資訊

下表列出了 ORB 中連線管理員的可用統計資訊。

**表 18–13** ORB 中連線管理程式的統計資訊

| 統計資訊 | 單位 | 資料類型 | 說明 |
|---|---|---|---|
| connectionsidle | 數目 | 計數統計資訊 | 提供與 ORB 的閒置連線的總數。 |
| connectionsinuse | 數目 | 計數統計資訊 | 提供 ORB 的使用中連線總數。 |
| totalconnections | 數目 | 限制範圍統計資訊 | 與 ORB 的連線總數。 |

## 執行緒池統計資訊

下表中列出了執行緒池的可用統計資訊。

**表 18–14** 執行緒池統計資訊

| 統計資訊 | 單位 | 資料類型 | 說明 |
|---|---|---|---|
| averagetimeinqueue | 毫秒 | 範圍統計資訊 | 在得到處理之前請求在佇列中等待的平均時間 (以毫秒爲單位)。 |
| averageworkcompletion-time | 毫秒 | 範圍統計資訊 | 完成指定所花費的平均時間 (以毫秒爲單位)。 |
| currentnumberofthreads | 數目 | 限制範圍統計資訊 | 目前的請求處理執行緒的數目。 |
| numberofavailablethreads | 數目 | 計數統計資訊 | 可用的執行緒數。 |
| numberofbusythreads | 數目 | 計數統計資訊 | 處於忙碌狀態的執行緒數。 |
| totalworkitemsadded | 數目 | 計數統計資訊 | 到目前爲止增加到工作佇列中的工作項目總數。 |

## 作業事件服務統計資訊

作業事件服務允許用戶端凍結作業事件子系統，以回復作業事件，並確定在凍結期間進行處理的作業事件。下表中列出了作業事件服務的可用統計資訊。

**表 18–15** 作業事件服務統計資訊

| 統計資訊 | 資料類型 | 說明 |
|---|---|---|
| activecount | 計數統計資訊 | 目前處於使用中狀態的作業事件數。 |
| activeids | 字串統計資訊 | 目前處於使用中狀態的作業事件的 ID。每個此類作業事件均可在凍結作業事件服務後轉返。 |

表 18–15 作業事件服務統計資訊 (續)

| 統計資訊 | 資料類型 | 說明 |
|---|---|---|
| committedcount | 計數統計資訊 | 已確定的作業事件數。 |
| rolledbackcount | 計數統計資訊 | 已轉返的作業事件數。 |
| state | 字串統計資訊 | 指示作業事件是否已凍結。 |

## Java 虛擬機器 (JVM) 統計資訊

JVM 具有始終處於啓用狀態的可監視屬性。下表中列出了 JVM 的可用統計資訊。

表 18–16 JVM 統計資訊

| 統計資訊 | 資料類型 | 說明 |
|---|---|---|
| heapsize | 限制範圍統計資訊 | 包含 JVM 的記憶體堆疊大小上限和下限的常駐記憶體佔用空間。 |
| uptime | 計數統計資訊 | JVM 已執行的時間。 |

## Java SE 中的 JVM 統計資訊

利用 Java SE，可從 JVM 取得額外的監視資訊。將監視層級設定爲 [低] 以啓用這些附加資訊的顯示。將監視層級設定爲 [高] 還可以檢視與系統中每個活性執行緒相關的資訊。如需有關 Java SE 的附加監視功能的更多資訊，請參閱 http://java.sun.com/javase/6/docs/technotes/guides/management/ 中標題爲「*Monitoring and Management for the Java Platform*」的文件。

以下網頁討論 Java SE 監視工具：http://java.sun.com/javase/6/docs/technotes/tools/#manage。

下表顯示 Java SE 中 JVM 類別載入作業的可用統計資訊。

表 18–17 Java SE 的 JVM 統計資訊 - 類別載入

| 統計資訊 | 資料類型 | 說明 |
|---|---|---|
| loadedclasscount | 計數統計資訊 | 目前載入 JVM 的類別的數目。 |
| totalloadedclasscount | 計數統計資訊 | 自 JVM 開始執行以來已載入的類別的總數。 |
| unloadedclasscount | 計數統計資訊 | 自 JVM 開始執行以來已從 JVM 中卸載的類別的數目。 |

下表顯示 Java SE 中 JVM 編譯的可用統計資訊。

**表 18–18** Java SE 的 JVM 統計資訊 - 編譯

| 統計資訊 | 資料類型 | 說明 |
|---|---|---|
| totalcompilationtime | 計數統計資訊 | 編譯所花費的累積時間 (以毫秒爲單位)。 |

下表顯示 Java SE 中 JVM 資源回收的可用統計資訊。

**表 18–19** Java SE 的 JVM 統計資訊 - 資源回收

| 統計資訊 | 資料類型 | 說明 |
|---|---|---|
| collectioncount | 計數統計資訊 | 已發生的回收的總數。 |
| collectiontime | 計數統計資訊 | 累積的收集時間 (以毫秒爲單位)。 |

下表顯示 Java SE 中 JVM 記憶體的可用統計資訊。

**表 18–20** Java SE 的 JVM 統計資訊 - 記憶體

| 統計資訊 | 資料類型 | 說明 |
|---|---|---|
| objectpendingfinalizationcount | 計數統計資訊 | 擱置結束操作的物件的大約數目。 |
| initheapsize | 計數統計資訊 | 最初由 JVM 請求的堆疊的大小。 |
| usedheapsize | 計數統計資訊 | 目前正在使用的堆疊的大小。 |
| maxheapsize | 計數統計資訊 | 可用於記憶體管理的最大記憶體容量 (以位元組爲單位)。 |
| committedheapsize | 計數統計資訊 | 確定可由 JVM 使用的記憶體容量 (以位元組爲單位)。 |
| initnonheapsize | 計數統計資訊 | 最初由 JVM 請求的非堆疊區域的大小。 |
| usednonheapsize | 計數統計資訊 | 目前正在使用的非堆疊區域的大小。 |
| maxnonheapsize | 計數統計資訊 | 可用於記憶體管理的最大記憶體容量 (以位元組爲單位)。 |
| committednonheapsize | 計數統計資訊 | 確定可由 JVM 使用的記憶體容量 (以位元組爲單位)。 |

下表顯示 Java SE 中 JVM 作業系統的可用統計資訊。

**表 18–21** Java SE 的 JVM 統計資訊 - 作業系統

| 統計資訊 | 資料類型 | 說明 |
|---|---|---|
| arch | 字串統計資訊 | 作業系統架構。 |
| availableprocessors | 計數統計資訊 | 可用於 JVM 的處理器的數目。 |
| name | 字串統計資訊 | 作業系統名稱。 |
| version | 字串統計資訊 | 作業系統版本。 |

下表顯示 Java SE 中 JVM 執行階段的可用統計資訊。

**表 18–22** Java SE 的 JVM 統計資訊 - 執行階段

| 統計資訊 | 資料類型 | 說明 |
|---|---|---|
| name | 字串統計資訊 | 表示正在執行的 JVM 的名稱 |
| vmname | 字串統計資訊 | JVM 實作名稱。 |
| vmvendor | 字串統計資訊 | JVM 實作供應商。 |
| vmversion | 字串統計資訊 | JVM 實作版本。 |
| specname | 字串統計資訊 | JVM 規格名稱。 |
| specvendor | 字串統計資訊 | JVM 規格供應商。 |
| specversion | 字串統計資訊 | JVM 規格版本。 |
| managementspecversion | 字串統計資訊 | 由 JVM 實作的管理規格版本。 |
| classpath | 字串統計資訊 | 系統類別載入器搜尋類別檔案時所使用的類別路徑。 |
| librarypath | 字串統計資訊 | Java 程式庫路徑。 |
| bootclasspath | 字串統計資訊 | 啓動程式類別載入器搜尋類別檔案時所使用的類別路徑。 |
| inputarguments | 字串統計資訊 | 傳送給 JVM 的輸入引數。不包括 main 方法的引數。 |
| uptime | 計數統計資訊 | JVM 的正常執行時間 (以毫秒爲單位)。 |

下表顯示 Java SE 中 JVM ThreadInfo 的可用統計資訊。

**表 18–23** Java SE 的 JVM 統計資訊 - 執行緒資訊

| 統計資訊 | 資料類型 | 說明 |
|---|---|---|
| threadid | 計數統計資訊 | 執行緒 ID。 |
| threadname | 字串統計資訊 | 執行緒名稱。 |
| threadstate | 字串統計資訊 | 執行緒狀態。 |
| blockedtime | 計數統計資訊 | 執行緒進入 BLOCKED 狀態以來所經歷的時間 (以毫秒爲單位)。如果已停用執行緒競爭狀態監視功能，則傳回 -1。 |
| blockedcount | 計數統計資訊 | 執行緒進入 BLOCKED 狀態的總次數。 |
| waitedtime | 計數統計資訊 | 執行緒處於 WAITING 狀態以來所經歷的時間 (以毫秒爲單位)。如果已停用執行緒競爭狀態監視功能，則傳回 -1。 |
| waitedcount | 計數統計資訊 | 執行緒處於 WAITING 或 TIMED_WAITING 狀態的總次數。 |
| lockname | 字串統計資訊 | 監視鎖定的字串表示，表示執行緒被暫停而無法進入或執行緒正在透過 Object.wait 方法等待接收通知。 |
| lockownerid | 計數統計資訊 | 包含某個物件的監視鎖定的執行緒 ID，該執行緒在該物件上發生區段化。 |
| lockownername | 字串統計資訊 | 包含某個物件的監視所定的執行緒名稱，該執行緒在該物件上發生區段化。 |
| stacktrace | 字串統計資訊 | 與該執行緒相關聯的堆疊追蹤。 |

下表顯示 Java SE 中 JVM 執行緒的可用統計資訊。

**表 18–24** Java SE 的 JVM 統計資訊 - 執行緒

| 統計資訊 | 資料類型 | 說明 |
|---|---|---|
| threadcount | 計數統計資訊 | 目前的活性常駐程式執行緒和非常駐程式執行緒數。 |
| peakthreadcount | 計數統計資訊 | JVM 啓動或峰重設以來的峰活性執行緒計數。 |
| totalstartedthreadcount | 計數統計資訊 | JVM 啓動以來建立和/或啓動的執行緒總數。 |
| daemonthreadcount | 計數統計資訊 | 目前的活性常駐程式執行緒數。 |

表 18–24 Java SE 的 JVM 統計資訊 - 執行緒 (續)

| 統計資訊 | 資料類型 | 說明 |
|---|---|---|
| allthreadids | 字串統計資訊 | 所有活性執行緒 ID 清單。 |
| currentthreadcputime | 計數統計資訊 | 如果已啓用 CPU 時間測量，則表示目前執行緒的 CPU 時間 (以納秒爲單位)。如果已停用 CPU 時間測量，則傳回 -1。 |
| monitordeadlockedthreads | 字串統計資訊 | 處於監視死結狀態的執行緒 ID 清單。 |

## 生產 Web 容器 (PWC) 統計資訊

下表說明下列 PWC 元件和服務的可用統計資訊。

**備註 –** 這些統計資訊僅可供叢集和企業設定檔使用。

- 表 18–25
- 表 18–26
- 表 18–27
- 表 18–28
- 表 18–29
- 表 18–30
- 表 18–31
- 表 18–32

下表列出了 PWC 虛擬伺服器的統計資訊。

表 18–25 PWC 虛擬伺服器統計資訊

| 屬性名稱 | 資料類型 | 說明 |
|---|---|---|
| id | 字串統計資訊 | 虛擬伺服器的 ID。 |
| mode | 字串統計資訊 | 虛擬伺服器所處的模式。選項包括 unknown 或 active。 |
| hosts | 字串統計資訊 | 由該虛擬伺服器提供服務的主機的名稱。 |
| interfaces | 字串統計資訊 | 配置了虛擬伺服器的介面 (偵聽程式) 的類型。 |

下表中列出了 PWC 請求的可用統計資訊。

表 18–26 PWC請求統計資訊

| 屬性名稱 | 資料類型 | 說明 |
|---|---|---|
| method | 字串統計資訊 | 用於請求的方法。 |
| uri | 字串統計資訊 | 上一個被處理的URI。 |
| countrequests | 計數統計資訊 | 已處理的請求數。 |
| countbytestransmitted | 計數統計資訊 | 傳輸的位元組數，如果此資訊不可用，則值爲0。 |
| countbytesreceived | 計數統計資訊 | 接收的位元組數，如果此資訊不可用，則值爲0。 |
| ratebytesreceived | 計數統計資訊 | 在某段伺服器定義的時間間隔內的資料接收速率，如果此資訊不可用，則值爲0 |
| maxbytestransmissionrate | 計數統計資訊 | 資料在某段伺服器定義的時間間隔內的最大傳輸速率，如果此資訊不可用，則值爲0。 |
| countopenconnections | 計數統計資訊 | 目前開啓的連線數，如果此資訊不可用，則值爲0。 |
| maxopenconnections | 計數統計資訊 | 同時開啓的連線的最大數目，如果此資訊不可用，則值爲0。 |
| count2xx | 計數統計資訊 | 代碼爲2XX的回應的總數。 |
| count3xx | 計數統計資訊 | 代碼爲3XX的回應的總數。 |
| count4xx | 計數統計資訊 | 代碼爲4XX的回應的總數。 |
| count5xx | 計數統計資訊 | 代碼爲5XX的回應的總數。 |
| countother | 計數統計資訊 | 具有其他回應代碼的回應總數。 |
| count200 | 計數統計資訊 | 代碼爲200的回應的總數。 |
| count302 | 計數統計資訊 | 代碼爲302的回應的總數。 |
| count304 | 計數統計資訊 | 代碼爲304的回應的總數。 |
| count400 | 計數統計資訊 | 代碼爲400的回應的總數。 |
| count401 | 計數統計資訊 | 代碼爲401的回應的總數。 |
| count403 | 計數統計資訊 | 代碼爲403的回應的總數。 |
| count404 | 計數統計資訊 | 代碼爲404的回應的總數。 |
| count503 | 計數統計資訊 | 代碼爲503的回應的總數。 |

快取資訊小節提供了有關目前檔案快取的使用方式的資訊。下表列出了 PWC 檔案快取的統計資訊。

**表 18–27** PWC檔案快取統計資訊

| 屬性名稱 | 資料類型 | 說明 |
|---|---|---|
| flagenabled | 計數統計資訊 | 指示是否啓用了檔案快取。禁用時有效值爲 0，啓用時有效值爲 1。 |
| secondsmaxage | 計數統計資訊 | 有效快取項目的最長存在時間 (以秒爲單位)。 |
| countentries | 計數統計資訊 | 目前的快取項目數。單一快取項目表示單一 URI。 |
| maxentries | 計數統計資訊 | 同步式快取項目的最大數目。 |
| countopenentries | 計數統計資訊 | 與開啓的檔案關聯的項目數。 |
| maxopenentries | 計數統計資訊 | 與開啓的檔案關聯的同步式快取項目的最大數目。 |
| sizeheapcache | 計數統計資訊 | 用於快取內容的堆疊儲存區空間。 |
| maxheapcachesize | 計數統計資訊 | 用於快取檔案內容的最大堆疊儲存區空間。 |
| sizemmapcache | 計數統計資訊 | 記憶體對映的檔案內容所使用的位址空間大小。 |
| maxmmapcachesize | 計數統計資訊 | 檔案快取用於記憶體對映檔案內容的最大位址空間大小。 |
| counthits | 計數統計資訊 | 快取查找成功的次數。 |
| countmisses | 計數統計資訊 | 快取查找失敗的次數。 |
| countinfohits | 計數統計資訊 | 檔案資訊查找成功的次數。 |
| countinfomisses | 計數統計資訊 | 快取的檔案資訊遺失的數目。 |
| countcontenthits | 計數統計資訊 | 快取的檔案內容的命中次數。 |
| countcontentmisses | 計數統計資訊 | 檔案資訊查找失敗的次數。 |

本小節提供有關伺服器的 HTTP 層級持續作用的系統的資訊。下表中列出了 PWC 持續作用的可用統計資訊。

**表 18–28** PWC持續作用統計資訊

| 屬性名稱 | 資料類型 | 說明 |
|---|---|---|
| countconnections | 計數統計資訊 | 處於持續作用模式的連線數。 |
| maxconnections | 計數統計資訊 | 允許同步處於持續作用模式的最大連線數。 |

表 18–28 PWC持續作用統計資訊 *(續)*

| 屬性名稱 | 資料類型 | 說明 |
|---|---|---|
| counthits | 計數統計資訊 | 處於持續作用模式的連線隨後進行了有效請求的總次數。 |
| countflushes | 計數統計資訊 | 伺服器關閉持續作用的連線的次數。 |
| countrefusals | 計數統計資訊 | 可能由於有太多的持續性連線而使伺服器無法將連線傳送到持續作用執行緒的次數。 |
| counttimeouts | 計數統計資訊 | 伺服器因用戶端連線逾時且沒有任何活動而終止持續作用連線的次數。 |
| secondstimeout | 計數統計資訊 | 關閉閒置持續作用連線之前經歷的時間 (以秒爲單位)。 |

DNS 快取快取 IP 位妝M DNS 名稱。依預設，伺服器的 DNS 快取處於停用狀態。單一快取項目表示單一 IP 位址或 DNS 名稱查詢。下表中列出了 PWC DNS 的可用統計資訊。

表 18–29 PWC DNS 統計資訊

| 屬性名稱 | 資料類型 | 說明 |
|---|---|---|
| flagcacheenabled | 計數統計資訊 | 指出 DNS 快取是否被啓用 (開啓)。禁用時爲 0，啓用時爲 1。 |
| countcacheentries | 計數統計資訊 | 目前位於快取中的 DNS 項目數。 |
| maxcacheentries | 計數統計資訊 | 快取中可容納的 DNS 項目的最大數目。 |
| countcachehits | 計數統計資訊 | DNS 快取查找成功的次數。 |
| countcachemisses | 計數統計資訊 | DNS 快取查找失敗的次數。 |
| flagasyncenabled | 計數統計資訊 | 指示是否啓用了非同步 DNS 查找。禁用時爲 0，啓用時爲 1。 |
| countasyncnamelookups | 計數統計資訊 | 非同步 DNS 名稱查找的總數。 |
| countasyncaddrlookups | 計數統計資訊 | 非同步 DNS 位妞d找的總數。 |
| countasynclookupsinprogress | 計數統計資訊 | 正在進行的非同步查找的數目。 |

下表列出了 PWC 執行緒池的統計資訊。

**表 18–30** PWC執行緒池統計資訊

| 屬性名稱 | 資料類型 | 說明 |
|---|---|---|
| id | 字串統計資訊 | 執行緒池 ID。 |
| countthreadsidle | 計數統計資訊 | 目前處於閒置狀態的請求處理執行緒數。 |
| countthreads | 計數統計資訊 | 目前的請求處理執行緒的數目。 |
| maxthreads | 計數統計資訊 | 可同時存在的請求處理執行緒的最大數目。 |
| countqueued | 計數統計資訊 | 佇列等候由此執行緒池進行處理的請求數。 |
| peakqueued | 計數統計資訊 | 佇列中同步容納的最大請求數。 |
| maxqueued | 計數統計資訊 | 佇列一次可容納的最大請求數。 |

連線佇列是請求被處理前保存這些請求的佇列。連線佇列的統計資訊顯示佇列中的階段作業數以及連線被接受前的平均延遲。下表列出了 PWC 連線佇列的統計資訊。

**表 18–31** PWC連線佇列統計資訊

| 屬性名稱 | 資料類型 | 說明 |
|---|---|---|
| id | 字串統計資訊 | 連線佇列的 ID。 |
| counttotalconnections | 計數統計資訊 | 已接受的連線總數。 |
| countqueued | 計數統計資訊 | 目前位於佇列中的連線數。 |
| peakqueued | 計數統計資訊 | 佇列中同步容納的最大連線數。 |
| maxqueued | 計數統計資訊 | 連線佇列的最大大小。 |
| countoverflows | 計數統計資訊 | 佇列太滿而無法容納連線的次數。 |
| counttotalqueued | 計數統計資訊 | 已佇列的連線總數。某個給定連線可能會被多次加入佇列，因此 counttotalqueued 可能大於或等於 counttotalconnections。 |
| tickstotalqueued | 計數統計資訊 | 連線在佇列中所花費的週期總數。週期是由系統決定的時間單位。 |
| countqueued1minuteaverage | 計數統計資訊 | 前 1 分鐘內處於佇列狀態的平均連線數。 |
| countqueued5minuteaverage | 計數統計資訊 | 前 5 分鐘內處於佇列狀態的平均連線數。 |
| countqueued15minuteaverage | 計數統計資訊 | 前 15 分鐘內處於佇列狀態的平均連線數。 |

下表列出了 PWC HTTP服務的統計資訊。

表 18–32 PWC HTTP 服務統計資訊

| 屬性名稱 | 資料類型 | 說明 |
|---|---|---|
| id | 字串統計資訊 | HTTP 服務的實例名稱。 |
| versionserver | 字串統計資訊 | HTTP 服務的版本編號。 |
| timestarted | 字串統計資訊 | 啓動 HTTP 服務的時間 (GMT)。 |
| secondsrunning | 計數統計資訊 | HTTP 服務啓動以來所經歷的時間 (以秒爲單位)。 |
| maxthreads | 計數統計資訊 | 每個實例中的最大工作者執行緒數。 |
| maxvirtualservers | 計數統計資訊 | 每個實例中可以配置的最大虛擬伺服器數目。 |
| flagprofilingenabled | 計數統計資訊 | 是否啓用了 HTTP 服務效能設定檔。有效值爲 0 或 1。 |
| flagvirtualserveroverflow | 計數統計資訊 | 表示是否配置了多於 maxvirtualservers 的虛擬伺服器。如果設定爲 1，則不會爲所有的虛擬伺服器追蹤統計資訊。 |
| load1minuteaverage | 計數統計資訊 | 前 1 分鐘內請求的平均負載。 |
| load5minuteaverage | 計數統計資訊 | 前 5 分鐘內請求的平均負載。 |
| load15minuteaverage | 計數統計資訊 | 前 15 分鐘內請求的平均負載。 |
| ratebytestransmitted | 計數統計資訊 | 在某段伺服器定義的時間間隔內資料的傳輸速率。如果此資訊不可用，則結果爲 0。 |
| ratebytesreceived | 計數統計資訊 | 在某段伺服器定義的時間間隔內資料的接收速率。如果此資訊不可用，則結果爲 0。 |

# 啓用與停用監視

本小節包含下列主題：



## 使用管理主控台配置監視層級

在管理主控台中配置監視：

- 對於開發者設定檔，請移至 [配置] → [監視]
- 對於叢集和企業設定檔，請移至 [配置] → [配置] → [監視]

依預設，所有元件和服務的監視功能均處於關閉狀態。若要啓用監視功能，請在組合方塊中選取 [低] 或 [高]。若要關閉監視功能，請在組合方塊中選取 [關閉]。

如需配置監視的詳細資訊，請參閱管理主控台線上說明。

### ▼ 使用 asadmin 配置監視層級

1 **使用 get 指令可以查找目前已對哪些服務和元件啓用了監視功能：**

```
asadmin> get --user admin-user server.monitoring-service.module-monitoring-levels.*
```

傳回：

```
server.monitoring-service.module-monitoring-levels.
connector-connection-pool = OFF
server.monitoring-service.module-monitoring-levels.
connector-service = OFF
server.monitoring-service.module-monitoring-levels.ejb-container = OFF
server.monitoring-service.module-monitoring-levels.http-service = OFF
server.monitoring-service.module-monitoring-levels.jdbc-connection-pool = OFF
server.monitoring-service.module-monitoring-levels.jms-service = OFF
server.monitoring-service.module-monitoring-levels.jvm = OFF
server.monitoring-service.module-monitoring-levels.orb = OFF
server.monitoring-service.module-monitoring-levels.thread-pool = OFF
server.monitoring-service.module-monitoring-levels.transaction-service = OFF
server.monitoring-service.module-monitoring-levels.web-container = OFF
```

2 **使用 set 指令可以啓用監視功能。**

例如，若要啓用對 HTTP 服務的監視功能，請輸入以下指令：

```
asadmin> set --user admin-user
server.monitoring-service.module-monitoring-levels.http-service=LOW
```

若要停用監視功能，請使用 set 指令並將監視層級指定爲 OFF。

# 檢視監視資料



## 在管理主控台中檢視監視資料

在開發者設定檔中，若要檢視監視資料，請移至 [Application Server] → [監視]。

在叢集和企業設定檔中，若要檢視獨立實例的監視資料，請移至 [獨立實例] → [實例] → [監視]。若要檢視叢集實例的監視資料，請移至 [叢集] → [叢集] → [實例] → [監視]。

您可以選取和檢視 JVM、伺服器、應用程式、執行緒池、HTTP 服務、作業事件服務、記錄統計資訊，以及呼叫流程統計資訊的監視資料。第 162 頁的 「關於可監視物件的樹狀結構」圖表說明這些元件和服務的組織方式。

如需有關檢視或配置監視的詳細資訊，請參閱管理主控台線上說明。

如需有關每個元件或服務之屬性的更多資訊，請參閱第 165 頁的 「關於受監視的元件和服務的統計資訊」。

# 使用 asadmin 工具檢視監視資料

本小節包含下列主題：



## ▼ 使用 asadmin monitor 指令檢視監視資料

`asadmin` 具有兩種檢視監視資料的方法。第一種方法是使用 `monitor` 指令。此指令會列印經常監視的統計資訊，並可選擇篩選統計資訊，以及擷取輸出至「逗點分隔值 (CSV)」檔案。

1 **若要檢視監視資料，請使用 monitor 指令，並指定監視資料的類型：** `httplistener`、`keepalive`、`filecache`、`connectionqueue`、`jdbcpool`、`jvm`、`threadpool`、`servlet`、`connection`、`connectorpool`、`endpoint`、`entitybean`、`messagedriven`、`statefulsession`、`statelesssession`、`httpservice` **或** `webmodule`。

例如，若要檢視 `server` 上的 `jvm` 的資料，請輸入下列指令：

```
asadmin>monitor --type jvm --user adminuser server

                                JVM Monitoring
UpTime(ms)                                      HeapSize(bytes)
current                  min        max         low        high       count
327142979                0          531628032   0          45940736   45940736
```

2 **若要檢視監視資料，並將輸出傳送給 CSV 檔案，請使用 filename 選項。例如：**

```
asadmin> monitor --type jvm --filename myoutputfile --user adminuser server
```

## ▼ 使用 asadmin get 和 list 指令檢視監視資料

monitor 指令在大多數情況下都很有用。不過，它並未提供所有可監視物件的完整清單。若要使用 asadmin 工具檢視所有可監視資料，請使用 asadmin list 和 asadmin get 指令，後面跟著可監視物件的帶點名稱，方法如下。

1 **若要檢視可監視物件的名稱，請使用 asadmin list 指令。**

例如，若要檢視伺服器實例上已啓用監視功能的應用程式元件和子系統的清單，請在終端機視窗中鍵入以下指令：

```
asadmin> list --user adminuser --monitor server
```

上述指令將傳回已啓用監視功能的應用程式元件與子系統清單，例如：

```
server.resources
server.connector-service
server.orb
server.jms-service
server.jvm
server.applications
server.http-service
server.thread-pools
```

如需有關使用 list 指令的更多範例，請參閱第 189 頁的 「list 和 get 指令的範例」。如需有關可與 list 指令配合使用之帶點名稱的更多資訊，請參閱第 188 頁的 「瞭解和指定含點名稱」。

2 **若要顯示已啓用監視功能的應用程式元件或子系統的監視統計資訊，請使用 asadmin get 指令。**

若要取得統計資訊，請在終端機視窗中鍵入 asadmin get 指令，並指定在先前步驟中由 list 指令顯示的名稱。以下範例嘗試獲取某個特定物件的子系統的所有屬性：

```
asadmin> get --user adminuser --monitor server.jvm.*
```

該指令將傳回以下屬性和資料：

```
server.jvm.dotted-name = server.jvm
server.jvm.heapsize-current = 21241856
server.jvm.heapsize-description = Provides statistical information about
   the JVM's memory heap size.
server.jvm.heapsize-highwatermark = 21241856
server.jvm.heapsize-lastsampletime = 1080232913938
server.jvm.heapsize-lowerbound = 0
server.jvm.heapsize-lowwatermark = 0
server.jvm.heapsize-name = JvmHeapSize
```

```
server.jvm.heapsize-starttime = 1080234457308
server.jvm.heapsize-unit = bytes
server.jvm.heapsize-upperbound = 518979584
server.jvm.uptime-count = 1080234457308
server.jvm.uptime-description = Provides the amount of time the JVM has
   been running.
server.jvm.uptime-lastsampletime = 1080234457308
server.jvm.uptime-name = JvmUpTime
server.jvm.uptime-starttime = 1080232913928
server.jvm.uptime-unit = milliseconds
```

如需有關使用 get 指令的更多範例，請參閱第 189 頁的「list 和 get 指令的範例」。如需有關可與 get 指令配合使用之帶點名稱的更多資訊，請參閱第 188 頁的「瞭解和指定含點名稱」。

## 瞭解和指定含點名稱

在 asadmin list 和 get 指令中，指定可監視物件的帶點名稱。所有子物件均使用點 (.) 字元做爲分隔符來命名，因此這些子物件稱爲**帶點名稱**。如果子節點爲單一型，則僅需監視物件類型即可命名物件；否則，需要形式爲 type.name 的名稱來命名物件。

例如，http-service 就是一種有效的可監視物件類型，並且爲單一型。若要命名表示實例 server 之 http-service 的單一型子節點，則帶點名稱爲：

```
server.http-service
```

另一個範例，application 爲一種有效的可監視物件類型，但並非單一型。例如，若要命名表示應用程式 Petstore 的非單一型子節點，則帶點名稱爲：

```
server.applications.petstore
```

含點名稱也可以命名可監視物件中的特定屬性。例如，http-service 具有名爲 bytesreceived-lastsampletime 的可監視屬性。以下名稱可以命名 bytesreceived 屬性：

```
server.http-service.server.http-listener-1.
  bytesreceived-lastsampletime
```

管理員無需知道 asadmin list 和 get 指令的有效帶點名稱。使用 list 指令可以顯示可用的可監視物件，而使用帶有萬用字元參數的 get 指令可以檢查任意可監視物件的所有可用屬性。

使用具有帶點名稱的 list 和 get 指令的基本假設爲：

- 使用任何具有帶點名稱且後面**不**帶有萬用字元 (*) 的 list 指令，得到的結果爲目前節點的直接子節點。例如，list --user *adminuser* --monitor server 將列出屬於 server 節點的所有直接子節點。

- 使用任何具有帶點名稱且後面帶有 .* 形式的萬用字元的 list 指令，得到的結果爲目前節點的子節點階層式樹狀結構。例如，list --user *adminuser* --monitor server.applications.* 將列出 applications 的所有子節點及其後續子節點等。
- 使用任何具有帶點名稱且前面或後面帶有 **dottedname*、*dotted* name* 或 ***dotted name** 形式的萬用字元的 list 指令，得到的結果爲符合常規表示式 (由提供的符合式樣建立) 的所有節點及其子節點。
- 使用後面帶有 .* 或 * 的 get 指令，得到的結果爲屬於要符合的目前節點的一組屬性及其值。

如需更多資訊，請參閱第 194 頁的 「list 和 get 指令在所有層級上的預期輸出」 。

## list 和 get 指令的範例

本小節包含下列主題：



## list --user admin-user --monitor 指令的範例

list 指令可針對指定的伺服器實例名稱，提供有關目前正在監視之應用程式元件與子系統的資訊。使用此指令，您可以查看伺服器實例的可監視元件與子元件。如需 list 範例的更完整清單，請參閱第 194 頁的 「list 和 get 指令在所有層級上的預期輸出」 。

### 範例 1

```
asadmin> list --user admin-user --monitor server
```

上述指令將傳回已啓用監視功能的應用程式元件與子系統清單，例如：

```
server.resources
server.orb
server.jvm
server.jms-service
server.connector-service
server.applications
server.http-service
server.thread-pools
```

還可以列出指定的伺服器實例中目前所監視的應用程式。這對於使用 get 指令從某應用程式中搜尋特定監視統計資訊很有幫助。

### 範例 2

```
asadmin> list --user admin-user --monitor server.applications
```

傳回：

```
server.applications.adminapp
 server.applications.admingui
server.applications.myApp
```

如需更完整的範例，請參閱第 191 頁的 「使用 PetStore 範例」。

## get --user admin-user --monitor 指令的範例

此指令可擷取以下受監視的資訊：

- 一個元件或子系統內的全部受監視屬性
- 一個元件或子系統內特定的受監視屬性

  當特定元件或子系統所請求的屬性不存在時，將會傳回錯誤。同樣，當元件或子系統所請求的特定屬性不在作用中時，也會傳回錯誤。

  請參閱第 194 頁的 「list 和 get 指令在所有層級上的預期輸出」，以取得有關使用 `get` 指令的更多資訊。

### 範例 1

嘗試從某子系統中取得特定物件的所有屬性：

```
asadmin> get --user admin-user --monitor server.jvm.*
```

傳回：

```
server.jvm.dotted-name= server.jvm
server.jvm.heapsize-current = 21241856
server.jvm.heapsize-description = Provides statistical information about
   the JVM's memory heap size.
server.jvm.heapsize-highwatermark = 21241856
server.jvm.heapsize-lastsampletime = 1080232913938
server.jvm.heapsize-lowerbound = 0
server.jvm.heapsize-lowwatermark = 0
server.jvm.heapsize-name = JvmHeapSize
server.jvm.heapsize-starttime = 1080234457308
server.jvm.heapsize-unit = bytes
server.jvm.heapsize-upperbound = 518979584
server.jvm.uptime-count = 1080234457308
server.jvm.uptime-description = Provides the amount of time the JVM has
   been running.
server.jvm.uptime-lastsampletime = 1080234457308
server.jvm.uptime-name = JvmUpTime
server.jvm.uptime-starttime = 1080232913928
server.jvm.uptime-unit = milliseconds
```

### 範例 2

嘗試從 Java EE 應用程式中取得所有屬性：

```
asadmin> get --user admin-user --monitor server.applications.myJavaEEApp.*
```

傳回：

```
No matches resulted from the wildcard expression.
CLI137 Command get failed.
```

該 Java-EE 應用程式層級上沒有可監視的屬性，因此顯示此回覆。

### 範例 3

嘗試從某子系統中取得特定屬性：

```
asadmin> get --user admin-user --monitor server.jvm.uptime-lastsampletime
```

傳回：

```
server.jvm.uptime-lastsampletime = 1093215374813
```

### 範例 4

嘗試從某子系統屬性中取得不明的屬性：

```
asadmin> get --user admin-user --monitor server.jvm.badname
```

傳回：

```
No such attribute found from reflecting the corresponding Stats
interface: [badname]
CLI137 Command get failed.
```

## ▼ 使用 PetStore 範例

以下範例說明如何將 `asadmin` 工具用於監視目的。

使用者要檢查將範例 `PetStore` 應用程式部署至 Application Server 之後，在該應用程式中呼叫某個方法的次數。已部署該應用程式的實例名稱爲 `server`。結合 `list` 與 `get` 指令，即可存取所需的方法統計資訊。

1. **啓動 Application Server 和 `asadmin` 工具。**
2. **設定一些有用的環境變數，以避免為每個指令均輸入這些變數：**

```
asadmin> export AS_ADMIN_USER=admin AS_ADMIN_PASSWORD=admin123
asadmin>export AS_ADMIN_HOST=localhost AS_ADMIN_PORT=4848
```

3 **列出實例 server 的可監視元件：**

```
asadmin> list --user adminuser --monitor server*
```

傳回 (輸出將與以下內容類似)：

```
server
server.applications
server.applications.CometEJB
server.applications.ConverterApp
server.applications.petstore
server.http-service
server.resources
server.thread-pools
```

可監視元件清單包括 thread-pools、http-service、resources 以及所有已部署 (與已啓用) 的 applications。

4 **列出 PetStore 應用程式中的可監視子元件 (可以使用 -m 替代 --monitor)：**

```
asadmin> list -m server.applications.petstore
```

傳回：

```
server.applications.petstore.signon-ejb_jar
server.applications.petstore.catalog-ejb_jar
server.applications.petstore.uidgen-ejb_jar
server.applications.petstore.customer-ejb_jar
server.applications.petstore.petstore-ejb_jar
server.applications.petstore.petstore\.war
server.applications.petstore.AsyncSenderJAR_jar
server.applications.petstore.cart-ejb_jar
```

5 **列出 PetStore 應用程式之 EJB 模組 signon-ejb_jar 中的可監視子元件：**

```
asadmin> list -m server.applications.petstore.signon-ejb_jar
```

傳回：

```
server.applications.petstore.signon-ejb_jar.SignOnEJB
server.applications.petstore.signon-ejb_jar.UserEJB
```

6 **列出 PetStore 應用程式之 EJB 模組 signon-ejb_jar 的實體 Bean UserEJB 中的可監視子元件：**

```
asadmin> list -m server.applications.petstore.signon-ejb_jar.UserEJB
```

傳回 (爲節省空間而移除帶點名稱)：

```
server.applications.petstore.signon-ejb_jar.UserEJB.bean-cache
server.applications.petstore.signon-ejb_jar.UserEJB.bean-methods
server.applications.petstore.signon-ejb_jar.UserEJB.bean-pool
```

**7** 列出實體 Bean UserEJB (位於 PetStore 應用程式的 EJB 模組 signon-ejb_jar 中) 之 getUserName 方法中的可監視子元件：

```
asadmin> list -m
server.applications.petstore.signon-ejb_jar.UserEJB.bean-methods.getUserName
```

傳回：

```
Nothing to list at server.applications.petstore.signon-ejb_jar.
UserEJB.bean-methods.getUserName. To get the valid names beginning with a
string, use the wildcard "*" character. For example, to list all names
that begin with "server", use "list server*".
```

**8** 該方法沒有可監視的子元件。取得 getUserName 方法的所有可監視統計資訊。

```
asadmin> get -m
server.applications.petstore.signon-ejb_jar.UserEJB.bean-methods.getUserName.*
```

傳回：

```
server.applications.petstore.signon-ejb_jar.UserEJB.bean-methods.
getUserName.executiontime-count = 0
server.applications.petstore.signon-ejb_jar.UserEJB.bean-methods.
getUserName.executiontime-description = Provides the time in milliseconds
   spent during the last successful/unsuccessful attempt to execute the
   operation.
server.applications.petstore.signon-ejb_jar.UserEJB.bean-methods.
getUserName.executiontime-lastsampletime = 1079981809259
server.applications.petstore.signon-ejb_jar.UserEJB.bean-methods.
getUserName.executiontime-name = ExecutionTime
server.applications.petstore.signon-ejb_jar.UserEJB.bean-methods.
getUserName.executiontime-starttime = 1079980593137
server.applications.petstore.signon-ejb_jar.UserEJB.bean-methods.
getUserName.executiontime-unit = count
server.applications.petstore.signon-ejb_jar.UserEJB.bean-methods.
getUserName.methodstatistic-count = 0
server.applications.petstore.signon-ejb_jar.UserEJB.bean-methods.
getUserName.methodstatistic-description = Provides the number of times an
   operation was called, the total time that was spent during the
   invocation and so on.
server.applications.petstore.signon-ejb_jar.UserEJB.bean-methods.
getUserName.methodstatistic-lastsampletime = 1079980593137
server.applications.petstore.signon-ejb_jar.UserEJB.bean-methods.
getUserName.methodstatistic-maxtime = 0
server.applications.petstore.signon-ejb_jar.UserEJB.bean-methods.
getUserName.methodstatistic-mintime = 0
server.applications.petstore.signon-ejb_jar.UserEJB.bean-methods.
getUserName.methodstatistic-name = ExecutionTime
server.applications.petstore.signon-ejb_jar.UserEJB.bean-methods.
getUserName.methodstatistic-starttime = 1079980593137
```

```
server.applications.petstore.signon-ejb_jar.UserEJB.bean-methods.
getUserName.methodstatistic-totaltime = 0
server.applications.petstore.signon-ejb_jar.UserEJB.bean-methods.
getUserName.methodstatistic-unit =
 server.applications.petstore.signon-ejb_jar.UserEJB.bean-methods.
getUserName.totalnumerrors-count = 0
server.applications.petstore.signon-ejb_jar.UserEJB.bean-methods.
getUserName.totalnumerrors-description = Provides the total number of errors
   that occured during invocation or execution of an operation.
server.applications.petstore.signon-ejb_jar.UserEJB.bean-methods.
getUserName.totalnumerrors-lastsampletime = 1079981809273
server.applications.petstore.signon-ejb_jar.UserEJB.bean-methods.
getUserName.totalnumerrors-name = TotalNumErrors
server.applications.petstore.signon-ejb_jar.UserEJB.bean-methods.
getUserName.totalnumerrors-starttime = 1079980593137
server.applications.petstore.signon-ejb_jar.UserEJB.bean-methods.
getUserName.totalnumerrors-unit = count
server.applications.petstore.signon-ejb_jar.UserEJB.bean-methods.
getUserName.totalnumsuccess-count = 0
server.applications.petstore.signon-ejb_jar.UserEJB.bean-methods.
getUserName.totalnumsuccess-description = Provides the total number of
   successful invocations of the method.
server.applications.petstore.signon-ejb_jar.UserEJB.bean-methods.
getUserName.totalnumsuccess-lastsampletime = 1079981809255
server.applications.petstore.signon-ejb_jar.UserEJB.bean-methods.
getUserName.totalnumsuccess-name = TotalNumSuccess
server.applications.petstore.signon-ejb_jar.UserEJB.bean-methods.
getUserName.totalnumsuccess-starttime = 1079980593137
server.applications.petstore.signon-ejb_jar.UserEJB.bean-methods.
getUserName.totalnumsuccess-unit = count
```

**9 如果還需要取得執行時間等特定統計資訊，請使用如下指令：**

```
asadmin> get -m server.applications.petstore.signon-ejb_jar.
UserEJB.bean-methods.getUserName.executiontime-count
```

傳回：

```
server.applications.petstore.signon-ejb_jar.UserEJB.bean-methods.
getUserName.executiontime-count = 1
```

## list 和 get 指令在所有層級上的預期輸出

下表顯示了樹的各個層級的指令、含點名稱及相應的輸出。

**表 18–33** 頂層

| 指令 | 含點名稱 | 輸出 |
|---|---|---|
| `list -m` | `server` | `server.applicationsserver.thread-poolsserver.resourcesserver.http-serviceserver.transaction-serviceserver.orb.connection-managersserver.orb.connection-managers.orb\.Connections\.Inbound\.AcceptedConnectionsserver.jvm` |
| `list -m` | `server.*` | 此節點下的子節點的階層結構。 |
| `get -m` | `server.*` | 無輸出，但顯示一條訊息，說明此節點上沒有屬性。 |

下表列出了應用程式層級的指令、帶點名稱及對應的輸出。

**表 18–34** 應用程式層級

| 指令 | 含點名稱 | 輸出 |
|---|---|---|
| `list -m` | `server.applications`<br>或<br>`*applications` | `appl1app2web-module1_warejb-module2_jar...` |
| `list -m` | `server.applications.*`<br>或<br>`*applications.*` | 此節點下的子節點的階層結構。 |
| `get -m` | `server.applications.*`<br>或<br>`*applications.*` | 無輸出，但顯示一條訊息，說明此節點上沒有屬性。 |

下表列出了應用程式層級上的獨立模組和企業應用程式的指令、帶點名稱及對應的輸出。

**表 18–35** 應用程式 - 企業應用程式和獨立模組

| 指令 | 含點名稱 | 輸出 |
|---|---|---|
| `list -m` | `server.applications.app1`<br>或<br>`*app1`<br>備註：僅在已部署企業應用程式之後，此層級才可用。如果部署了獨立模組，則此層級不可用。 | `ejb-module1_jarweb-module2_warejb-module3_jarweb-module3_war...` |

表 18–35 應用程式 - 企業應用程式和獨立模組 *(續)*

| 指令 | 含點名稱 | 輸出 |
|---|---|---|
| `list -m` | `server.applications.app1.*`<br>或<br>`*app1.*` | 此節點下的子節點的階層結構。 |
| `get -m` | `server.applications.app1.*`<br>或<br>`*app1.*` | 無輸出，但顯示一條訊息，說明此節點上沒有屬性。 |
| `list -m` | `server.applications.app1.ejb-module1_jar`<br>或<br>`*ejb-module1_jar`<br>或<br>`server.applications.ejb-module1_jar` | `bean1bean2bean3...` |
| `list -m` | `server.applications.app1.ejb-module1_jar`<br>或<br>`*ejb-module1_jar`<br>或<br>`server.applications.ejb-module1_jar` | 此節點下的子節點的階層結構。 |
| `get -m` | `server.applications.app1.ejb-module1_jar.*`<br>或<br>`*ejb-module1_jar.*`<br>或<br>`server.applications.ejb-module1_jar.*` | 無輸出，但顯示一條訊息，說明此節點上沒有屬性。 |
| `list -m` | `server.applications.app1.ejb-module1_jar.bean1`<br>備註：在獨立模組中，不會顯示包含應用程式名稱 (此範例中爲 `app1`) 的節點。 | 子節點清單：<br>`bean-poolbean-cachebean-method` |
| `list -m` | `server.applications.app1.ejb-module1_jar.bean1`<br>備註：在獨立模組中，不會顯示包含應用程式名稱 (此範例中爲 `app1`) 的節點。 | 子節點的階層結構及該節點和所有後續子節點的所有屬性的清單。 |

表 18–35 應用程式 - 企業應用程式和獨立模組 (續)

| 指令 | 含點名稱 | 輸出 |
|---|---|---|
| get -m | server.applications.app1.ejb-module1_jar.bean1.*<br>備註：在獨立模組中，不會顯示包含應用程式名稱 (此範例中爲 app1) 的節點。 | 以下屬性及其關聯值：<br>CreateCount_CountCreateCount_<br>DescriptionCreateCount_<br>LastSampleTimeCreateCount_<br>NameCreateCount_<br>StartTimeCreateCount_<br>UnitMethodReadyCount_<br>CurrentMethodReadyCount_<br>DescriptionMethodReadyCount_<br>HighWaterMarkMethodReadyCount_<br>LastSampleTimeMethodReadyCount_<br>LowWaterMarkMethodReadyCount_<br>NameMethodReadyCount_<br>StartTimeMethodReadyCount_<br>UnitRemoveCount_CountRemoveCount_<br>DescriptionRemoveCount_<br>LastSampleTimeRemoveCount_<br>NameRemoveCount_StartTimeAttribute<br>RemoveCount_Unit |
| list -m | server.applications.app1.ejb-module1_jar.bean1.bean-pool<br>備註：在獨立模組中，不會顯示包含應用程式名稱 (此範例中爲 app1) 的節點。 | 無屬性，但顯示一條訊息，說明：server.applications.appl.ejb-module1_jar.bean1-cache 上沒有要列出的內容。若要取得以字串開始的有效名稱，請使用萬用字元 (*)。例如，若要列出以 server 開始的所有名稱，請使用 list server*。 |
| get -m | server.applications.app1.ejb-module1_jar.bean1.bean-pool.*<br>備註：在獨立模組中，不會顯示包含應用程式名稱 (此範例中爲 app1) 的節點。 | 與 EJB 池屬性對應的屬性和值清單，如表 18–4 中所示。 |
| list -m | server.applications.app1.ejb-module1_jar.bean1.bean-cache<br>備註：在獨立模組中，不會顯示包含應用程式名稱 (此範例中爲 app1) 的節點。 | 無屬性，但顯示一條訊息，說明「使用帶有 --monitor 選項的 get 指令可以檢視此節點的屬性和值。」 |
| get -m | server.applications.app1.ejb-module1_jar.bean1.bean-cache.*<br>備註：在獨立模組中，不會顯示包含應用程式名稱 (此範例中爲 app1) 的節點。 | 與 EJB 快取屬性對應的屬性和值清單，如表 18–5 中所示。 |
| list -m | server.applications.app1.ejb-module1_jar.bean1.bean-method.method1<br>備註：在獨立模組中，不會顯示包含應用程式名稱 (此範例中爲 app1) 的節點。 | 無屬性，但顯示一條訊息，說明「使用帶有 --monitor 選項的 get 指令可以檢視此節點的屬性和值。」 |

表 18–35 應用程式 - 企業應用程式和獨立模組 (續)

| 指令 | 含點名稱 | 輸出 |
|---|---|---|
| get -m | server.applications.app1.ejb-module1_jar.bean1.bean-method.method1.*<br>備註：在獨立模組中，不會顯示包含應用程式名稱 (此範例中爲 app1) 的節點。 | 與 EJB 方法屬性對應的屬性和值清單，如表 18–2 中所示。 |
| list -m | server.applications.app1.web-module1_war | 顯示指定給模組的虛擬伺服器。 |
| get -m | server.applications.app1.web-module1_war.* | 無輸出，但顯示一條訊息，說明此節點上沒有屬性。 |
| list -m | server.applications.app1.web-module1_war.virtual_server | 顯示已註冊的 Servlet 清單。 |
| get -m | server.applications.app1.web-module1_war.virtual_server.* | 無輸出，但顯示一條訊息，說明此節點上沒有屬性。 |
| list -m | server.applications.app1.web-module1_war.virtual_server.servlet1 | 無屬性，但顯示一條訊息，說明「使用帶有 --monitor 選項的 get 指令可以檢視此節點的屬性和值。」 |
| get -m | server.applications.app1.web-module1_war.virtual_server.servlet1.* | 與 Web 容器 (Servlet) 屬性對應的屬性和值清單，如表 18–7 中所示。 |

下表列出了 HTTP 服務層級的指令、帶點名稱及對應的輸出。

表 18–36 HTTP 服務層級

| 指令 | 含點名稱 | 輸出 |
|---|---|---|
| list -m | server.http-service | 虛擬伺服器清單。 |
| get -m | server.http-service.* | 無輸出，但顯示一條訊息，說明此節點上沒有屬性。 |
| list -m | server.http-service.server | HTTP 偵聽程式清單。 |
| get -m | server.http-service.server.* | 無輸出，但顯示一條訊息，說明此節點上沒有屬性。 |
| list -m | server.http-service.server.http-listener1 | 無屬性，但顯示一條訊息，說明「使用帶有 --monitor 選項的 get 指令可以檢視此節點的屬性和值。」 |
| get -m | server.http-service.server.* | 與 HTTP 服務屬性對應的屬性和值清單，如表 18–9 中所示。 |

下表列出了執行緒池層級的指令、帶點名稱及對應的輸出。

**表 18–37** 執行緒池層級

| 指令 | 含點名稱 | 輸出 |
|---|---|---|
| `list -m` | `server.thread-pools` | 執行緒池名稱清單。 |
| `get -m` | `server.thread-pools.*` | 無輸出，但顯示一條訊息，說明此節點上沒有屬性。 |
| `list -m` | `server.thread-pools.orb\.threadpool\.thread-pool-1` | 無屬性，但顯示一條訊息，說明「使用帶有 --monitor 選項的 get 指令可以檢視此節點的屬性和值。」 |
| `get -m` | `server.thread-pools..orb\.threadpool\.thread-pool-1.*` | 與執行緒池屬性對應的屬性和值清單，如表 18–14 中所示。 |

下表列出了資源層級的指令、帶點名稱及對應的輸出。

**表 18–38** 資源層級

| 指令 | 含點名稱 | 輸出 |
|---|---|---|
| `list -m` | `server.resources` | 池名稱清單。 |
| `get -m` | `server.resources.*` | 無輸出，但顯示一條訊息，說明此節點上沒有屬性。 |
| `list -m` | `server.resources.jdbc-connection-pool-pool.connection-pool1` | 無屬性，但顯示一條訊息，說明「使用帶有 --monitor 選項的 get 指令可以檢視此節點的屬性和值。」 |
| `get -m` | `server.resources.jdbc-connection-pool-pool.connection-pool1.*` | 與連線池屬性對應的屬性和值清單，如表 18–10 中所示。 |

下表列出了作業事件服務層級的指令、帶點名稱及對應的輸出。

**表 18–39** 作業事件服務層級

| 指令 | 含點名稱 | 輸出 |
|---|---|---|
| `list -m` | `server.transaction-service` | 無屬性，但顯示一條訊息，說明「使用帶有 --monitor 選項的 get 指令可以檢視此節點的屬性和值。」 |
| `get -m` | `server.transaction-service.*` | 與作業事件服務屬性對應的屬性和值清單，如表 18–15 中所示。 |

下表列出了 ORB 層級的指令、帶點名稱及對應的輸出。

表 18–40 ORB 層級

| 指令 | 含點名稱 | 輸出 |
|---|---|---|
| `list -m` | `server.orb` | `server-orb.connection-managers` |
| `get -m` | `server.orb.*` | 無輸出，但顯示一條訊息，說明此節點上沒有屬性。 |
| `list -m` | `server.orb.connection-managers` | ORB 連線管理員的名稱。 |
| `get -m` | `server.orb.connection-managers.*` | 無輸出，但顯示一條訊息，說明此節點上沒有屬性。 |
| `list -m` | `server.orb.connection-managers.orb\.Connections\.Inbound\.AcceptedConnections` | 無屬性，但顯示一條訊息，說明「使用帶有 --monitor 選項的 get 指令可以檢視此節點的屬性和值。」 |
| `get -m` | `server.orb.connection-managers.orb\.Connections\.Inbound\.AcceptedConnections.` | 與 ORB 連線管理員屬性對應的屬性和值清單，如表 18–13 中所示。 |

下表列出了 JVM 層級的指令、帶點名稱及對應的輸出。

表 18–41 JVM 層級

| 指令 | 含點名稱 | 輸出 |
|---|---|---|
| `list -m` | `server.jvm` | 無屬性，但顯示一則訊息，說明「使用帶有 `--monitor` 選項的 get 指令可以檢視此節點的屬性和值。」 |
| `get -m` | `server.jvm.*` | 與 JVM 屬性對應的屬性和值清單，如表 18–16 中所示。 |

# 使用 JConsole

本小節包含下列主題：

- 第 201 頁的 「保護 JConsole 與 Application Server 間連線的安全」
- 第 202 頁的 「將 JConsole 連線至 Application Server 的必要條件」
- 第 202 頁的 「將 JConsole 連線至 Application Server」
- 第 203 頁的 「以安全的方式將 JConsole 連線至 Application Server」

Application Server 的管理 (管理與監視) 是以 JMX 技術爲基礎的。這表示受管理的元件會以 Application Server 之 JVM 中執行的 MBeanServer 之 MBean 表示。

Java SE 5 增強 JVM 管理與監視功能的方法，是加入 Platform MBean Server 以及加入 MBean 來配置 VM。Application Server 會利用這些增強功能，並使用 Platform MBean Server 註冊其 MBean。因此 JMX 連接器用戶端可統一檢視 JVM MBean 以及 Application Server MBean。

爲了檢視所有 MBean，Application Server 提供一個名爲「系統 JMX 連接器伺服器」的標準 JMX 連接器伺服器配置。在 Application Server 啓動過程中，會啓動此 JMX 連接器伺服器的實例。任何相容的 JMX 連接器用戶端都可以使用此連接器伺服器連線到伺服器。

Java SE 也提供一項工具，可連線 MBean 伺服器及檢視註冊到此伺服器的 MBean。JConsole 就是其中一種常用的 JMX 連接器用戶端，並隨附在標準 Java SE 發行軟體中。如需有關 JConsole 的更多詳細資訊，請參閱
`http://java.sun.com/javase/6/docs/technotes/guides/management/jconsole.html`

當您配置 JConsole 搭配 Application Server 使用時，Application Server 會成爲 JMX 連接器的伺服器端，而 JConsole 會成爲 JMX 連接器慣用的用戶端。第 202 頁的 「將 JConsole 連線至 Application Server」 shows how to make a successful connection .

# 保護 JConsole 與 Application Server 間連線的安全

對 Application Server 或任何 JMX 連接器伺服器端的連線方式，會因連線的傳輸層安全性而有些許差異。若伺服器端是安全的 (保證傳輸層安全性)，則必須在客戶機端進行一些額外配置。

- 依預設，Application Server 的開發者設定檔是使用不安全的系統 JMX 連接器伺服器進行配置的。
- 依預設，Application Server 的叢集和企業設定檔是使用安全的系統 JMX 連接器伺服器進行配置的。
- 用於通訊的協定是 RMI/JRMP。若啓用了 JMX 連接器的安全性，則使用的協定是 RMI/JRMP over SSL。

---

**備註 –** RMI over SSL 未提供額外的檢查，因此無法確定用戶端是否與預定的伺服器通訊。因此，使用 JConsole 時，可能會發生將使用者名稱與密碼傳送到惡意主機的情況。確認安全性是否受影響完全是管理員的責任。

---

當您在例如 `appserver.sun.com` 的機器上安裝開發者設定檔網域時，會在 Domain Administration Server (DAS) `domain.xml` 檔案中看到以下內容：

```
<!- – The JSR 160 "system-jmx-connector" – –><jmx-connector accept-all="false"
address="0.0.0.0" auth-realm-name="admin-realm" enabled="true" name="system"
port="8686" protocol="rmi_jrmp" security-enabled="false"/> <!- – The JSR 160
"system-jmx-connector" – –>
```

JMX 連接器的 `security-enabled` 旗標爲 *false*。如果是執行叢集或企業設定檔，或已在開發者設定檔中開啓 JMX 連接器的安全性，則此旗標會設定爲 *true*。

```
<!- – The JSR 160 "system-jmx-connector" – –><jmx-connector accept-all="false"
address="0.0.0.0" auth-realm-name="admin-realm" enabled="true" name="system"
port="8686" protocol="rmi_jrmp" security-enabled="true"/> ...</jmx-connector><!-
– The JSR 160 "system-jmx-connector" – –>
```

# 將 JConsole 連線至 Application Server 的必要條件

JConsole 設定分爲兩個部分：伺服器端與客戶機端。在此範例中，Application Server 網域會安裝在名爲 `appserver.sun.com` 的機器上，此機器是功能強大的 Solaris 伺服器。這是伺服器端。

客戶機端也必須安裝 Application Server。接著，我們假設用戶端是已安裝 Java SE 6.0 與 Application Server 的 Windows 機器。

---

**備註 –** 只有當您的 Application Server 網域已在遠端機器上啓用安全性時 (叢集和企業設定檔的預設值)，才需要在用戶端上安裝 Application Server。若只是要在上述 Solaris 機器管理 Application Server 開發者設定檔網域，則不需要在此用戶端機器上安裝 Application Server。

---

若伺服器與客戶機端位於相同的機器上，您可以使用 `localhost` 來指定主機名稱。

# ▼ 將 JConsole 連線至 Application Server

此程序說明未在 JMX 連接器上啓用安全性的情況下，如何將 JConsole 連線至 Application Server。依預設，不會在 Application Server 上針對開發者設定檔啓用安全性。

1. **啓動 `appserver.sun.com` 上的網域**。
2. **執行 *JDK_HOME*`/bin/jconsole` 以啓動 JConsole**。
3. **在 JConsole 的 [連線至代理程式] 標籤中，輸入使用者名稱、密碼、主機名稱與連接埠 (預設為 8686)**。

   使用者名稱指的是網域管理使用者名稱，密碼指的是網域管理密碼。
4. **按一下 [連線]**。

   在 JConsole 視窗中，您將會在不同標籤中看到所有 MBean 與 VM 資訊等。

## ▼ 以安全的方式將 JConsole 連線至 Application Server

此程序說明在 JMX 連接器上啓用了安全性的情況下，如何將 JConsole 連線至 Application Server。依預設，會在 Application Server 叢集或企業設定檔上啓用安全性。若已在開發者設定檔的 JMX 連接器上啓用安全性，請使用此程序。

1. **在用戶端機器 (安裝 JConsole 的機器) 上安裝 Application Server。**

   只有在需要讓 JConsole 知道您信任之 Domain Administration Server 的伺服器憑證位置時，才需要執行此步驟。若要取得該憑證，請呼叫至少一個 *remote* `asadmin` 指令 (爲呼叫此指令，您必須在本機安裝 Application Server)。

2. **在 `appserver.sun.com` 上啓動 Application Server。**

   由於這是叢集或企業網域，所以系統 JMX 連接器伺服器是安全的。若要在開發者設定檔 JMX 連接器上啓用安全性，請參閱管理主控台線上說明。

3. **從本機 Application Server 安裝中，執行** *install-dir*`\bin\asadmin list --user admin --secure=true --host appserver.sun.com --port 4848` **(其中，4848 是伺服器的管理連接埠)。**

   雖然我們在此範例中選擇 `asadmin list` 指令，您仍可以執行任何遠端 `asadmin` 指令。系統會提示您接受 `appserver.sun.com` 之 DAS 所傳送的憑證。

4. **按下 `y` 接受 `appserver.sun.com` 之 DAS 所傳送的憑證。**

   伺服器憑證會儲存在名爲 `.asadmintruststore` 的檔案中 (此檔案位於用戶端機器的主目錄中)。

   ---

   **備註 –** 若您的伺服器機器與用戶端機器是同一部 (也就是說，您也在 `appserver.sun.com` 上執行 JConsole)，則不需要執行此步驟。

   ---

5. **使用以下 JConsole 指令，讓 JConsole 知道信任清單存放區位置：**

   *JDK-dir*`\bin\jconsole.exe -J-Djavax.net.ssl.trustStore="C:\Documents and Settings\`*user*`\.asadmintruststore"`

6. **執行** *JDK_HOME*`/bin/jconsole` **以啓動 JConsole**

7. **在 JConsole 的 [連線至代理程式] 標籤中，輸入使用者名稱、密碼、主機名稱與連接埠 (預設為 8686)。**

   使用者名稱指的是網域管理使用者名稱，密碼指的是網域管理密碼。

8. **按一下 [連線]。**

   在 JConsole 視窗中，您將會在不同標籤中看到所有 MBean 與 VM 資訊等。

◆ ◆ ◆ 第 19 章

# 19 配置管理規則

本節包含有關設定管理策略以自動化日常管理作業、爲不同的執行階段條件配置 Application Server 的自我調校，以及藉由防止故障以增強可用性的資訊。本節也包含有關自我管理範本的資訊，這些範本爲可自訂的預先定義的管理規則。

本小節包含下列主題：



## 關於管理規則

管理規則可讓您自動化日常管理作業、爲不同的執行階段條件配置 Application Server 的自我調校，以及藉由防止故障以增強可用性。管理規則包含發生指定的事件或達到設定的臨界值時，應採取的動作。您可以依據指定的事件，設定可自動採取更正動作的管理規則。

管理規則由兩部分組成 — 事件和動作：

- **事件** 使用 JMX 通知機制觸發預先定義的動作。
- 發生關聯的事件時，會觸發**動作**。 動作是一種 MBean，是實作 `javax.management.NotificationListener` 的通知偵聽程式。

例如，事件可以是 EJB 記錄程式所記錄的 SEVERE 訊息，而動作可以透過記錄訊息內容發送警示給管理員。發生事件時，事件資料會附在 `javax.management.Notification` 的 `userData` 其中傳送。

規則中指定的動作必須實作爲自訂 MBean。因此，在配置管理規則之前，應該部署自訂 MBean 接收事件通知，並採取適當的動作。如需有關開發與部署自訂 MBean 的詳細資訊，請參閱「Sun Java System Application Server 9.1 Developer's Guide」中的第 14 章「Developing Custom MBeans」。

Application Server 提供一些有用的事件，您可以撰寫自訂 MBean 以發出通知，延伸這些事件的功能。您可以透過變更每個事件的特性，進一步自訂每個事件。

以下是可以使用的事件類型：

- 監視事件：監視 MBean 的屬性。監視事件與 `javax.management.monitor` 套裝軟體具有類似的功能。除了監視簡單的屬性外，監視事件也能像 Java SE 5 `javax.management.monitor` 一樣，支援監視複雜的屬性。
- 通知事件：來自自訂 MBean 的事件通知。使用這些事件可撰寫自訂事件，藉此延伸事件字典。可發出通知的任何 MBean 都可以是事件。
- 系統事件：
  - 生命週期：伺服器啓動、關機和終止的事件。
  - 記錄：指定的記錄程式撰寫記錄項目時所觸發的事件。例如，您可以建立管理規則，在 EJB 容器記錄程式記錄 SEVERE 記錄項目時，傳送警示給管理員。
  - 計時器：在指定的日期和時間、指定的間隔等時機所觸發的事件。這些事件與 `javax.management.timer` 套裝軟體具有類似的功能。
  - 追蹤：進入和結束 HTTP/IIOP 請求方法、EJB 方法和 Web 方法時所觸發的事件。例如，您可以設計 servlet 篩選，使用 Web 方法的入和結束事件，將與 servlet 的互動記錄爲管理規則。
  - 叢集：叢集或實例啓動、停止或故障時所觸發的事件。這些事件會使用群組管理系統叢集監視。

# 配置管理規則

若要在「管理主控台」中配置管理規則：

- 在開發者設定檔中，移至 [配置] → [管理規則]
- 在叢集和企業設定檔中，移至 [配置] → [配置] → [管理規則]

---

**備註 –** 在此頁面上，核取 [啓用所有規則] 以全域啓用管理規則。如果沒有全域啓用管理規則，將不會執行任何管理規則。

---

此外，若要啓用個別的管理規則，則必須在此頁面上按一下規則旁邊的方塊，並按一下 [啓用] 以啓用規則。

也必須在目標上啓用規則的 MBean。若要啓用 MBean，請移至 [自訂 MBean] → [MBean]。在 [編輯自訂 MBean] 頁面上，按一下 [目標] 標籤以存取 [自訂 MBean 目標] 頁面，您可以在此頁面上啓用部分或所有目標上的 MBean。

如需詳細資訊，請參閱線上說明。

若要從指令行建立管理規則，請使用 `create-management-rule` 指令。若要設定管理規則的特性，請使用 `get` 和 `set` 指令。若要列出和刪除管理規則，請使用 `list-management-rules` 和 `delete-management-rule`。如需更多資訊，請參閱這些指令的線上說明，或參閱「Sun Java System Application Server 9.1 Reference Manual」。

◆ ◆ ◆ 第 20 章

# 20 Java 虛擬機器和進階設定

Java 虛擬機器 (JVM) 爲解譯的運算引擎，負責執行已編譯 Java 程式中的位元碼。JVM 會將 Java 位元碼轉譯爲主機電腦的本機指令。Application Server 是 Java 程序，因此需要 JVM 才能執行，並支援在其上執行的 Java 應用程式。JVM 設定爲 Application Server 配置的一部分。

本章說明如何配置 Java 虛擬機器 (JVM™) 和其他進階設定。它包含以下小節：



## 調校 JVM 設定

定義相關設定以強化 Java 虛擬機器的使用，也屬於 Application Server 配置的一部分。若要使用管理主控台變更 JVM 配置，請選取 [Application Server] > [JVM 設定] 標籤，並依照以下說明定義一般 JVM 設定：

- Java 首頁：輸入 Java 軟體安裝目錄的名稱。Application Server 依賴 Java SE 軟體。

  **備註 –** 如果輸入不存在的目錄名稱或不受支援的 Java EE 軟體版本的安裝目錄名稱，則 Application Server 將無法啓動。

- Javac 選項：爲 Java 程式設計語言編譯器輸入指令行選項。部署 EJB 元件後，Application Server 將執行編譯器。
- 除錯：若要設定以 JPDA (Java 平台除錯程式架構) 進行除錯，請選取這個 [已啓用] 核取方塊。

  JPDA 由應用程式開發者使用。
- 除錯選項：指定在啓用除錯功能的情況下，要傳遞至 JVM 的 JPDA 選項。
- RMI 編譯選項：爲 `rmic` 編譯器輸入指令行選項。部署 EJB 元件後，Application Server 將執行 `rmic` 編譯器。

- 位元碼前處理器：輸入以逗號分隔的類別名稱清單。每個類別都必須實作 `com.sun.appserv.BytecodePreprocessor` 介面。將按指定次序呼叫這些類別。

  效能評測器等工具也許需要 [位元碼預處理程式] 欄位中的項目。效能評測器產生用於分析伺服器效能的資訊。

# 配置進階設定

若要使用管理主控台設定進階應用程式配置，請選取 [Application Server] > [進階] 標籤 > [應用程式配置] 標籤，並依下列指示設定應用程式配置：

- 重新載入：選取此核取方塊，以啓用動態重新載入應用程式。

  啓用動態重新載入之後 (預設爲啓用)，當您變更應用程式或模組的程式碼或部署描述元時，就無須重新部署該應用程式或模組。您只需要將變更過的 JSP 或類別檔案複製到應用程式或模組的部署目錄。伺服器會定期檢查變更，並以自動且動態的方式，重新以變更項目部署應用程式。動態重新載入在開發環境中非常有用，因爲它能快速測試程式碼變更。但在生產環境中，動態重新載入可能會使效能降低。另外，每當重新載入完成時，轉換時間內的階段作業都會變得無效。用戶端必須重新啓動階段作業。
- 重新載入輪詢間隔：爲應用程式和模組定義檢查程式碼變更的間隔，以及動態重新載入的間隔。預設值爲 2。
- 管理階段作業逾時：指定管理階段作業可在靜止幾分鐘後逾時。

另外，請依下列指示定義部署設定：

- 自動部署：選取此核取方塊，以啓用自動部署應用程式。

  自動部署包含將應用程式或模組檔案 (JAR、WAR、RAR 或 EAR) 複製到特定的目錄中，由 Application Server 在此進行自動部署。
- 自動部署輪詢間隔：爲應用程式和模組定義檢查程式碼變更的間隔，以及動態重新載入的間隔。預設值爲 2。
- 檢驗器：核取 [啓用檢驗器] 方塊，以驗證您的部署描述元檔案。這是選擇性設定。
- 預編譯：核取 [啓用預編譯] 方塊，以預編譯所有 JSP 檔案。

◆ ◆ ◆ 附錄 A

# 自動重新啓動網域或節點代理程式

如果網域或節點代理程式意外停止 (例如，您需要重新啓動機器)，您可以將系統配置爲自動重新啓動網域或節點代理程式。

本附錄包含下列主題：



## 在 Solaris 10 上自動重新啓動

Solaris 10 使用者可使用指令 `asadmin create-service`，建立會重新啓動節點代理程式或 Domain Administration Server (DAS) 的服務。所建立的服務可使用 Solaris 服務管理功能 (SMF)。

服務啓動的程序則取決於該服務是重新啓動 DAS 還是節點代理程式。

- 如果服務重新啓動 DAS，則程序是 `asadmin start-domain`。
- 如果服務重新啓動節點代理程式，則程序是 `asadmin start-node-agent`。

此服務授予此程序的權限，是執行此程序的使用者權限。當您使用指令 `asadmin create-service` 來建立 SMF 服務，預設使用者是超級使用者。如果需要不同的使用者來執行程序，請在 `method_credential` 中指定使用者。

如果此程序連結到 Solaris 作業系統的權限連接埠，則此程序需要 `net_privaddr` 權限。Solaris 作業系統的權限連接埠具有小於 1024 的連接埠號碼。

若要判斷使用者是否具有 `net_privaddr` 權限，請以該使用者身份登入，然後鍵入指令 `ppriv -l | grep net_privaddr`。

若要執行 `asadmin create-service` 指令，必須具有 `solaris.smf.*` 授權。請參閱 `useradd` 和 `usermod` 線上手冊來瞭解設定授權的方法。您也必須具有目錄樹的寫入權

限：/var/svc/manifest/application/SUNWappserver。超級使用者通常具有這兩種權限。此外 PATH 中必須有 Solaris 10 管理指令 (例如 svccfg、svcs 和 auths)。如需執行此指令的完整資訊，請參閱 create-service(1)。

語法如下所示：

```
asadmin create-service [--name service-name] [--type das|node-agent]
--passwordfile password-file [--serviceproperties serviceproperties]
domain-or-node-agent-configuration-directory
```

例如，若要爲 domain1 建立名爲 domain1 的服務：

1. 請執行下列指令：

   **asadmin create-service --type das --passwordfile password.txt /appserver/domains/domain1**

   此指令會建立服務以自動重新啓動網域 domain1。在背景中，指令會從範本建立清單檔案並驗證檔案，然後將檔案匯入成服務。

   ---

   **備註 –** 如果特定的 Application Server 網域不應具有預設的使用者權限，請修改服務清單並重新匯入服務。若要判斷使用者權限，請以該使用者的身份登入，然後鍵入指令 ppriv -l。

   ---

2. 建立服務後，請使用 svacdm enable 指令啓用服務：

   **svacdm enable /appserver/domains/domain1**

3. 啓用後，如果網域中斷，SMF 將予以重新啓動。

管理服務時，下列 Solaris 指令很有用：

- auths
- smf_security
- svcadm
- svccfg
- rbac
- useradd
- usermod

如需這些指令的更多資訊，請參閱指令線上手冊。

# 在 Solaris 9 和 Linux 平台上使用 inittab 自動重新啓動

若要在 Solaris 9 或 Linux 平台上重新啓動網域，請在 `/etc/inittab` 檔案中增加一行文字。

如果您使用 `/etc/rc.local` 或系統的等效檔案，請在 `/etc/rc.local` 中增加一行，以呼叫所需的 `asadmin` 指令。

例如，若要重新啓動安裝在 `opt/SUNWappserver` 目錄中的 Application Server 之 `domain1`，請使用名爲 `password.txt` 的密碼檔案：

```
das:3:respawn:/opt/SUNWappserver/bin/asadmin start-domain --user admin
--passwordfile /opt/SUNWappserver/password.txt domain1
```

將這些文字放在一行上。前三個字母是程序的唯一指示符，可以進行更改。

重新啓動節點代理程式的語法與此相似。例如，若要重新啓動安裝在 `opt/SUNWappserver` 目錄中的 Application Server 之 `agent1`，請使用名爲 `password.txt` 的密碼檔案：

```
das:3:respawn:/opt/SUNWappserver/bin/asadmin start-node-agent --user admin
--passwordfile /opt/SUNWappserver/password.txt agent1
```

# 在 Microsoft Windows 平台上自動重新啓動

若要在 Microsoft Windows 上自動重新啓動，請建立 Windows 服務，並防止在使用者登出時關閉服務。

## 建立 Windows 服務

將 Sun Java System Application Server 隨附的 `appservService.exe` 和 `appserverAgentService.exe` 可執行檔與 Microsoft 提供的服務控制指令 (`sc.exe`) 配合使用。

`sc.exe` 指令隨附於 Windows XP，並位於 Windows 安裝目錄的 `system32` 子目錄 (通常爲 `C:\windows\system32` 或 `C:\winnt\system32`) 中。編寫本文件時，已可以從 `ftp://ftp.microsoft.com/reskit/win2000/sc.zip` 下載 Windows 2000 `sc.exe`。如需有關使用 `sc.exe` 的更多資訊，請參閱 http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dndllpro/html/msdn_scmslite.asp。

使用 `appservService.exe` 和 `appservAgentService.exe`，如下所示：

```
C:\winnt\system32\sc.exe create service-name binPath= \"fully-qualified-path-to-appservService.exe
\"fully-qualified-path-to-asadmin.bat start-command\"
\"fully-qualified-path-to-asadmin.bat stop-command\""
start= auto DisplayName= "display-name"
```

**備註 –** binpath 與等號 (=) 之間沒有空格。等號之後與路徑之前必須有空格。

例如，若要建立可以啓動和停止網域 domain1 的名爲 SunJavaSystemAppServer DOMAIN1 的服務，請使用密碼檔案 C:\Sun\AppServer\password.txt：

```
C:\windows\system32\sc.exe create domain1 binPath=
"C:\Sun\AppServer\lib\appservService.exe \"C:\Sun\AppServer\bin\asadmin.bat
start-domain --user admin --passwordfile C:\Sun\AppServer\password.txt domain1\"
\"C:\Sun\AppServer\bin\asadmin.bat stop-domain domain1\"" start= auto
DisplayName= "SunJavaSystemAppServer DOMAIN1"
```

若要建立可以啓動和停止節點代理程式 agent1 的服務，請使用：

```
C:\windows\system32\sc.exe create agent1 binPath=
"C:\Sun\AppServer\lib\appservAgentService.exe \"C:\Sun\AppServer\bin\asadmin.bat
start-node-agent --user admin --passwordfile C:\Sun\AppServer\password.txt agent1\"
\"C:\Sun\AppServer\bin\asadmin.bat stop-node-agent agent1\"" start= auto
DisplayName= "SunJavaSystemAppServer AGENT1"
```

**備註 –** 做爲 binPath= 參數的一部分而輸入的啓動和停止指令必須具有正確的語法。請在指令提示符號下執行這些指令以進行測試。如果指令無法正確啓動或停止網域或節點代理程式，則說明該服務無法正常工作。

**備註 –** 請勿將服務與 asadmin start 和 stop 指令混合使用來進行啓動和停止。否則，將導致伺服器狀態不同步。例如，即使元件沒有執行，該服務可能也會顯示已啓動元件。爲避免發生這種情況，請在使用服務時始終使用 sc.exe 指令啓動和停止元件。

如果 sc.exe create 指令並未正確地建立服務，請刪除服務後再試一次。若要刪除服務，請使用 sc.exe delete "*service-name*" 指令。

# 防止使用者登出時服務關閉

依預設，Java VM 會從 Windows 擷取訊號，這些訊號會指示作業系統正在關閉或使用者正在登出，並會自行正常關閉。此運作方式會導致在使用者登出 Windows 時，Application Server 服務關閉。若要防止在使用者登出時服務關閉，請設定 -Xrs Java VM 選項 (http://java.sun.com/j2se/1.3/docs/tooldocs/solaris/java.html#Xrs)。

要設定 -Xrs Java VM 選項，請在 *as-install*\domains*domain-name*\config\domain.xml 檔案定義 Java VM 選項的區段中增加下一行：

```
<jvm-options>-Xrs</jvm-options>
```

如果 Application Server 服務正在執行，請停止並重新啓動該服務，讓變更生效。

---

**備註 –** 在一些 Windows 2003 Server 安裝中，將 `-Xrs` 選項增加到 `domain.xml` 檔案並不能防止服務關閉。在此情況下，請將選項增加到 *as-install*`\lib\processLauncher.xml` 檔案，如下所示：

```
<process name="as-service-name">
   ...
   <sysproperty key="-Xrs"/>
   ...
```

---

# 自動重新啓動的安全性

如果使用叢集或企業設定檔，則在自動重新啓動 Application Server 時，需要管理密碼和主密碼。如果使用開發者設定檔，則不需要密碼

以下列其中一種方式，處理叢集和企業設定檔的密碼和主密碼需求：

- 在 Microsoft Windows 上，將服務配置爲要求使用者輸入密碼。
  1. 在 [服務] 控制台中，按兩下您所建立的服務。
  2. 在 [特性] 視窗中，按一下 [登入] 標籤。
  3. 核取 [允許服務與桌面互動]，以在啓動元件時提示輸入所需的密碼。

     必須登入才能看到提示，鍵入的項目不會在螢幕上顯示出來。如果選擇使用服務，這種方法最爲安全，但需要進行使用者互動才能啓動服務。

     如果未設定 [與桌面互動] 選項，服務將保持「啓動擱置」狀態，並且將顯示爲掛機。中止該服務程序即可從此狀態中恢復。
- 在 Windows 或 UNIX 上，使用 `--savemasterpassword=true` 選項建立網域，並建立儲存管理密碼的密碼檔案。啓動元件時，使用 `--passwordfile` 選項指向包含密碼的檔案。

  例如：

  1. 建立具有已儲存主密碼的網域。在下面的語法中，系統將提示您輸入管理密碼和主密碼：

     ```
     asadmin create-domain --adminport 4848 --adminuser admin
     --savemasterpassword=true --instanceport 8080 domain1
     ```
  2. 在 Windows 上，建立使用密碼檔案的服務，以填入管理員密碼：

     ```
     C:\windows\system32\sc.exe create domain1 binPath=
     "C:\Sun\AppServer\lib\appservService.exe \"C:\Sun\AppServer\bin\asadmin.bat
     start-domain --user admin
     --passwordfile C:\Sun\AppServer\password.txt domain1\"
     ```

```
\"C:\Sun\AppServer\bin\asadmin.bat stop-domain domain1\"" start= auto
DisplayName= "SunJavaSystemAppServer DOMAIN1"
```

密碼檔案 `password.txt` 的路徑爲 `C:\Sun\AppServer\password.txt`。該檔案包含以下格式的密碼

```
AS_ADMIN_password=password
```

例如，對於密碼 `adminadmin`：

```
AS_ADMIN_password=adminadmin
```

3. 在 UNIX 上，請使用爲 `inittab` 檔案增加的行中的 `--passwordfile` 選項：

```
das:3:respawn:/opt/SUNWappserver/bin/asadmin start-domain --user admin
--passwordfile /opt/SUNWappserver/password.txt domain1
```

密碼檔案 `password.txt` 的路徑爲 `/opt/SUNWappserver/password.txt`。該檔案包含以下格式的密碼

```
AS_ADMIN_password=password
```

例如，對於密碼 `adminadmin`：

```
AS_ADMIN_password=adminadmin
```

◆ ◆ ◆ 附錄 B

# domain.xml 的含點名稱屬性

此附錄說明可用於描述 MBean 及其屬性的帶點名稱屬性。`domain.xml` 檔案中的每個元素均有對應的 MBean。由於使用這些名稱的語法是利用小數點號來分隔名稱，所以這些名稱稱爲「**帶點名稱**」。

本附錄包含下列主題：



## 頂層元素

`domain.xml` 檔案中的所有頂層元素均必須符合以下條件：

- 每個伺服器、配置、叢集或節點代理程式的名稱都必須是專屬名稱。
- 不能將伺服器、配置、叢集或節點代理程式命名爲 `domain`。
- 不能將伺服器實例命名爲 `agent`。

下表列出了頂層元素及其對應的含點名稱字首。

| 元素名稱 | 含點名稱字首 |
|---|---|
| applications | `domain.applications` |
| resources | `domain.resources` |
| configurations | `domain.configs` |
| servers | `domain.servers`<br>您可以用 *server-name* 名稱格式存取此元素中包含的每個伺服器。其中，*server-name* 是伺服器子元素的名稱屬性值。 |

| 元素名稱 | 含點名稱字首 |
|---|---|
| clusters | `domain.clusters`<br>您可以用 *cluster-name* 名稱格式存取此元素中包含的每個叢集。其中，*cluster-name* 是叢集子元素的名稱屬性值。 |
| node-agents | `domain.node-agents` |
| lb-configs | `domain.lb-configs` |
| system-property | `domain.system-property` |

別名包括兩個層級：

1. 透過第一層級的別名可以存取伺服器實例或叢集的屬性而無需 `domain.servers` 或 `domain.clusters` 前綴。因此，舉例來說，格式爲 `server1` 的帶點名稱可對映到 `domain.servers.server1` 帶點名稱 (其中 `server1` 爲伺服器實例)。
2. 第二層級的別名用於表示叢集或獨立伺服器實例 (目標) 的配置、應用程式和資源。

下表列出了以伺服器名稱或叢集名稱黃Y的含點名稱，這些含點名稱被別名化爲網域下的頂層名稱：

| 含點名稱 | 別名目標 | 註釋 |
|---|---|---|
| *target*`.applications.*` | `domain.applications.*` | 該別名解析爲僅由**目標**參照的應用程式。 |
| *target*`.resources.*` | `domain.resources.*` | 此別名可解析成所有 `jdbc-connection-pool`、`connector-connection-pool`、`resource-adapter-config` 和 *target* 參照的所有其他資源。 |

下表列出了以伺服器名稱或叢集名稱黃Y的含點名稱，這些含點名稱在伺服器或叢集所參考的配置中被別名化爲頂層名稱。

| 含點名稱 | 別名目標 |
|---|---|
| *target*`.http-service` | *config-name*`.http-service` |
| *target*`.iiop-service` | *config-name*`.iiop-service` |
| *target*`.admin-service` | *config-name*`.admin-service` |
| *target*`.web-container` | *config-name*`.web-container` |
| *target*`.ejb-container` | *config-name*`.ejb-container` |
| *target*`.mdb-container` | *config-name*`.mdb-container` |
| *target*`.jms-service` | *config-name*`.jms-service` |

| 含點名稱 | 別名目標 |
| --- | --- |
| *target*.log-service | *config-name*.log-service |
| *target*.security-service | *config-name*.security-service |
| *target*.transaction-service | *config-name*.transaction-service |
| *target*.monitoring-service | *config-name*.monitoring-service |
| *target*.java-config | *config-name*.java-config |
| *target*.availability-service | *config-name*.availability-service |
| *target*.thread-pools | *config-name*.thread-pools |

# 不能別名化的元素

不應對叢集實例進行別名化。若要取得叢集實例的系統特性，應依循以下方法使用帶點名稱屬性：domain.servers.*clustered-instance-name*.*system-property*，而非 *clustered-instance-name*.*system-property*。

◆ ◆ ◆ 附錄 C

# asadmin 公用程式

Application Server 包含一個名爲 asadmin 的指令行管理公用程式。您可以使用 asadmin 公用程式來啓動與停止 Application Server、管理使用者、資源和應用程式。

本章包括下列小節：

# asadmin 公用程式

您可以使用 asadmin 公用程式來執行 Application Server 的所有管理作業。您可以使用此 asadmin 公用程式來取代管理員介面。

asadmin 公用程式可呼叫子指令，以便指定您想要執行的操作或作業。子指令是區分大小寫的。短選項引數具有一個破折號 (-)；而長選項引數具有兩個破折號 (--)。選項可控制公用程式如何執行子指令。選項也須區分大小寫。大部分選項都需要引數值，但布林值選項除外 (此類型選項可將功能 [開啓] 或 [關閉])。運算元會跟在引數值後面，且兩者中間以空格、定位字元 (Tab) 或雙破折號 (--) 分隔。asadmin 公用程式會將選項與選項值後面的任何項目視爲運算元。

asadmin 可用在指令 shell 呼叫或多重指令模式 (也稱爲 multimode) 中。在指令 shell 呼叫中，您必須從指令 shell 呼叫 asadmin 公用程式。asadmin 會執行指令，然後結束。在多重指令模式中，您只需要呼叫一次 asadmin，其隨後即可接受多個指令，直到您結束 asadmin 並返回一般指令 shell 呼叫。在多重指令模式中，您設定的環境變數會用於所有後續指令，直到您結束 multimode。您可以傳送檔案或標準輸入 (管道) 中事先準備好的指令清單，以提供指令。此外，您可以從多重模式階段作業呼叫 multimode；一旦結束第二個多重模式環境，就會返回原始的多重模式環境。

您也可以在互動式或非互動式選項中執行 asadmin 公用程式。依預設，已啓用互動式選項。它會提示您輸入必要的引數。您可以在任何情況下，在指令 shell 呼叫中使用互動式選項。當您從指令提示符號一次執行一個子指令，以及從檔案執行 multimode 時，可以在 multimode 中使用互動式選項。multimode 中的子指令 (經由輸入串流的管道)，以及從其他程式呼叫的子指令，都無法以互動式選項執行。

本機子指令可在沒有管理伺服器的情況下執行。然而，使用者必須登入至主控網域的機器，才能執行子指令並擁有該安裝與網域目錄的存取 (權限)。遠端子指令的執行方式永遠是先連線到管理伺服器，然後在該處執行子指令。這時一定要有執行中的管理伺服器。 所有遠端子指令都需要下列選項：

- -u --user 授權的網域 Application Server 管理使用者名稱。
- --passwordfile 包含下列格式的網域 Application Server 密碼的檔案：**AS_ADMIN_PASSWORD=***password*。其中，*password* 是實際的管理員密碼。
- -H --host 正在執行網域 Application Server 的機器名稱。
- -p --port 正在偵聽管理請求的網域 Application Server 的連接埠號碼。Platform Edition 的預設連接埠號碼是 4848。
- -s --secure 若爲 true，則使用 SSL/TLS 與網域 Application Server 進行通訊。
- -t --terse 表示所有輸出資料都必須簡潔，即通常不會使用容易理解的句子，而會優先使用格式完整的資料以供程序檔使用。預設是 false。
- -e --echo 將此選項設定爲 true，則指令行敘述會回應在標準輸出上。預設是 false。
- -I --interactive 如果設定爲 true (預設值)，則只會提示必要的密碼選項。
- -h --help 顯示指令的說明文字。

對於可在本機或遠端執行的子指令，若已在環境或指令行中設定 `--host`、`--port`、`--user` 和 `--passwordfile` 選項中的任何一個選項，則此子指令會以遠端模式執行。此外，對於可在本機或遠端執行的子指令，若將 `--local` 選項設定爲 true，則將在本機執行子指令。此外，若在指令行或環境中未設定任何本機選項，則依預設該子指令會在本機執行。若將 `--local` 選項設定爲 true，則會覆寫本機的 `--host`、`--port`、`--user` 和 `--passwordfile` 設定 (即使已指定)。子指令將在本機模式中執行。

可在本機執行的子指令將接受 `--domain` 選項來指定想要的網域，如果只有一個網域，則會假設該網域爲預設網域。如果有多個網域，`--domain` 選項爲必要的選項。對於可在本機或遠端執行的子指令，在指定 `--host`、`--port`、`--user` 及 `--passwordfile` 選項的情況下在遠端執行時，將忽略 `--domain` 選項。如果子指令將在遠端模式中執行，則忽略 `--domain` 選項。請注意，每個網域有一個管理實例，因此在具有多個網域的一部機器中，本機執行必須指定網域，遠端執行則必須爲該網域的管理實例指定 `--host`、`--port`、`--user` 和 `--passwordfile` 選項。

爲了安全性考量，您可以從檔案設定子指令的密碼，而不要在指令行輸入該密碼。`--passwordfile` 選項會使用包含密碼的檔案。此檔案的有效內容如下：

**範例 C–1** 密碼檔案內容

```
AS_ADMIN_PASSWORD=value
AS_ADMIN_ADMINPASSWORD=value
AS_ADMIN_USERPASSWORD=value
AS_ADMIN_MASTERPASSWORD=value
```

如果已將 **`AS_ADMIN_PASSWORD`** 匯出至全域環境，則指定 `--passwordfile` 選項將產生使用 `--password` 選項的相關警告。取消設定 **`AS_ADMIN_PASSWORD`** 可防止此情況的發生。您無法透過指令行或環境變數傳播主密碼，但可在 `passwordfile` 中指定。

若要使用 `--secure` 選項，則必須使用 `set` 指令，在 `domain.xml` 的 `admin` `http-listener` 中啓用安全性 `--enabled` 旗標。使用 `asadmin` 子指令來建立和/或刪除時，必須重新啓動伺服器，新建的指令才會生效。使用 `start-domain` 指令來重新啓動伺服器。

若要在 Solaris 平台上存取 Application Server 指令行介面子指令的線上手冊，請將 `$AS_INSTALL/man` 增加到您的 `MANPATH` 環境變數中。

您可以呼叫 `--help` 選項，以取得任何 `asadmin` 公用程式子指令的完整用法資訊。如果指定子指令，則會顯示該子指令的用法資訊。若使用不含子指令的 `--help` 選項，則會顯示所有可用子指令的清單。

# 遠端指令的共用選項

所有遠端指令都需要以下共用選項：

**表 C–1** 遠端指令必需的選項

| 選項 | 定義 |
|---|---|
| `--host` | 正在執行網域管理伺服器的機器名稱。預設值是 localhost。 |
| `--port` | 用於進行管理的 HTTP/S 連接埠。您應將瀏覽器指向此連接埠，以便管理網域。例如，`http://localhost:4848`。Platform Edition 的預設連接埠號碼是 4848。 |
| `--user` | 授權的網域管理伺服器管理使用者名稱。若已使用 `asadmin login` 指令向網域認證，則針對此特定網域執行後續作業時，不需要指定 `--user` 選項。 |
| `--passwordfile` | `--passwordfile` 選項指定包含特殊格式密碼項目的檔案名稱。密碼項目必須具有 `AS_ADMIN_` 前綴，後面接著大寫字母的密碼名稱。<br><br>例如，若要指定網域管理伺服器密碼，請使用具有以下格式的項目：`AS_ADMIN_PASSWORD=`*password*，其中 *password* 是實際的管理員密碼。可指定的其他密碼包括 `AS_ADMIN_MAPPEDPASSWORD`、`AS_ADMIN_USERPASSWORD` 和 `AS_ADMIN_ALIASPASSWORD`。<br><br>所有遠端指令都必須指定管理密碼以向網域管理伺服認證 (透過 `--passwordfile` 或 `asadmin login`，或在指令提示符號以互動式方式完成)。`asadmin login` 指令只能用於指定管理密碼。對於必須爲遠端指令指定的其他密碼，請使用 `--passwordfile`，或在指令提示符號輸入。<br><br>若已使用 `asadmin login` 指令向網域認證，則針對此特定網域執行後續作業時，不需要透過 `--passwordfile` 選項指定管理密碼。不過，這只適用於 `AS_ADMIN_PASSWORD` 選項。當個別指令 (例如，`update-file-user`) 要求您輸入密碼時，您仍需要提供其他密碼 (例如，`AS_ADMIN_USERPASSWORD`)。<br><br>爲了安全性考量，`asadmin` 將不會讀取指定爲環境變數的密碼。 |
| `--secure` | 若設定爲 true，將使用 SSL/TLS 與網域管理伺服器通訊。 |
| `--interactive` | 如果設定爲 true (預設值)，將只提示您輸入必需的密碼選項。 |
| `--terse` | 表示任何輸出資料都必須簡潔，即通常不會使用人性化的句子，而會優先使用格式完整的資料以供程序檔使用。 預設是 false。 |
| `--echo` | 將此選項設定爲 true，則指令行敘述會回應在標準輸出上。預設是 false。 |
| `--help` | 顯示指令的說明文字。 |

# Multimode 指令

您可以使用 multimode 指令來處理 asadmin 指令。指令行介面會提示您指定指令、執行該指令、顯示該指令的結果，然後提示您執行下一個指令。此外，在此模式中設定的所有 asadmin 選項名稱會用於所有後續指令。您可以設定環境並執行指令，直到鍵入「exit」或「quit」結束 multimode。您可以傳送檔案或標準輸入 (管道) 中事先準備好的指令清單，以提供指令。您可以從 *multimode* 階段作業呼叫 multimode；一旦結束第二個 *multimode* 環境，就會返回原始的 *multimode* 環境。

# Get、Set 與 List 指令

asadmin get、set 與 list 指令共同爲 Application Server 的抽象階層提供瀏覽機制。共有兩種階層：configuration 和 monitoring，且這些指令適用於兩者。list 指令提供管理元件 (具有唯讀或可修改的屬性) 之完全合格的帶點名稱。

configuration 階層提供可修改的屬性；而來自 monitoring 階層的管理元件只有唯讀屬性。configuration 階層並非嚴格基於網域的模式文件；而 monitoring 階層則稍有不同。

使用 list 指令以存取所要階層中的特定管理元件。接著，呼叫 get 與 set 指令以取得管理元件屬性的名稱與值，或設定手邊管理元件屬性的值。使用萬用字元 (*) 選項可擷取符合指定之完全合格的帶點名稱的所有項目。請參閱範例，以進一步瞭解可能瀏覽的階層和管理元件。

Application Server 帶點名稱使用「.」(點號) 做爲分隔完整名稱各部分的分隔符。這類似在 UNIX 檔案系統中使用「/」字元，來分隔檔案的絕對路徑名稱層級。建立 get、set 與 list 指令接受的帶點名稱時，適用以下規則。請注意，某些特定指令採用某些額外的語義。

- 名稱中的兩個連續部分皆會以 . (點號) 來分隔。
- 名稱中的某部分通常可用來識別 Application Server 子系統和/或其特定實例。例如：web-container、log-service 與 thread-pool-1 等
- 若名稱本身的任何部分包含 . (點號)，則必須在它的前面加上 \ (反斜線) 以退出，這樣該「.」才不會被當成分隔符。
- * (星號) 可用在帶點名稱中的任何地方，且它的作用如同常規表示式中的萬用字元符號。此外，* 可摺疊帶點名稱的所有部分。您可以將「<classname>this.is.really.long.hierarchy </classname>」這類較長的帶點名稱縮寫爲「<classname>th*.hierarchy</classname>」。但請注意，必須一律使用 . 來分隔名稱的每個部分。
- 所有帶點名稱的頂層切換都是 --monitor 或 -m (必須在指定的指令行另行指定)。此切換是否存在，代表進行 Application Server 管理時選取哪一個階層：監視和配置。
- 若您知道精確的完整帶點名稱 (不含任何萬用字元符號)，則 list 與 get/set 的語義將有些許差異：

- list 指令會將此完整的帶點名稱視爲抽象階層中父系節點的完整名稱。爲 list 指令提供此名稱時，此指令只會傳回下一層子節點的名稱。例如，list server.applications.web-module 將列出已部署至該網域或預設伺服器的所有 Web 模組。
- get 和 set 指令會將這個完整的帶點名稱視爲節點 (其帶點名稱本身是移除此帶點名稱最後一個部分後所得的名稱) 屬性的完全合格的名稱，並取得/設定該屬性的值。若此屬性存在，則上述情況爲眞。您永遠不會遇到此情況，因爲爲了尋找階層中特定節點的屬性名稱，您必須使用萬用字元符號 *。例如，server.applications.web-module.JSPWiki.context-root* 將傳回已部署至該網域或預設伺服器之 Web 應用程式的環境根目錄。

list 指令是這三個指令的瀏覽能力基礎。若要使用 set 或 get 來設定或取得特定 Application Server 子系統的屬性，必須先知道其帶點名稱。list 指令可引導您找到該子系統的帶點名稱。例如，若要在大型檔案系統 (開頭爲 /) 中找出特定檔案的修改日期 (屬性)，可以使用上述指令。首先，您必須找出該檔案在檔案系統中的位置，然後檢視其屬性。因此，可用來瞭解 appserver 中階層的前兩個指令是：* list "*" 和 <command>* list * --monitor。請參閱 get、set 或 list 指令線上手冊，以辨別這些指令的排序輸出。

# 伺服器生命週期指令

伺服器生命週期指令指的是可用來建立/刪除或啓動/停止網域或實例的指令。

**表 C–2** 伺服器生命週期指令

| 指令 | 定義 |
|---|---|
| create-domain | 建立網域的配置。網域是管理名稱空間。每個網域都具有其配置，它是儲存在一組檔案中。您可以在指定的 Application Server 安裝中建立任何數目的網域 (各自擁有不同的管理身份)。網域可獨立存在，不需依賴其他網域。使用者只要可存取指定系統上的 asadmin 程序檔，即可建立網域並將其配置儲存在所選的資料夾中。依預設，網域配置會建立在 *install_dir*/domains 目錄中。您可以置換此位置，以便將配置儲存在其他地方。。 |
| delete-domain | 刪除指定的網域。該網域必須已經存在，且已停止使用。 |
| start-domain | 啓動網域。若未指定網域目錄，將啓動預設 *install_dir*/domains 目錄中的網域。若其中有兩個以上的網域，則必須指定 *domain_name* 運算元。 |
| stop-domain | 停止指定網域的 Domain Administration Server。 |
| restore-domain | 從備份目錄復原網域下的檔案。 |
| list-domains | 列出網域。若未指定網域目錄，會列出預設 *install_dir*/domains 目錄中的網域。若有一個以上的網域，則必須指定 *domain_name* 運算元。 |

表 C–2 伺服器生命週期指令 (續)

| 指令 | 定義 |
|---|---|
| backup-domain | 備份指定網域下的檔案。 |
| login | 可讓您登入網域。若已在不同的機器 (本機) 建立不同的 Application Server 網域，從這些機器中的任何一部呼叫 asadmin 即可管理位於其他地方 (遠端) 的網域。當您選擇特定機器做爲管理用戶端，且要使用它來管理多個網域與伺服器時，這個方法非常有用。用來管理位於其他地方之網域的 asadmin 指令稱爲遠端指令。asadmin login 指令可簡化管理此類遠端網域的作業。您只能以互動式模式執行 login 指令。它會提示您輸入管理使用者名稱與密碼。成功登入之後，會在該使用者的主目錄建立 .asadminpass 檔案。該檔案即爲使用 --savelogin 選項時 create-domain 指令會修改的檔案。網域必須正在執行中，您才能執行此指令。 |
| create-instance | 建立本機或遠端機器上常駐的新伺服器實例。 |
| delete-instance | 刪除伺服器實例。您可在遠端或本機上執行此指令。使用者需以管理伺服器的密碼進行認證。此外，實例必須已存在於管理伺服器所服務的網域中。請謹慎使用此指令，因爲其具有破壞性且無法還原。 |

# List 與 Status 指令

list 與 status 指令可顯示已部署之元件的狀態。

表 C–3 List 與 Status 指令

| 指令 | 定義 |
|---|---|
| show-component-status | 取得已部署之元件的狀態。狀態是由伺服器傳回的字串表示。可能的狀態字串包含「status of *app-name* is enabled」或「status of *app-name* is disabled」。 |
| list-components | 列出所有已部署的 Java EE 5 元件。若未指定 --type 選項，將列出所有元件。 |
| list-sub-components | 列出已部署之模組或已部署之應用程式的模組中的 EJB 或 Servlet。若未指定模組，將列出所有模組。 |
| enable | 啓用指定的元件。若指定的元件已啓用，將重新啓用該元件。元件必須先經部署後才能啓用。若尚未部署元件，將傳回錯誤訊息。 |
| disable | 立即停用指定的元件。元件必須先經部署。若尚未部署元件，將傳回錯誤訊息。 |
| export | 將變數名稱標記成要自動匯出至後續指令的環境。除非取消設定所有後續指令或結束多重模式，否則所有後續指令都會使用指定的變數名稱值。 |

表 C–3 List 與 Status 指令 (續)

| | |
|---|---|
| get | 取得屬性的名稱與值。 |
| set | 設定一個或多個可配置屬性的值。 |
| list | 列出可配置的元素。在 Solaris 上，執行使用 * 做爲選項值或運算元的指令時，必須加上引號。 |
| unset | 移除針對 multimode 環境設定的一或多個變數。指定的變數與其關聯值將不再存在於環境中。 |

# 部署指令

部署指令可部署應用程式或取得用戶端 stub。

表 C–4 部署指令

| 指令 | 定義 |
|---|---|
| deploy | 部署企業應用程式、Web 應用程式、EJB 模組、連接器模組或應用程式用戶端模組。若某個元件已部署或已存在，且 --force 選項已設定爲 true，則會強制重新部署該元件。 |
| deploydir | 從開發目錄直接部署應用程式。部署目錄中必須有符合 Java EE 規格的適當目錄階層與部署描述元。 |
| get-client-stubs | 從伺服器到本機目錄，取得 AppClient 獨立模組或包含 AppClient 模組之應用程式的用戶端 stub JAR 檔案。執行此指令之前，必須先部署該應用程式或模組。 |

# 版本指令

版本指令會傳回版本字串，並顯示所有 asadmin 指令的清單，還允許您安裝授權檔案。

表 C–5 版本指令

| 指令 | 定義 |
|---|---|
| version | 顯示版本資訊。若此指令無法使用指定的使用者/密碼和主機/連接埠與管理伺服器通訊，它將會擷取本機版本並顯示警告訊息。 |
| help | 顯示所有 asadmin 公用程式指令的清單。指定特定指令可顯示關於該指令的用法資訊 |
| install-license | 防止未經授權即使用 Application Server。允許您安裝授權檔案。 |

表 C–5 版本指令 (續)

| 指令 | 定義 |
| --- | --- |
| shutdown | 以正常方式關閉管理伺服器與所有執行中的實例。若要重新啓動管理伺服器，必須以手動方式重新啓動。 |

# 訊息佇列管理指令

您可以使用訊息佇列管理指令來管理 JMS 目標。

表 C–6 訊息佇列指令

| 指令 | 定義 |
| --- | --- |
| create-jmsdest | 建立 JMS 實體目標。除了實體目標之外，您也可以使用 create-jms-resource 指令來建立具有指定實體目標之 Name 特性的 JMS 目標資源。 |
| delete-jmsdest | 移除指定的 JMS 目標。 |
| flush-jmsdest | 清除指定目標的 JMS 服務配置中，實體目標內的訊息。 |
| list-jmsdest | 列出 JMS 實體目標。 |
| jms-ping | 檢查 JMS 服務 (又稱爲 JMS 提供者) 是否已啓動並正在執行。當您啓動 Application Server 時，預設會啓動 JMS 服務。此外，它只會使用 ping 指令偵測 JMS 服務中的預設 JMS 主機。若使用 ping 指令偵測不到內建的 JMS 服務，則會顯示錯誤訊息。 |

# 資源管理指令

您可以使用資源指令來管理應用程式中使用的各種資源。

表 C–7 資源管理指令

| 指令 | 定義 |
| --- | --- |
| create-jdbc-connection-pool | 以指定的 JDBC 連線池名稱註冊新的 JDBC 連線池。 |
| delete-jdbc-connection-pool | 刪除 JDBC 連線池。運算元可識別要刪除的 JDBC 連線池。 |
| list-jdbc-connection-pools | 取得已建立的 JDBC 連線池。 |
| create-jdbc-resource | 建立新的 JDBC 資源。 |
| delete-jdbc-resource | 移除具有指定之 JNDI 名稱的 JDBC 資源。 |
| list-jdbc-resources | 顯示已建立之 JDBC 資源的清單。 |

表 C–7 資源管理指令 *(續)*

| 指令 | 定義 |
|---|---|
| create-jms-resource | 建立 Java 訊息服務 (JMS) 連線工廠資源或 JMS 目標資源。 |
| delete-jms-resource | 移除指定的 JMS 資源。 |
| list-jms-resources | 列出現有 JMS 資源 (目標與連線工廠資源)。 |
| create-jndi-resource | 註冊 JNDI 資源。 |
| delete-jndi-resource | 移除具有指定之 JNDI 名稱的 JNDI 資源。 |
| list-jndi-resources | 識別所有現有 JNDI 資源。 |
| list-jndi-entries | 瀏覽並查詢 JNDI 樹狀結構。 |
| create-javamail-resource | 建立 JavaMail 階段作業資源。 |
| delete-javamail-resource | 移除指定的 JavaMail 階段作業資源。 |
| list-javamail-resources | 列出現有 JavaMail 階段作業資源。 |
| create-persistence-resource | 註冊持續性資源。 |
| delete-persistence-resource | 移除持續性資源。當您刪除持續性資源時，此指令也會移除 JDBC 資源 (若該資源是使用 create-persistence-resource 指令所建立)。 |
| list-persistence-resources | 顯示所有持續性資源。 |
| create-custom-resource | 建立自訂資源。自訂資源指定實作 javax.naming.spi.ObjectFactory 介面的自訂全伺服器資源物件工廠。 |
| delete-custom-resource | 移除自訂資源。 |
| list-custom-resources | 列出自訂資源。 |
| create-connector-connection-pool | 增加具有指定之連線池名稱的新連接器連線池。 |
| delete-connector-connection-pool | 移除使用運算元 connector_connection_pool_name 指定的連接器連線池。 |
| list-connector-connection-pools | 列出已建立的連接器連線池。 |
| create-connector-resource | 註冊具有指定之 JNDI 名稱的連接器資源。 |
| delete-connector-resource | 移除具有指定之 JNDI 名稱的連接器資源。 |
| list-connector-resources | 取得所有連接器資源。 |
| create-admin-object | 建立具有指定之 JNDI 名稱的受管理物件。 |
| delete-admin-object | 移除具有指定之 JNDI 名稱的受管理物件。 |
| list-admin-objects | 列出所有受管理物件。 |

表 C–7 資源管理指令 (續)

| 指令 | 定義 |
|---|---|
| create-resource-adapter-config | 爲連接器模組建立配置資訊。 |
| delete-resource-adapter-config | 刪除爲連接器模組在 domain.xml 中建立的配置資訊。 |
| list-resource-adapter-configs | 列出 domain.xml 中的連接器模組配置資訊 |
| add-resources | 在指定的 XML 檔案中建立指定資源。*xml_file_path* 是包含要建立之資源的 XML 檔案的路徑。DOCTYPE 應該指定爲 resources.xml 檔案中的 *install_dir*/lib/dtds/sun-resources_1_2.dtd。 |
| ping-connection-pool | 測試 JDBC 連線池與連接器連線池中的連線池是否可用。例如，若爲稍後即將部署的應用程式建立新的 JDBC 連線池，部署該應用程式之前可使用此指令來測試該 JDBC 池。使用 ping 指令偵測連線池之前，您必須建立具有認證的連線池，並確定企業伺服器或資料庫已啓動。 |

# 配置指令

配置指令可讓您建構 IIOP 偵聽程式、生命週期模組、HTTP 和 IIOP 偵聽程式、效能評測器及其他子系統。

本小節包含下列主題：


- 第 231 頁的 「HTTP 和 IIOP 偵聽程式指令」
- 第 232 頁的 「生命週期與稽核模組指令」
- 第 232 頁的 「效能評測器和 SSL 指令」
- 第 233 頁的 「JVM 選項和虛擬伺服器指令」
- 第 233 頁的 「執行緒池和認證範圍指令」
- 第 234 頁的 「作業事件和計時器指令」


## HTTP 和 IIOP 偵聽程式指令

HTTP 與 IIOP 偵聽程式指令可協助您管理偵聽程式。只有遠端模式支援這些指令。

表 C–8 IIOP 偵聽程式指令

| 指令 | 定義 |
|---|---|
| create-http-listener | 增加新的 HTTP 偵聽程式通訊端。 |
| delete-http-listener | 移除指定的 HTTP 偵聽程式。 |
| list-http-listeners | 列出現有 HTTP 偵聽程式。 |

表 C–8 IIOP 偵聽程式指令 (續)

| 指令 | 定義 |
| --- | --- |
| create-iiop-listener | 建立 IIOP 偵聽程式。 |
| delete-iiop-listener | 移除指定的 IIOP 偵聽程式。 |
| list-iiop-listeners | 列出現有 IIOP 偵聽程式。 |

# 生命週期與稽核模組指令

生命週期與稽核模組指令可協助您控制生命週期模組，與實作稽核功能的選擇性外掛程式模組。只有遠端模式支援這些指令。

表 C–9 生命週期模組指令

| 指令 | 定義 |
| --- | --- |
| create-lifecycle-module | 建立生命週期模組。生命週期模組提供的方式，能讓您在 Application Server 環境中執行短期或長期 Java 作業。 |
| delete-lifecycle-module | 移除指定的生命週期模組。 |
| list-lifecycle-modules | 列出現有生命週期模組。 |
| create-audit-module | 爲實作稽核功能的外掛程式模組增加指定的稽核模組。 |
| delete-audit-module | 移除指定的稽核模組。 |
| list-audit-modules | 列出所有稽核模組。 |

# 效能評測器和 SSL 指令

效能評測器和 SSL 指令可讓您管理效能評測器和 SSL 用戶端配置。只有遠端模式支援這些指令。

表 C–10 效能評測器和 SSL 指令

| 指令 | 定義 |
| --- | --- |
| create-profiler | 建立效能評測器元素。伺服器實例會依據 Java 配置中的效能評測器元素，連接至特定的效能評測器。 變更效能評測器之後必須重新啓動伺服器。 |
| delete-profiler | 刪除指定的效能評測器元素。伺服器實例會依據 Java 配置中的效能評測器元素連接至特定的效能評測器。變更效能評測器之後必須重新啓動伺服器。 |

表 C–10 效能評測器和SSL指令 *(續)*

| 指令 | 定義 |
| --- | --- |
| `create-ssl` | 在選取的HTTP偵聽程式、IIOP偵聽程式或IIOP服務中建立並配置SSL元素，以保護偵聽程式/服務上的通訊安全。 |
| `delete-ssl` | 刪除選取的HTTP偵聽程式、IIOP偵聽程式或IIOP服務中的SSL元素。 |

# JVM選項和虛擬伺服器指令

您可以使用JVM選項和虛擬伺服器指令來控制這些元素。只有遠端模式支援這些指令。

表 C–11 JVM選項和虛擬伺服器指令

| 指令 | 定義 |
| --- | --- |
| `create-jvm-option` | 在Java配置或`domain.xml`檔案的效能評測器元素中建立JVM選項。若爲效能評測器建立了JVM選項，則會使用這些選項來記錄執行特定效能評測器所需的設定。您必須重新啓動伺服器，新建的JVM選項才會生效。 |
| `delete-jvm-option` | 移除Java配置或`domain.xml`檔案效能評測器元素中的JVM選項。 |
| `create-virtual-server` | 建立指定的虛擬伺服器。Application Server中的虛擬功能可讓偵聽多個主機位址的單一HTTP伺服器程序服務多個URL網域。若兩部虛擬伺服器提供相同的應用程式，它們仍會共用相同的實體資源池。 |
| `delete-virtual-server` | 移除具有指定之虛擬伺服器ID的虛擬伺服器。 |

# 執行緒池和認證範圍指令

您可以使用執行緒池和認證範圍指令來控制這些元素。只有遠端模式支援這些指令。

表 C–12 執行緒池和認證範圍指令

| 指令 | 定義 |
| --- | --- |
| `create-threadpool` | 建立具有指定名稱的執行緒池。您可以指定池中執行緒的最大數目與最小數目、工作佇列數目，以及執行緒的閒置逾時。已建立的執行緒池可用於服務IIOP請求，並由資源配接卡用來服務工作管理請求。已建立的執行緒池可用於多個資源配接卡。 |
| `delete-threadpool` | 移除具有指定之ID的執行緒池。 |
| `list-threadpools` | 列出所有執行緒池。 |

表 C–12 執行緒池和認證範圍指令 *(續)*

| 指令 | 定義 |
| --- | --- |
| create-auth-realm | 增加指定的認證範圍。 |
| delete-auth-realm | 移除指定的認證範圍。 |

# 作業事件和計時器指令

您可以使用作業事件和計時器指令來控制作業事件和計時器子系統，以及暫停任何執行中的作業事件。只有遠端模式支援這些指令。

表 C–13 作業事件指令

| 指令 | 定義 |
| --- | --- |
| freeze-transaction | 在所有執行中作業事件都已暫停的情況下，固定作業事件子系統。回復任何執行中的作業事件之前，請先呼叫此指令。針對任何已固定的作業事件子系統呼叫此指令沒有任何效用。 |
| unfreeze-transaction | 繼續所有已暫停的執行中作業事件。在已固定的作業事件上呼叫此指令。 |
| recover-transactions | 手動回復擱置的作業事件。 |
| rollback-transaction | 回復指定的作業事件。 |
| unpublish-from-registry | |
| list-timers | 列出特定伺服器實例所擁有的計時器 |

# 登錄指令

登錄指令讓您發佈或取消發佈 Web 服務工件。

表 C–14 作業事件指令

| 指令 | 定義 |
| --- | --- |
| publish-to-registry | 將 Web 服務工件發佈至登錄。 |
| unpublish-from-registry | 從登錄取消發佈 Web 服務工件。 |
| list-registry-locations | |

# 使用者管理指令

您可以使用這些使用者指令來管理檔案範圍認證支援的使用者。只有遠端模式支援這些指令。

**表C–15** 使用者管理指令

| 指令 | 定義 |
|---|---|
| `create-file-user` | 在金鑰檔案中建立具有指定使用者名稱、密碼和群組的項目。您可以建立多個群組，並使用冒號 (:) 加以分隔。 |
| `delete-file-user` | 在金鑰檔案中刪除具有指定使用者名稱的項目。 |
| `update-file-user` | 使用指定的 `user_name`、`user_password` 和群組，更新金鑰檔案中的現有項目。您可以輸入多個群組，並使用冒號 (:) 加以分隔。 |
| `list-file-users` | 建立檔案範圍認證支援的檔案使用者清單。 |
| `list-file-groups` | 管理檔案範圍認證支援的檔案使用者與群組。此指令可列出檔案使用者中的可用群組。 |

# 規則和監視指令

您可以使用規則和監視指令來管理規則和監視伺服器。只有遠端模式支援這些指令。

**表C–16** 規則和監視指令

| 指令 | 定義 |
|---|---|
| `create-management-rule` | 建立新的管理規則，以透過智慧型的方式自我管理 Application Server 安裝與已部署的應用程式。 |
| `delete-management-rule` | 移除指定的管理規則。 |
| `create-transformation-rule` | 建立可套用至 Web 服務作業的 XSLT 變換規則。規則可套用到請求或回應。 |
| `delete-transformation-rule` | 刪除指定之 Web 服務的 XSLT 變換規則。 |
| `start-callflow-monitoring` | 從 Web 容器、EJB 容器與 JDBC 收集資料並建立資料關聯性，以提供完整的請求呼叫流程/路徑。只有當 `callflow-monitoring` 設定爲 ON 時才會收集資料。 |
| `stop-callflow-monitoring` | 停止收集請求呼叫流程資訊。 |

# 資料庫指令

您可以使用資料庫指令來啓動和停止 Java DB 資料庫 (基於 Apache Derby)。只有本機模式支援這些指令。

**表 C–17** 資料庫指令

| 指令 | 定義 |
|---|---|
| start-database | 啓動 Application Server 隨附的 Java DB 伺服器。此指令只適用於已部署至 Application Server 的應用程式。 |
| stop-database | 停止 Java DB 伺服器的程序。Java DB 伺服器隨附於 Application Server。 |

# 診斷與記錄指令

診斷與記錄指令可協助您疑難排解 Application Server 的問題。只有遠端模式支援這些指令。

**表 C–18** 診斷與記錄指令

| 指令 | 定義 |
|---|---|
| `generate-diagnostic-report` | 產生包含 Application Server 安裝詳細資訊 (例如，Application Server 實例的配置詳細資訊、記錄詳細資訊或程序專用資訊) 之指標或瀏覽連結的 HTML 報告。 |
| `display-error-statistics` | 顯示自上次伺服器重新啓動後，`server.log` 中記錄的嚴重性與警告的摘要清單。 |
| `display-error-distribution` | 顯示實例 `server.log` 中整個模組的錯誤分佈狀況。 |
| `display-log-records` | 根據指定時間戳記顯示指定模組的所有錯誤訊息。 |

# Web 服務指令

您可以使用 Web 服務指令來監視已部署的 Web 服務並管理變換規則。

**表 C–19** Web 服務指令

| 指令 | 定義 |
|---|---|
| `configure-webservice-management` | 配置已部署之 Web 服務的監視或 `maxhistory` 屬性。 |
| `create-transformation-rule` | 建立可套用至 Web 服務作業的 XSLT 變換規則。規則可套用到請求或回應。 |

表 C–19 Web服務指令 (續)

| 指令 | 定義 |
| --- | --- |
| delete-transformation-rule | 刪除指定之 Web 服務的 XSLT 變換規則。 |
| list-transformation-rules性 | 列出指定之 Web 服務的所有變換規則 (按照規則的套用順序)。 |
| publish-to-registry | 將 Web 服務工件發佈至登錄。 |
| unpublish-from-registry | 從登錄取消發佈 Web 服務工件。 |
| list-registry-locations | 顯示已配置之 Web 服務登錄存取點的清單。 |

# 安全性服務指令

您可以使用這些安全指令來控制連接器連線池的安全對映。只有遠端模式支援這些指令。

表 C–20 安全指令

| 指令 | 定義 |
| --- | --- |
| create-connector-security-map | 建立指定之連接器連線池的安全對映。若該安全對映不存在，則會建立新的安全對映。此外，在基於容器管理式作業事件的方案中，您可以使用此指令將應用程式的呼叫者身份 (主體或使用者群組) 對映到適當的企業資訊系統 (EIS) 主體。一個或多個指定的安全對映可能會與連接器連線池關聯。連接器安全對映配置支援使用萬用字元星號 (*) 來表示所有使用者或所有使用者群組。爲順利執行此指令，您必須先建立連接器連線池。EIS 是指控管組織資料的任何系統。它可以是主機、訊息傳送系統、資料庫系統或應用程式。 |
| delete-connector-security-map | 刪除指定之連接器連線池的安全對映。 |
| update-connector-security-map | 修改指定之連接器連線池的安全對映。 |
| list-connector-security-map | 列出屬於指定之連接器連線池的安全對映。 |
| create-message-security-provider | 可供管理員爲指定的訊息層 (domain.xml 的 message-security-config 元素，此檔案指定 Application Server 的參數與特性) 建立 provider-config 子元素。 |
| delete-message-security-provide | 可供管理員爲指定的訊息層 (domain.xml 的 message-security-config 元素，此檔案指定 Application Server 的參數與特性) 刪除 provider-config 子元素。 |
| list-message-security-providers | 可供管理員列出指定之訊息層 (domain.xml 的 message-security-config 元素) 的所有安全訊息提供者 (provider-config 子元素)。 |

# 密碼指令

您可以使用密碼指令來管理密碼以確保 Application Server 的安全性。

**表 C–21** 密碼指令

| 指令 | 定義 |
|---|---|
| `create-password-alias` | 建立密碼的別名並將其儲存在 `domain.xml` 中。別名是格式爲 `${ALIAS=password-alias-password}` 的記號。別名名稱所對應的密碼是以已加密的格式儲存。此指令支援安全的互動式方式 (提示使用者輸入所有資訊)，以及更適合用於程序檔的方式 (密碼會傳遞到指令行)。 |
| `delete-password-alias` | 刪除密碼別名。 |
| `update-password-alias` | 更新指定之目標中的密碼別名 ID。 |
| `list-password-aliases` | 列出所有密碼別名。 |
| `change-admin-password` | 此遠端指令可修改管理密碼。此指令會以互動式方式執行，也就是會提示使用者輸入舊的管理密碼和新的管理密碼 (包含確認)。 |
| `change-master-password` | 此本機指令可修改主密碼。此指令會以互動式方式執行，也就是會提示使用者輸入舊的主密碼與新的主密碼。除非伺服器已停止，否則此指令將無法運作。 |

# 驗證指令

XML 檢驗器指令可驗證 `domain.xml` 檔案的內容。

**表 C–22** 驗證指令

| 指令 | 定義 |
|---|---|
| `verify-domain-xml` | 驗證 `domain.xml` 檔案的內容。 |

# 自訂 MBean 指令

您可以使用 MBean 指令來管理與註冊自訂 MBean。只有遠端模式支援這些指令。

**表 C–23** 自訂 MBean 指令

| 指令 | 定義 |
|---|---|
| `create-mbean` | 建立並註冊自訂 MBean。若目標 `MBeanServer` 並未執行，則不會註冊 MBean。 |

表 C–23 自訂 MBean 指令 (續)

| 指令 | 定義 |
| --- | --- |
| delete-mbean | 刪除自訂 MBean。確定目標 MBeanServer 正在執行。 |
| list-mbeans | 列出指定之目標的自訂 MBean。 |

# 服務指令

您可以使用服務指令來配置 Domain Administration Server (DAS) 的啓動。

表 C–24 服務指令

| 指令 | 定義 |
| --- | --- |
| create-service | 在自動啓動環境中配置 DAS 的啓動作業。在 Solaris 10 上，此指令會使用服務管理功能 (SMF)。這是本機指令，而且必須以具有超級使用者權限的作業系統層級使用者身份執行。只有 Solaris 10 才提供此指令。建立該服務之後，使用者必須啓動、啓用、停用、刪除或停止該服務。DAS 必須儲存在超級使用者可存取的資料夾中。您不能將該配置儲存在網路檔案系統上。該服務建立之後，即可由擁有 DAS 配置所在之資料夾的作業系統層級使用者控制。若要執行此指令，您必須具有 solaris.smf.* 授權。 |

# 特性指令

共用伺服器實例通常需要覆寫其參照配置中定義的屬性。您可透過名稱相同的系統特性，覆寫伺服器實例中的任何配置屬性。使用系統特性指令可管理這些共用的伺服器實例。

表 C–25 特性指令

| 指令 | 定義 |
| --- | --- |
| create-system-property | 針對網域、配置或伺服器實例，一次建立一個系統特性。 |
| delete-system-property | 移除網域、配置或伺服器實例的一個系統特性。 |
| list-system-properties | 顯示網域、配置或伺服器實例的系統特性。 |

# 索引

## S



## T



## W