



Sun Java System Application Server Enterprise Edition 8.2 管 理指南



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

文件號碼：820-0848
2007 年 3 月

Copyright 2007 Sun Microsystems, Inc. 4150 Network Circle, Santa Clara, CA 95054 U.S.A. 版權所有。

Sun Microsystems, Inc. 對本文件所述產品所採用的技術擁有相關智慧財產權。特別是 (但不僅限於)，這些智慧財產權可能包含一項或多項美國專利，或美國及其他國家/地區的待批專利。

美國政府權利 — 商業軟體。政府使用者均應遵守 Sun Microsystems, Inc. 的標準授權合約和 FAR 及其增補文件中的適用條款。

本發行物可能包含由協力廠商開發的材料。

本產品中的某些部分可能源自加州大學授權的 Berkeley BSD 系統的開發成果。UNIX 是在美國及其他國家/地區的註冊商標，已獲得 X/Open Company, Ltd. 專屬授權。

Sun、Sun Microsystems、Sun 標誌、Solaris 標誌、Java 咖啡杯標誌、docs.sun.com、Java 與 Solaris 是 Sun Microsystems, Inc. 在美國及其他國家/地區的商標或註冊商標。所有 SPARC 商標都是 SPARC International, Inc. 在美國及其他國家/地區的商標或註冊商標，經授權後使用。凡具有 SPARC 商標的產品都是採用 Sun Microsystems, Inc. 所開發的架構。

OPEN LOOK 與 SunTM Graphical User Interface (Sun 圖形化使用者介面) 都是由 Sun Microsystems, Inc. 為其使用者與授權者所開發的技術。Sun 感謝 Xerox 公司在研究和開發視覺化或圖形化使用者介面之概念上，為電腦工業所做的開拓性貢獻。Sun 已向 Xerox 公司取得 Xerox 圖形化使用者介面之非獨占性授權，該授權亦適用於使用 OPEN LOOK GUI 並遵守 Sun 書面授權合約的 Sun 公司授權者。

本出版物所涵蓋的產品和包含的資訊受到美國出口控制法規的控制，並可能受到其他國家/地區進出口法規的管轄。嚴禁核武、導彈、生化武器或海上核武等最終用途或一般使用者直接或間接使用本產品。嚴禁向被美國禁運的國家/地區或美國出口除外清單 (包括但不僅限於被拒人清單和特別指定的國家/地區清單) 上標識的實體出口或再出口本產品。

本文件以其「原狀」提供，對任何明示或暗示的條件、陳述或擔保，包括對適銷性、特殊用途的適用性或非侵權性的暗示保證，均不承擔任何責任，除非此免責聲明的適用範圍在法律上無效。

目錄

前言	19
1 入門	25
關於 Sun Java System Application Server	25
什麼是 Application Server ?	26
Application Server 架構	26
管理工具	28
Application Server 指令和概念	30
網域	30
Domain Administrative Server (DAS)	31
叢集	31
節點代理程式	31
伺服器實例	32
應用程式伺服器指令	34
Application Server 配置	41
變更 Application Server 配置	41
Application Server 中的連接埠	42
變更 J2SE 軟體	42
2 部署應用程式	43
部署生命週期	43
自動部署	44
部署未封裝的應用程式	45
使用部署規劃	45
使用 deploytool 公用程式	46
J2EE 歸檔檔案的類型	46
命名慣例	47

3 JDBC 資源	49
建立 JDBC 資源	49
建立 JDBC 連線池	50
JDBC 資源和連線池如何協同工作	52
設定資料庫存取	53
持續性管理程式資源	53
 4 配置 Java 訊息服務資源	55
關於 JMS 資源	55
Application Server 中的 JMS 提供者	55
JMS 資源	56
JMS 資源和連接器資源的關係	57
JMS 連線工廠	57
JMS 目標資源	57
JMS 實體目標	58
JMS 提供者	58
配置 JMS 提供者的一般特性	58
外來的 JMS 提供者	59
配置 JMS 的通用資源配接卡	59
資源配接卡特性	61
ManagedConnectionFactory 特性	63
受管理物件資源特性	64
啓動規格特性	64
 5 配置 JavaMail 資源	67
建立 JavaMail 階段作業	67
 6 JNDI 資源	69
J2EE 命名服務	69
命名參考與連結資訊	70
使用自訂資源	71
使用外部 JNDI 儲存庫和資源	71

7	連接器資源	73
	連接器連線池	73
	連接器資源	75
	受管理物件資源	75
8	J2EE 容器	77
	J2EE 容器的類型	77
	Web 容器	77
	EJB 容器	77
	配置 J2EE 容器	78
	配置一般 Web 容器設定	78
	配置 Web 容器階段作業	78
	配置虛擬伺服器設定	80
	配置一般 EJB 設定	80
	配置訊息導引 Bean 設定	81
	配置 EJB 計時器服務設定	81
9	配置安全性	83
	瞭解應用程式和系統安全性	83
	管理安全性的工具	84
	管理密碼安全性	85
	對 domain.xml 檔案中的密碼進行加密	85
	保護具有編碼密碼的檔案	86
	變更主密碼	86
	使用主密碼和金鑰庫	86
	變更管理員密碼	87
	關於認證與授權	87
	認證實體	87
	對使用者進行授權	88
	指定 JACC 提供者	88
	稽核認證與授權決策	89
	配置訊息安全性	89
	瞭解使用者、群組、角色和範圍	89
	使用者	90
	群組	90

角色	91
範圍	91
憑證和 SSL 簡介	92
關於數位憑證	92
關於安全套接字層	93
關於防火牆	94
使用 Administration Console 管理安全性	95
伺服器安全性設定	95
範圍和 file 範圍使用者	95
JACC 提供者	95
稽核模組	96
訊息安全性	96
HTTP 和 IIOP 偵聽程式安全性	96
管理服務安全性	96
安全性對映	97
使用憑證和 SSL	97
關於憑證檔案	97
使用 Java 安全套接字延伸 (JSSE) 工具	98
使用網路安全服務 (NSS) 工具	101
將 Application Server 與硬體加密加速器搭配使用	104
詳細資訊	110
10 配置訊息安全性	111
訊息安全性概況	111
瞭解 Application Server 中的訊息安全性	112
指定訊息安全性職責	112
關於安全性記號和安全性機制	113
訊息安全性術語字彙表	114
確保 Web 服務的安全	115
配置應用程式特定的 Web 服務安全性	116
確保範例應用程式的安全	116
配置 Application Server 以實現訊息安全性	116
請求策略配置和回應策略配置的動作	116
配置其他安全性功能	117
配置 JCE 提供者	118

訊息安全性設定	119
啟用訊息安全性的提供者	120
配置訊息安全性提供者	120
建立訊息安全性提供者	121
啟用應用程式用戶端的訊息安全性	121
設定應用程式用戶端配置的請求策略和回應策略	121
詳細資訊	122
11 作業事件	125
何為作業事件？	125
J2EE 技術中的作業事件	126
恢復作業事件	126
作業事件逾時值	127
作業事件記錄	127
關鍵點間隔	127
12 配置 HTTP 服務	129
虛擬伺服器	129
HTTP 偵聽程式	130
13 配置物件請求代理程式	133
CORBA	133
什麼是 ORB？	133
IIOP 偵聽程式	134
使用 ORB	134
協力廠商的 ORB	134
14 執行緒池	135
配置執行緒池	135
15 配置記錄	137
記錄檔的記錄	137
設定自訂記錄層級	138
記錄程式名稱空間階層結構	139

16 監視元件和服務	141
監視概述	141
關於可監視物件的樹狀結構	141
受監視的元件和服務的統計資訊	145
EJB 容器統計資訊	145
Web 容器統計資訊	149
HTTP 服務統計資訊	150
JDBC 連線池統計資訊	151
JMS/連接器服務統計資訊	152
ORB 中連線管理程式的統計資訊	153
執行緒池統計資訊	153
作業事件服務統計資訊	154
Java 虛擬機器 (JVM) 統計資訊	154
生產 Web 容器 (PWC) 統計資訊	158
啓用與停用監視	163
檢視監視資料	164
瞭解和指定帶點名稱	165
list 指令的範例	166
get 指令的範例	167
使用 PetStore 範例	169
list 和 get 指令在所有層級上的預期輸出	172
使用 JConsole	178
保護 JConsole 與 Application Server 間連線的安全	178
將 JConsole 連線至 Application Server 的必要條件	179
將 JConsole 連線至 Application Server	179
以安全的方式將 JConsole 連線至 Application Server	180
17 Java 虛擬機器和進階設定	183
調校 JVM 設定	183
配置進階選項	184
18 domain.xml 的帶點名稱屬性	185
頂層元素	185
不能別名化的元素	187
asadmin 指令	187

19 asadmin 公用程式	189
asadmin 指令用法	190
多重模式與互動式模式	190
本機指令	191
遠端指令	191
密碼檔案	192
Multimode 指令	193
List、Get 與 Set 指令	193
伺服器生命週期指令	194
List 與 Status 指令	195
部署指令	196
訊息佇列管理指令	196
資源管理指令	197
Application Server 配置指令	198
一般配置指令	199
HTTP、IIOP 與 SSL 偵聽程式指令	199
生命週期與稽核模組指令	200
效能評測器和 JVM 選項指令	200
虛擬伺服器指令	201
執行緒池指令	201
作業事件和計時器指令	201
使用者管理指令	202
監視資料指令	202
規則指令	203
資料庫指令	203
診斷與記錄指令	204
Web 服務指令	204
安全性服務指令	205
密碼指令	205
驗證 domain.xml 指令	206
自訂 MBean 指令	206
其他指令	207
 索引	 209

圖清單

圖 1-1	Application Server 架構	27
圖 1-2	Application Server 實例	33
圖 9-1	角色對映	90

表清單

表 1-1	使用連接埠的 Application Server 偵聽程式	42
表 6-1	JNDI 查找及其關聯的參考	70
表 9-1	Application Server 認證方法	88
表 10-1	訊息保護策略與 WS-Security SOAP 訊息安全性作業對映	117
表 15-1	Application Server 記錄程式名稱空間	139
表 16-1	EJB 統計資訊	145
表 16-2	EJB 方法統計資訊	146
表 16-3	EJB 階段作業儲存統計	146
表 16-4	EJB 池統計資訊	147
表 16-5	EJB 快取統計	148
表 16-6	計時器統計資訊	148
表 16-7	Web 容器 (Servlet) 統計資訊	149
表 16-8	Web 容器 (Web 模組) 統計資訊	149
表 16-9	HTTP 服務統計資訊 (僅適用於 Platform Edition)	150
表 16-10	JDBC 連線池統計	151
表 16-11	連接器連線池統計資訊	152
表 16-12	連接器工作管理統計資訊	153
表 16-13	ORB 中連線管理程式的統計資訊	153
表 16-14	執行緒池統計資訊	153
表 16-15	作業事件服務統計資訊	154
表 16-16	JVM 統計資訊	154
表 16-17	J2SE 5.0 的 JVM 統計資訊 - 類別載入	155
表 16-18	J2SE 5.0 的 JVM 統計資訊 - 編譯	155
表 16-19	J2SE 5.0 的 JVM 統計資訊 - 資源回收	155
表 16-20	J2SE 5.0 的 JVM 統計資訊 - 記憶體	155
表 16-21	J2SE 5.0 的 JVM 統計資訊 - 作業系統	156
表 16-22	J2SE 5.0 的 JVM 統計資訊 - 執行階段	156
表 16-23	J2SE 5.0 的 JVM 統計 — 執行緒資訊	157

表 16-24	J2SE 5.0 的 JVM 統計資訊 - 執行緒	158
表 16-25	PWC 虛擬伺服器統計資訊 (僅限於 EE)	158
表 16-26	PWC 請求統計 (僅限於 EE)	159
表 16-27	PWC 檔案快取統計資訊 (僅限於 EE)	160
表 16-28	PWC 持續作用統計 (僅限於 EE)	160
表 16-29	PWC DNS 統計 (僅限於 EE)	161
表 16-30	PWC 執行緒池統計資訊 (僅限於 EE)	161
表 16-31	PWC 連線佇列統計資訊 (僅限於 EE)	162
表 16-32	PWC HTTP 服務統計資訊 (僅限於 EE)	163
表 16-33	頂層	172
表 16-34	應用程式層級	172
表 16-35	應用程式 - 企業應用程式和獨立模組	173
表 16-36	HTTP 服務層級	176
表 16-37	執行緒池層級	176
表 16-38	資源層級	176
表 16-39	作業事件服務層級	177
表 16-40	ORB 層級	177
表 16-41	JVM 層級	178
表 19-1	遠端指令必需的選項	191
表 19-2	伺服器生命週期指令	194
表 19-3	List 與 Status 指令	195
表 19-4	部署指令	196
表 19-5	訊息佇列指令	196
表 19-6	資源管理指令	197
表 19-7	一般配置指令	199
表 19-8	IIOP 偵聽程式指令	199
表 19-9	生命週期模組指令	200
表 19-10	效能評測器和 JVM 選項指令	200
表 19-11	虛擬伺服器指令	201
表 19-12	執行緒池指令	201
表 19-13	作業事件指令	202
表 19-14	使用者管理指令	202
表 19-15	監視資料指令	203
表 19-16	規則指令	203
表 19-17	資料庫指令	203
表 19-18	診斷與記錄指令	204

表 19-19	Web 服務指令	204
表 19-20	安全指令	205
表 19-21	密碼指令	206
表 19-22	驗證 domain.xml 指令	206
表 19-23	自訂 MBean 指令	206
表 19-24	其他指令	207

範例清單

範例 16-1	應用程式節點樹狀結構	142
範例 16-2	HTTP 服務示意圖 (Platform Edition 版)	143
範例 16-3	HTTP 服務示意圖 (Enterprise Edition 版)	143
範例 16-4	資源示意圖	144
範例 16-5	連接器服務示意圖	144
範例 16-6	JMS 服務示意圖	144
範例 16-7	ORB 示意圖	144
範例 16-8	執行緒池示意圖	145
範例 19-1	語法範例	190
範例 19-2	help 指令範例	190
範例 19-3	密碼檔案內容	192

前言

「Sun Java System Application Server Enterprise Edition 8.2 管理指南」提供 Application Server 的系統管理資訊，包括配置、監視、安全性、資源管理與 Web 服務管理等。

本前言包含整個 Sun Java™ System 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。獨立 Application Server 文件的統一資源定址器 (URL) 為 <http://docs.sun.com/app/docs/coll/1310.4>。Sun Java Enterprise System (Java ES) Application Server 文件的 URL 為 <http://docs.sun.com/app/docs/coll/1310.3> 和 <http://docs.sun.com/app/docs/coll/1416.2>。如需 Application Server 的簡介，請按照表格中列出的順序參閱這些書籍。

表 P-1 Application Server 文件集中的書籍

書籍標題	說明
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™) 與資料庫驅動程式的完整表格式摘要。
Quick Start Guide	如何開始使用 Application Server 產品。
Installation Guide	安裝軟體及其元件。
Deployment Planning Guide	評估系統需求和企業狀況，確保以最適合您的站點的方式部署 Application Server。此外還說明了部署伺服器時應該注意的常見問題及注意事項。
Developer's Guide	建立和實作要在 Application Server 上執行的 Java 2 Platform, Enterprise Edition (J2EE™ 平台) 應用程式，這些應用程式遵循 J2EE 元件和 API 的開放式 Java 標準模型。其中包括和開發者工具、安全性、除錯、部署和建立生命週期模組的相關資訊。
J2EE 1.4 Tutorial	使用 J2EE 1.4 平台技術與 API 來開發 J2EE 應用程式。
管理指南	從 Administration Console 配置、管理和部署 Application Server 子系統和元件。
High Availability Administration Guide	高可用性資料庫安裝後的配置和管理說明。

表 P-1 Application Server 文件集中的書籍 (續)

書籍標題	說明
Administration Reference	編輯 Application Server 配置檔案 domain.xml。
Upgrade and Migration Guide	將應用程式遷移到新的 Application Server 程式設計模型，特別是從 Application Server 6.x 和 7 進行遷移。該指南還說明可導致與產品規格不相容的相鄰產品發行版本和配置選項之間的差異。
Performance Tuning Guide	調校 Application Server 以提昇效能。
Troubleshooting Guide	解決 Application Server 問題。
Error Message Reference	解決 Application Server 錯誤訊息。
Reference Manual	Application Server 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 可單獨購買或做為 Java ES 的元件購買，Java ES 是一種支援分散在網路或網際網路環境中的企業應用程式的軟體基礎架構。若是以 Java ES 的元件方式購買 Application Server，您應熟悉位於 <http://docs.sun.com/coll/1286.2> 和 <http://docs.sun.com/coll/1412.2> 的系統文件。有關 Java ES 與其元件的所有文件之 URL 為 <http://docs.sun.com/prod/entsys.5> 和 http://docs.sun.com/prod/entsys.5?l=zh_TW。

如需其他 Sun Java System 伺服器文件，請至：

- Message Queue 文件
- Directory Server 文件
- Web 伺服器 文件

此外，以下資源可能會有用：

- J2EE 1.4 規格 (<http://java.sun.com/j2ee/1.4/docs/index.html>)
- J2EE 1.4 指導文件 (<http://java.sun.com/j2ee/1.4/docs/tutorial/doc/index.html>)
- J2EE 藍圖 (<http://java.sun.com/reference/blueprints/index.html>)

預設路徑和檔案名稱

下表描述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
<i>install-dir</i>	表示 Application Server 的基底安裝目錄。	Solaris™ 平台上的 Sun Java Enterprise System (Java ES) 安裝： /opt/SUNWappserver/appserver Linux 平台上的 Java ES 安裝： /opt/sun/appserver/ 其他 Solaris 和 Linux 的安裝 (非超級使用者)： user's home directory/SUNWappserver 其他 Solaris 和 Linux 的安裝 (超級使用者)： /opt/SUNWappserver Windows，所有安裝： SystemDrive:\Sun\AppServer
<i>domain-root-dir</i>	表示包含所有網域的目錄。	Solaris 平台上的 Java ES 安裝： /var/opt/SUNWappserver/domains/ Linux 平台上的 Java ES 安裝： /var/opt/sun/appserver/domains/ 所有其他安裝： install-dir/domains/
<i>domain-dir</i>	表示網域的目錄。 在配置檔案中，您可能會看到 <i>domain-dir</i> 顯示如下： \${com.sun.aas.instanceRoot}	<i>domain-root-dir/domain-dir</i>
<i>instance-dir</i>	表示伺服器實例的目錄。	<i>domain-dir/instance-dir</i>

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	意義	範例
AaBbCc123	指令、檔案及目錄的名稱；螢幕畫面輸出。	請編輯您的 .login 檔案。 請使用 ls -a 列出所有檔案。 machine_name% you have mail.
AaBbCc123	您所鍵入的內容 (與螢幕畫面輸出相區別)。	machine_name% su Password:
AaBbCc123	將用實際的名稱或數值取代的預留位置。	移除檔案的指令為 rm 檔案名稱。
AaBbCc123	新術語，要強調的詞。	快取記憶體是儲存在本機的副本。 請 不要 儲存此檔案。 備註： 某些重點項目在線上以粗體顯示。
「AaBbCc123」 用於書名及章節名稱。		請參閱「使用者指南」中的第 6 章。

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。
{ }	包含為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。
-	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，然後鬆開 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、支援和培訓

Sun 網站提供和下列其他資源有關的資訊：

- 文件 (<http://www.sun.com/documentation/>)
- 支援 (<http://www.sun.com/support/>)
- 培訓 (<http://www.sun.com/training/>)

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)]。在線上表單中，請提供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，本書的文件號碼為 820-0848。

◆ ◆ ◆ 第 1 章

入門

本章針對 Sun Java System Application Server 的管理加以說明。Application Server 管理包含多項作業，如部署應用程式、建立並配置網域、伺服器實例和資源、控制 (啟動和停止) 網域和伺服器實例、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：

- 第 25 頁的「關於 Sun Java System Application Server」
- 第 30 頁的「Application Server 指令和概念」
- 第 41 頁的「Application Server 配置」

關於 Sun Java System Application Server

Sun Java System Application Server 提供 Java 2 Platform, Enterprise Edition (J2EE 平台) 1.4 相容平台，以開發伺服器端的 Java 應用程式，並提供伺服器端的 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。

Application Server 提供以下版本：

- Platform 版是免費的，主要用於軟體開發和部門層級的生產環境。
- Enterprise 版是針對關鍵業務的服務和大規模的生產環境而設計的。此版本透過負載平衡程式外掛程式和叢集管理，支援水平延展和服務持續。Enterprise 版同時透過高可用性資料庫 (HADB) 支援可靠的階段作業狀態管理。

本小節包含下列主題：

- 第 26 頁的「什麼是 Application Server？」
- 第 26 頁的「Application Server 架構」
- 第 28 頁的「管理工具」

什麼是 Application Server ？

Application Server 平台支援的服務範圍從 Web 發佈一直到企業範圍的作業事件處理，同時可讓開發者建立基於 JavaServer Pages (JSP)、Java Servlet 以及企業 Java Bean (EJB) 技術的應用程式。

Application Server Platform Edition 是一款用於開發、生產部署和再分發的**免費**軟體。如需有關再分發的更多資訊，請至

http://www.sun.com/software/products/appsrvr/appsrvr_oem.xml。

Application Server 提供了進階叢集和容錯移轉技術。Application Server 基礎架構支援多種類型之分散式應用程式的部署，同時也是建立基於服務導向架構 (SOA) 之應用程式的理想基礎。SOA 是一種設計方法，旨在最大化應用程式服務的重複使用。這些功能讓您可以執行可延伸的且具有高可用性的 J2EE 應用程式。

- **延伸性** - 延伸性是透過叢集實現的。叢集是一組應用程式伺服器實例，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 實例均部署有相同的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** - 可用性是指容錯移轉的能力。當一部伺服器實例發生故障時，叢集中另一部伺服器實例就會接管發生故障實例的階段作業，並以一致的方式繼續為用戶端提供服務。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 的高階架構。

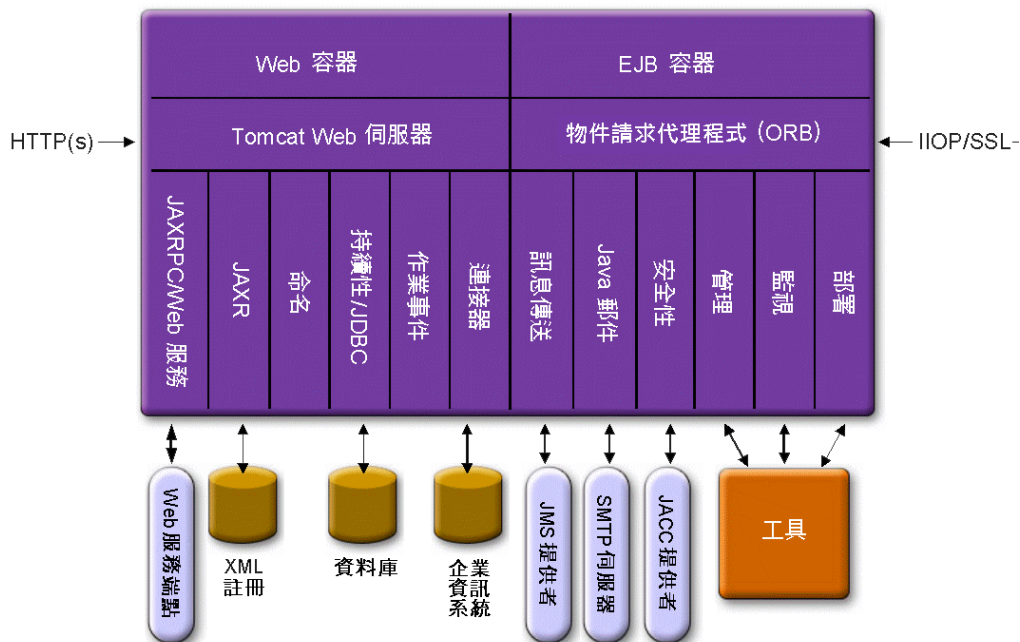


圖 1-1 Application Server 架構

- **容器** - 容器是一種執行階段環境，它為 J2EE 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 J2EE 容器：Web 和 EJB。Web 元件 (例如 JSP 頁面和 Servlet) 在 Web 容器內執行。企業 Java Bean 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 Java Bean 用戶端透過 IIOP 或 IIOP/SSL (安全) 協定與物件請求代理程式 (ORB) 進行通訊。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。
- **Web 服務** - 在 J2EE 平台上，可以部署一個 Web 應用程式，該應用程式可以提供由基於 XML 的 RPC 之 Java API (JAX-RPC) 實作的 Web 服務。J2EE 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** - J2EE 平台旨在使容器為應用程式提供服務。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。J2EE 應用程式通過查找物件的 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) 是一組為 J2EE 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

J2EE 平台使應用程式能夠存取應用程式伺服器之外的系統。應用程式通過稱為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。J2EE 平台可以通過以下 API 和元件存取外部系統：

- **JDBC** - 一種資料庫管理系統 (DBMS)，提供用於儲存、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為持續性資訊。Application Server 束包括 Java DB 資料庫管理系統。
- **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** - J2EE 連接器架構允許 J2EE 應用程式和現有企業資訊系統 (EIS) 之間的整合。應用程式通過稱為連接器或資源配接卡的可攜式 J2EE 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 的管理介面。管理工具使用這些介面與 Application Server 進行通訊。

管理工具

有多種不同的工具和 API 可用於管理 Sun Java System Application Server：

- 第 28 頁的「[Administration Console](#)」
- 第 29 頁的「[指令行介面 \(asadmin 公用程式\)](#)」
- 第 29 頁的「[JConsole](#)」
- 第 30 頁的「[Application Server Management Extension \(AMX\)](#)」

Administration Console

Administration Console 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 (也稱為 Domain Administration Server 或 DAS) 必須在執行狀態下，才能使用 Administration Console。您要有管理伺服器主機名稱和連接埠號才能啟動 Administration Console。預設管理伺服器的預設管理伺服器連接埠號是 4849。您還要有管理使用者名稱和密碼，才能登入 Administration Console。如需更多詳細資訊，請參閱相關的章節。

若要啟動 Administration Console，請在 Web 瀏覽器中鍵入以下內容：

`https://hostname:port`

例如：

`https://kindness.sun.com:4849`

如果 Administration Console 在執行管理伺服器的機器上執行，則可將主機名稱指定為 `localhost`。

在 Windows 上，從 [開始] 功能表啟動 Application Server Administration Console。

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。您可以執行由 Administration Console 所提供的相同管理作業集。可透過 Shell 的指令提示或其他程序檔或程式呼叫 asadmin 公用程式。asadmin 公用程式安裝於 *install-dir/bin* 目錄下。Sun Java System Application Server 在 Solaris 上的預設安裝根目錄是 */opt/SUNWappserver*。

若要啟動 asadmin 公用程式，請移至 *install-dir/bin* 目錄並輸入：

```
$ asadmin
```

若要列示 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。同時，Web 上也提供這些線上手冊 (HTML 格式)，請參閱「Sun Java System Application Server Enterprise Edition 8.2 Reference Manual」。

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 (預設為 8686)，然後選取 [連線]。

- [進階] 標籤：將 `JMXServiceURL` 識別為服務：`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 Management eXtension 是一個 API，它可顯示所有 Application Server 配置，並可將 JMX 管理 Bean 做為實作 AMX 介面的、易於使用的用戶端動態代理伺服器來進行監視。

如需有關使用 Application Server Management Extension 的更多資訊，請參閱「Sun Java System Application Server Enterprise Edition 8.2 Developer's Guide」中的第 16 章「Using the Java Management Extensions (JMX) API」。

Application Server 指令和概念

Sun Java System Application Server 由一或多個網域組成。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 (也稱為 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 30 頁的「網域」
- 第 31 頁的「Domain Administrative Server (DAS)」
- 第 31 頁的「叢集」
- 第 31 頁的「節點代理程式」
- 第 32 頁的「伺服器實例」
- 第 34 頁的「應用程式伺服器指令」

網域

網域是一起管理的實例群組。但是，一個應用程式伺服器實例只能屬於一個網域。除了管理界限外，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應用程式伺服器實例的特定群組 (網域)。不同的組織和管理員將所有伺服器實例分成個別的網域，就能共用單一的 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 (名為 `domain1`)。另外也會建立相關聯的網域管理伺服器 (名為 `server`)。您必須提供管理伺服器連接埠號。預設的管理伺服器連接埠是 4849。安裝程式還會查詢管理使用者名稱和密碼。安裝之後，還可以建立其他管理網域。

Domain Administrative 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。Administration Console 會與特定的 DAS 進行通訊，以管理相關聯的網域。每個 Administration Console 階段作業都可讓您配置並管理特定的網域。

Domain Administration Server (DAS) 是特別指定的應用程式伺服器實例，負責託管管理應用程式。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 有時也稱為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，因此被稱為預設伺服器。DAS 即是具有附加管理功能的伺服器實例。

每個 Administration Console 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，則必須啟動其他 Administration Console 階段作業以管理其他網域。為 Administration Console 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。

叢集

叢集是已命名之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，啟動、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，並重新啟動失敗的程序。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，節點代理程式為「永遠開啓」的元件，而且必須加以配置，使其由原生 O/S 節點啟動程式啟動 (如 Solaris/Linux `inetd`，或做為 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 J2EE 相容的單一 Java 虛擬機器，在單一節點上託管 J2EE 1.4 Application Server。每個伺服器實例在網域中都有唯一的名稱。叢集的伺服器實例為叢集的成員之一，其所接受的應用程式、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。

應用程式伺服器實例構成了應用程式部署的基礎。每個實例都屬於單一網域。除了 DAS 之外，每個伺服器實例都必須包含節點代理程式名稱的參照，該名稱定義實例將常駐的機器。

若您的拓樸包含遠端伺服器實例 (DAS 以外的伺服器實例)，請建立節點代理程式來管理遠端伺服器實例並協助其運作。節點代理程式負責建立、啟動、停止和刪除伺服器實例。使用指令行介面指令來設定節點代理程式。[圖 1-2](#) 詳細說明了應用程式伺服器實例。

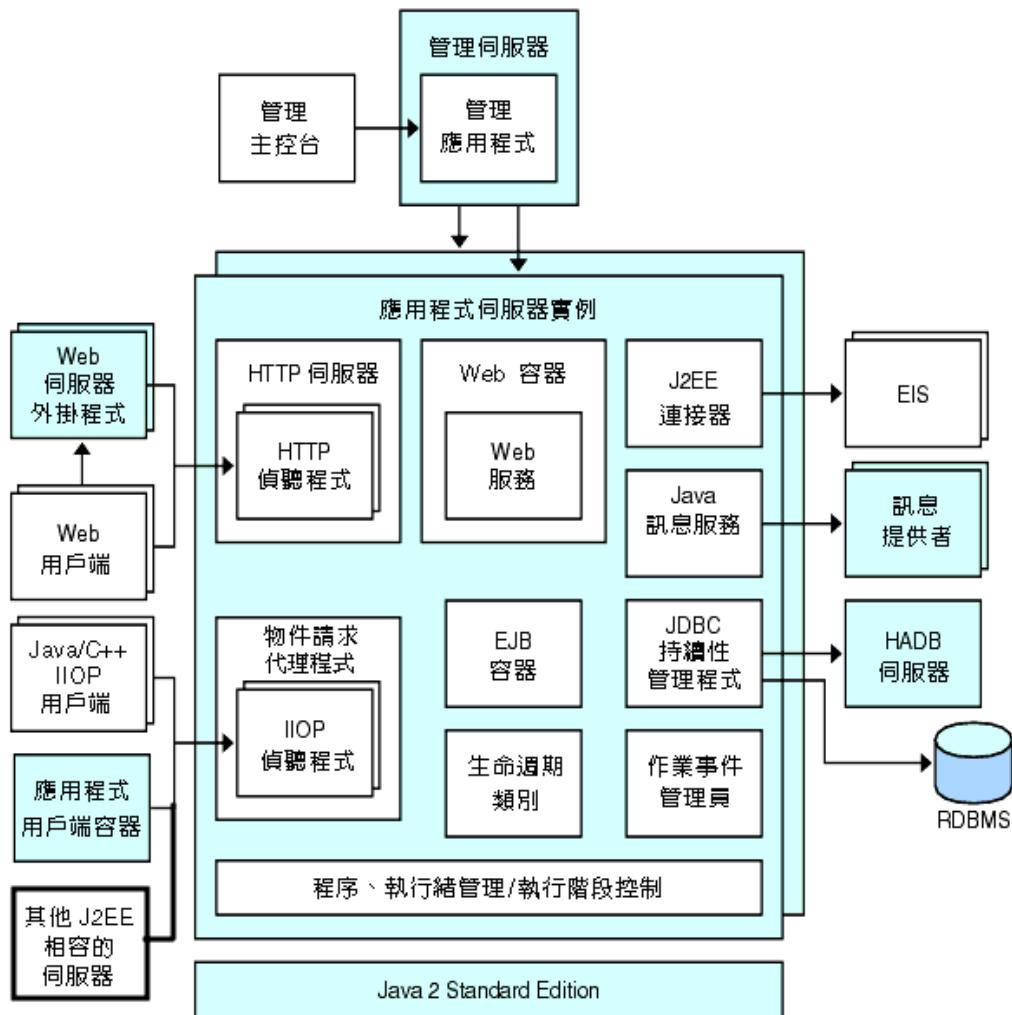


圖 1-2 Application Server 實例

Sun Java System Application Server 在安裝時會建立一個稱為 **server** 的應用程式伺服器實例。對於許多使用者而言，一個應用程式伺服器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附加的應用程式伺服器實例。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應用程式伺服器實例，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，您也可以針對每個應用程式伺服器實例建立虛擬伺服器。在單一安裝的應用程式伺服器實例內，您可以為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。這些虛擬伺服器不擴充應用程式伺服器實例。如需有關虛擬伺服器的更多資訊，請參閱第 12 章。

在作業部署中，您可以使用虛擬伺服器代替多重應用程式伺服器實例，用於多種目的。但是，如果虛擬伺服器不能滿足需求，您也可以使用多個應用程式伺服器實例。若您停止應用程式伺服器實例，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。

應用程式伺服器指令

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 34 頁的「建立網域」
- 第 35 頁的「刪除網域」
- 第 35 頁的「列示網域」
- 第 35 頁的「啟動網域」
- 第 35 頁的「在 Windows 上啟動預設網域」
- 第 36 頁的「停止網域」
- 第 36 頁的「在 Windows 上停止預設網域」
- 第 36 頁的「重新啟動網域」
- 第 36 頁的「建立叢集」
- 第 36 頁的「啟動叢集」
- 第 36 頁的「停止叢集」
- 第 37 頁的「建立節點代理程式」
- 第 37 頁的「啟動節點代理程式」
- 第 37 頁的「停止節點代理程式」
- 第 37 頁的「建立實例」
- 第 38 頁的「啟動實例」
- 第 38 頁的「停止實例」
- 第 38 頁的「重新啟動實例」
- 第 38 頁的「重新建立網域管理伺服器」
- 第 40 頁的「變更管理員密碼」

建立網域

網域是使用 `create-domain` 指令建立的。以下範例指令將建立名為 `mydomain` 的網域。Administration Server 在連接埠 1234 上進行偵聽，管理使用者名為 `hanan`。該指令提示您輸入管理密碼和主密碼。

```
$ asadmin create-domain --adminport 1234 --adminuser hanan mydomain
```

若要為 `mydomain` 網域啟動 Administration Console，請在瀏覽器中輸入以下 URL：

```
http://hostname:1234
```

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。如需完整的指令語法，請鍵入 `asadmin help create-domain`。

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，若要刪除名為 `mydomain` 的網域，請鍵入以下指令：

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

列示網域

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

```
$ asadmin list-domains
```

若要列示在其他目錄中建立的網域，請指定 `--domaindir` 選項。

啟動網域

啟動網域時，將啟動管理伺服器和應用程式伺服器實例。啟動應用程式伺服器實例之後，應用程式伺服器實例將持續執行、偵聽並接受請求。必須單獨啟動各個網域。

若要啟動網域，請鍵入 `asadmin start-domain` 指令並指定網域名稱。例如，若要啟動預設網域 (`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 上啟動預設網域

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

停止網域

停止網域將關閉該網域的管理伺服器 and 應用程式伺服器實例。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，Administration Console 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`。

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]。

重新啟動網域

重新啟動伺服器與重新啟動網域相同。若要重新啟動網域或伺服器，請停止然後再啟動網域。

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為 `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`。

啟動叢集

叢集是使用 `start-cluster` 指令啟動的。以下範例啟動了名為 `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
```

`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 start-cluster`。啟動某叢集後，即會啟動該叢集中的所有伺服器實例。沒有伺服器實例的叢集無法啟動。

停止叢集

叢集是使用 `stop-cluster` 指令停止的。以下範例停止了名為 `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啟動。

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為 `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`。該指令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`。

啟動節點代理程式

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啟動的。例如，若要啟動節點代理程式 `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`。

建立實例

伺服器實例是使用 `create-instance` 指令建立的。以下範例將建立一個名為 `myinstance` 的實例。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`。該指令提示您輸入管理密碼。

以下範例將建立一個名為 `myinstance` 的叢集伺服器實例。該指令提示您輸入管理密碼。

```
$ asadmin create-instance --host myhost --port 1234
--user admin --cluster mycluster --nodeagent mynodeagent myinstance
```

管理伺服器主機是 `myhost`，管理連接埠是 `1234`，管理使用者名稱是 `admin`，此伺服器實例所屬的叢集是 `mycluster`，管理此伺服器實例的節點代理程式是 `mynodeagent`。

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

若要建立獨立的伺服器實例，請勿指定 `--cluster` 選項。

以下範例將建立一個名為 `myinstance` 的獨立伺服器實例，由名為 `mynodeagent` 的節點代理程式所管理。

```
$ asadmin create-instance --host myhost --port 1234
--user admin --nodeagent mynodeagent myinstance
```

啓動實例

伺服器實例是使用 `start-instance` 指令啓動的。以下範例啓動了名為 `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

伺服器實例是使用 `stop-instance` 指令停止的。以下範例停止了名為 `myinstance` 的實例。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` 可予以叢集或保持獨立。

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安裝時上相同。
為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。若要啟動指令行互動模式，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。
只有具有相同架構並具有完全相同的安裝路徑 (即，兩台機器使用相同的 `install-dir` 和 `domain-root-dir`) 的兩台機器才支援備份網域的復原。
- 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為：`?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` 並將其替代為 `machine3`。這樣，您就可以將：

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

變更為：

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

7 將：

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

變更為：

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

8 在 `machine3` 上啟動復原的網域：

```
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 `install-dir/nodeagents/nodeagent/agent/config/das.properties` 中的 `agent.das.host` 特性值。

11 在 `machine2` 上重新啟動節點代理程式。

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

變更管理員密碼

若要重設管理員密碼，必須先停止網域中所有的節點代理程式。這會停止所有相關聯的伺服器實例。現在，所有的伺服器實例和節點代理程式都已停止，只有 Domain Administration Server (DAS) 仍在執行。

接著您就可以依照下列指示，變更管理使用者的密碼：

1. 使用指令行介面變更管理密碼：

```
asadmin update-file-user --authrealmname admin-realm ... --userpassword  
newpassword <admin-user-name>
```

2. 使用 Admin Console 變更管理密碼：

選取管理伺服器的配置節點 > [安全性] > [範圍] > [管理範圍] > [編輯檔案範圍使用者]。

這時會出現一則訊息，指示您已成功地變更管理員密碼。

3. 請依照下列說明，使用新密碼重新啟動 Domain Admin Server (DAS)：

使用指令行介面：`asadmin start-domain --user admin --password newpassword domain1`

假設以下配置：一個網域具有兩個節點代理程式 (`il-na`, `c1-na`) 以及三個實例 (位於名為 `c1` 之同一叢集的 `c1i1` 和 `c1i2`，以及一個獨立的伺服器實例 `il`)。

4. 使用新密碼重新啟動節點代理程式，但不啟動實例。

例如：

```
asadmin start-node-agent --user admin --password newpassword
--startinstances=false il-na asadmin
```

```
asadmin start-node-agent --user admin --password newpassword
--startinstances=false c1-na
```

5. 重新啟動伺服器和叢集。

```
asadmin start-node-agent --user admin --password newpassword ... c1
```

```
asadmin start-node-agent --user admin --password newpassword il
```

Application Server 配置

Sun Java System Application Server 的配置儲存在 `domain.xml` 檔案中。`domain.xml` 文件用於呈現 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 41 頁的「變更 Application Server 配置」](#)
- [第 42 頁的「Application Server 中的連接埠」](#)
- [第 42 頁的「變更 J2SE 軟體」](#)

變更 Application Server 配置

在進行以下任何配置變更時，必須重新啟動伺服器以使變更生效：

- 變更 JVM 選項
- 變更連接埠號
- 管理 HTTP 服務、IIOP 服務和 JMS 服務
- 管理執行緒池

如需說明，請參閱 [第 36 頁的「重新啟動網域」](#)。

在進行下列任何配置變更時，若已啟用動態配置，則無須重新啟動伺服器以使變更生效：

- 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
- 啟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 1-1 使用連接埠的 Application Server 偵聽程式

偵聽程式	預設連接埠號	說明
管理伺服器	4849	可透過 Administration Console 和 <code>asadmin</code> 公用程式存取網域的管理伺服器。對於 Administration Console，請在瀏覽器的 URL 中指定連接埠號。從遠端執行 <code>asadmin</code> 指令時，請使用 <code>--port</code> 選項指定連接埠號。
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	為安全通訊配置的 Web 應用程式在單獨的連接埠上進行偵聽。
IIOP	3700	企業 Bean (EJB 元件) 的遠端用戶端通過 IIOP 偵聽程式存取 Bean。
IIOP、SSL	3820/3890	另一個連接埠由為安全通訊配置的 IIOP 偵聽程式使用。
IIOP、SSL 和相互認證		另一個連接埠由為相互 (用戶端和伺服器) 認證配置的 IIOP 偵聽程式使用。
JMX	8686	另一個連接埠是由 JMX 連接器使用，以便與 DAS 進行通訊。

變更 J2SE 軟體

Application Server 依賴 Java 2 Standard Edition (J2SE) 軟體。安裝 Application Server 時，已指定 J2SE 軟體的目錄。如需變更 J2SE 軟體的說明，請參閱第 17 章。

部署應用程式

本章說明如何在 Application Server 上部署 (安裝) J2EE 應用程式。它包含以下小節：

- 第 43 頁的「部署生命週期」
- 第 44 頁的「自動部署」
- 第 45 頁的「部署未封裝的應用程式」
- 第 45 頁的「使用部署規劃」
- 第 46 頁的「使用 `deploytool` 公用程式」
- 第 46 頁的「J2EE 歸檔檔案的類型」
- 第 47 頁的「命名慣例」

部署生命週期

安裝 Application Server 並啟動網域之後，即可部署 (安裝) J2EE 應用程式和模組。在部署期間和變更應用程式時，應用程式或模組可能會經過以下階段：

1. 初始部署

部署應用程式或模組之前，請啟動網域。

將應用程式或模組部署 (安裝) 到特定的獨立伺服器實例或叢集。由於應用程式和模組封裝在歸檔檔案中，因此在部署期間應指定歸檔檔案名稱。預設為部署到預設伺服器實例 `server`。

如果部署到伺服器實例或叢集，則應用程式或模組將存在於網域的中央儲存庫中，並由部署到的所有叢集或伺服器實例參照為目標。

您還可以使用 `asadmin deploy` 指令 (而非 Administration Console) 部署到網域。若只將應用程式或模組部署到網域，它會存在於網域的中央儲存庫中，但在您增加參照之前，不會有任何伺服器實例或叢集參照它。

部署是動態的：部署應用程式或模組後，無需重新啟動伺服器實例即可使用應用程式。如果重新啟動了伺服器實例，所有已部署的應用程式和模組仍將處於部署狀態並且可用。

2. 啟用或停用

依預設，將啟用已部署的應用程式或模組，這表示如果應用程式或模組已部署到可存取的伺服器實例或叢集，則可以執行該應用程式或模組並且可由用戶端對其進行存取。若要防止存取，請停用應用程式或模組。在部署之後，已停用的應用程式或模組並未從網域中解除安裝，而且可以輕鬆地將其啟用。

3. 新增或刪除已部署應用程式或模組的目標

部署後，應用程式或模組將存在於中央儲存庫中，並可由多個伺服器實例和/或叢集參照。最初，做為目標部署到的伺服器實例或叢集將參考應用程式或模組。

在部署應用程式或模組之後，若要變更參照應用程式或模組的伺服器實例和叢集，請使用 **Administration Console** 變更應用程式或模組的目標，或使用 `asadmin` 工具變更應用程式參照。由於應用程式本身儲存在中央儲存庫中，因此新增或刪除目標將新增或刪除不同目標上同一版本的應用程式。但是，可以在一個目標上啟用而在另一個目標上停用部署到多個目標的應用程式，因此即使應用程式被一個目標參考，也只有在該目標上啟用它時使用者才能對其進行使用。

4. 重新部署

若要替代已部署的應用程式或模組，請將其重新部署。重新部署將自動取消部署之前已部署的應用程式或模組，並代之以新應用程式或模組。

當透過 **Administration Console** 重新部署時，重新部署的應用程式或模組將部署到網域中，並且所有參照該應用程式或模組的獨立或叢集伺服器實例將自動接收新的版本 (如果已啟用動態重新配置)。如果使用 `asadmin deploy` 指令來重新部署，請將 `domain` 指定為目標。

對於生產環境，請使用捲動升級 (升級應用程式而不中斷服務)。

5. 取消部署

若要解除安裝應用程式或模組，請取消部署應用程式或模組。

自動部署

自動部署功能使您能夠透過將預先封裝的應用程式或模組複製到 `domain-dir/autodeploy` 目錄來部署該應用程式或模組。

例如，將名為 `hello.war` 的檔案複製到 `domain-dir/autodeploy` 目錄。若要取消部署應用程式，請從 `autodeploy` 目錄中移除 `hello.war` 檔案。

您也可以使用 **Administration Console** 或 `asadmin` 工具來取消部署應用程式。在這種情況下，歸檔檔案將保留。

備註 – 自動部署僅適用於預設伺服器實例。

自動部署功能旨在用於開發環境。它與階段作業持續性 (一種生產環境功能) 不相容。如果已啟用自動部署，請勿啟用階段作業持續性

部署未封裝的應用程式

此功能適用於進階開發者。

使用目錄部署僅部署到預設伺服器實例 (server)。您不能使用它來部署到叢集或獨立伺服器實例。

包含未封裝的應用程式或模組的目錄有時稱為展開的目錄。目錄的內容必須與對應的 J2EE 歸檔檔案內容相符。例如，如果部署某一目錄中的 Web 應用程式，則該目錄的內容必須與對應的 WAR 檔案的內容相同。如需有關必需的目錄內容的資訊，請參閱相應的規格。

您可以直接在展開的目錄中變更部署描述元檔案。

如果您的環境配置為使用動態重新載入，則還可以從目錄中動態重新載入已部署的應用程式。如需更多資訊，請參閱第 184 頁的「配置進階選項」。

使用部署規劃

此功能適用於進階開發者。

部署規劃是指僅包含特定於 Application Server 的部署描述元的 JAR 檔案。有關這些部署描述元 (例如 `sun-application.xml`) 的說明，請參閱「Application Server Developer's Guide」。部署規劃是「JSR 88: J2EE Application Deployment」實作的一部分。使用部署規劃可以部署不包含特定於 Application Server 的部署描述元的應用程式或模組。

若要使用部署規劃進行部署，請指定 `asadmin deploy` 指令的 `--deploymentplan` 選項。例如，以下指令將根據 `mydeployplan.jar` 檔案中指定的規劃來部署 `myrosterapp.ear` 檔案中的企業應用程式。

```
$ asadmin deploy --user admin ---deploymentplan mydeployplan.jar myrosterapp.ear
```

在企業應用程式 (EAR) 的部署規劃檔案中，`sun-application.xml` 檔案位於根目錄下。根據以下語法來儲存每個模組的部署描述元：`module-name.sun-dd-name`，其中 `sun-dd-name` 取決於模組類型。如果模組包括 CMP 映檔檔案，則該檔案將命名為 `module-name.sun-cmp-mappings.xml`。`.dbschema` 檔案儲存在根層級目錄下，並用磅符號 (#) 替代每個正斜線字元 (/)。下面列示的內容顯示了企業應用程式 (EAR) 的部署規劃檔案的結構。

```
$ jar -tvf mydeployplan.jar
420 Thu Mar 13 15:37:48 PST 2003 sun-application.xml
370 Thu Mar 13 15:37:48 PST 2003 RosterClient.war.sun-web.xml
418 Thu Mar 13 15:37:48 PST 2003 roster-ac.jar.sun-application-client.xml
1281 Thu Mar 13 15:37:48 PST 2003 roster-ejb.jar.sun-ejb-jar.xml
2317 Thu Mar 13 15:37:48 PST 2003 team-ejb.jar.sun-ejb-jar.xml
```

```
3432 Thu Mar 13 15:37:48 PST 2003 team-ejb.jar.sun-cmp-mappings.xml
84805 Thu Mar 13 15:37:48 PST 2003 team-ejb.jar.RosterSchema.dbschema
```

在 Web 應用程式或模組檔案的部署規劃中，特定於 Application Server 的部署描述元位於根層級目錄下。如果獨立 EJB 模組包括 CMP Bean，則部署規劃包括位於根層級的 sun-cmp-mappings.xml 和 .dbschema 檔案。在下面列示的內容中，部署規劃介紹了 CMP Bean。

```
$ jar r -tvf myotherplan.jar
3603 Thu Mar 13 15:24:20 PST 2003 sun-ejb-jar.xml
3432 Thu Mar 13 15:24:20 PST 2003 sun-cmp-mappings.xml
84805 Thu Mar 13 15:24:20 PST 2003 RosterSchema.dbschema
```

使用 deploytool 公用程式

適用於軟體開發者的 deploytool 公用程式可以封裝並部署 J2EE 應用程式和模組。如需有關如何使用 deploytool 的說明，請參閱「The J2EE 1.4 Tutorial」。

J2EE 歸檔檔案的類型

軟體供應商將應用程式或模組封裝在歸檔檔案中。若要部署應用程式或模組，請指定歸檔檔案名稱。歸檔檔案的內容和結構是按照 J2EE 平台的規格定義的。J2EE 歸檔檔案的類型有以下幾種：

- Web 應用程式歸檔 (WAR)：WAR 檔案由 Servlet 和 JSP 等 Web 元件以及靜態 HTML 頁面、JAR 檔案、標籤檔案庫和公用程式類別組成。WAR 檔案名稱具有 .war 副檔名。
- EJB JAR：EJB JAR 檔案包含一個或多個企業 Bean (用於 EJB 技術的元件)。EJB JAR 檔案還包括企業 Bean 所需的任何公用程式類別。EJB JAR 檔案的名稱具有 .jar 副檔名。
- J2EE 應用程式用戶端 JAR：該 JAR 檔案包含藉由 RMI/IIOP 存取伺服器端元件 (如企業 Bean) 的 J2EE 應用程式用戶端的程式碼。在 Administration Console 中，J2EE 應用程式用戶端被稱為「應用程式用戶端」。J2EE 應用程式用戶端 JAR 檔案的名稱具有 .jar 副檔名。
- 資源配接卡歸檔檔案 (RAR)：RAR 檔案存有資源介面。資源介面是按照 J2EE 連接器架構規格定義的，它是允許企業 Bean 和 Web 元件和應用程式用戶端存取資源和外部企業系統的可攜式元件。資源介面經常稱為連接器。RAR 檔案名稱具有 .rar 副檔名。
- 企業應用程式歸檔 (EAR)：EAR 檔案存有一個或多個 WAR 檔案、EJB JAR 檔案、RAR 檔案或 J2EE 應用程式用戶端 JAR 檔案。EAR 檔案名稱具有 .ear 副檔名。

軟體供應商可以將應用程式組譯為單一 EAR 檔案或多個獨立的 WAR 檔案、EJB JAR 檔案和應用程式用戶端 JAR 檔案。在管理工具中，用於所有類型檔案的部署頁面和指令都是類似的。

命名慣例

在指定網域中，已部署的應用程式名稱和模組名稱必須是專屬名稱。

- 如果使用 Administration Console 進行部署，請在 [應用程式名稱] 欄位中指定名稱。
- 如果使用 `asadmin deploy` 指令進行部署，則應用程式或模組的預設名稱爲要部署的 JAR 檔案的前綴。例如，如果部署 `hello.war` 檔案，則 Web 應用程式的名稱爲 `hello`。若要置換預設名稱，請指定 `--name` 選項。

在一個應用程式中，不同類型的模組可以具有相同的名稱。部署應用程式時，將使用 `_jar`、`_war` 和 `_rar` 後綴來命名儲存個別模組的目錄。一個應用程式內，類型相同的模組必須具有專屬名稱。此外，在一個應用程式內，資料庫綱目檔的名稱必須是專屬名稱。

建議將類似於 Java 套裝軟體的命名機制用於在 `ejb-jar.xml` 檔案的 `<module-name>` 部分中找到的模組檔案名稱、EAR 檔案名稱、模組名稱，以及在 `ejb-jar.xml` 檔案的 `<ejb-name>` 部分找到的 EJB 名稱。使用這種類似於套裝軟體的命名機制可以確保不會發生名稱衝突。該命名慣例的優勢不僅適用於 Application Server，也適用於其他 J2EE Application Server。

EJB 元件的 JNDI 查詢名稱也必須是專屬名稱。建立連續的命名慣例可能會非常有用。例如，將應用程式名稱和模組名稱附加到 EJB 名稱中是保證名稱爲專屬名稱的方式。在這種情況下，`mycompany.pkging.pkgingEJB.MyEJB` 即爲模組 `pkgingEJB.jar` 內 EJB 的 JNDI 名稱，該模組封裝於應用程式 `pkging.ear` 中。

請確定該套裝軟體和檔案的名稱不含有空格或作業系統不支援的非法字元。

JDBC 資源

JDBC 資源 (資料源) 為應用程式提供連線至資料庫的方法。通常，管理員要為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，可以為一個資料庫建立多個 JDBC 資源。)

本章包括下列小節：

- 第 49 頁的「建立 JDBC 資源」
- 第 50 頁的「建立 JDBC 連線池」
- 第 52 頁的「JDBC 資源和連線池如何協同工作」
- 第 53 頁的「設定資料庫存取」
- 第 53 頁的「持續性管理程式資源」

建立 JDBC 資源

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

JDBC 資源 (資料源) 為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。

若要建立 JDBC 資源，請指定識別資源的專屬 JNDI 名稱。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 名稱可為 `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 Administration Console 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`。

若要使用 Admin Console 來建立 JDBC 資源，請選取 [資源] > [JDBC 資源]。指定資源設定，如下所示：

- JNDI 名稱：指定唯一的名稱。JNDI 名稱可用來為分散式運算環境中的元件加以組織與定位，就如同圖書館裡的卡片目錄，可用來將書籍分門別類並表示擺放位置。因此，JNDI 名稱便成為存取 JDBC 資源的重要方法。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。
- 池名稱：選擇要和新的 JDBC 資源相關聯的連線池。
- 說明：鍵入資源的簡短說明。
- 狀態：如果要使資源不可用，請取消選取 [啓用] 核取方塊。依預設，建立資源之後立即可以使用資源 (已啓用)。

若要使用指令行公用程式建立 JDBC 資源，請使用 `create-jdbc-resource` 指令。

建立 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。

在您建立連線池時，實際上就是在定義與特定資料庫連線的相關事項。建立池之前，您必須首先安裝並整合 JDBC 驅動程式。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。

必須輸入 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：若 JDBC 驅動程式具有資源類型和資料庫的資料來源類別，則必須輸入 [資料來源類別名稱] 欄位值。
- 必需的特性，例如資料庫名稱 (URL)、使用者名稱和密碼

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 Administration Console 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的各個方面。

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

定義您所安裝的 JDBC 驅動程式所指定的一般設定值。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
DataSource Class Name	實作 DataSource 和/或 XADataSource API 的供應商特定的類別名稱。該類別位於 JDBC 驅動程式中。
Resource Type	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)。

另外，您必須定義一組常駐於池中的實體資料庫連線。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。該值還確定了首次建立池或應用程式伺服器啟動時置於池中的連線的數目。
池的最大大小	池中連線的最大數目。
池設定大小數量	當池向最小池大小收縮時，將成批調整大小。此值確定批次中的連線數目。將該值設置過大會延遲連線循環；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。
最長等待時間	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。

選擇性地，應用程式伺服器可以在將連線傳送給應用程式之前驗證連線。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	<p>應用程式伺服器可以使用三種方法來驗證資料庫連線：auto-commit、metadata 和 table。</p> <p>auto-commit 和 metadata - Application Server 透過呼叫 <code>con.setAutoCommit()</code> 方法和 <code>con.getMetaData()</code> 方法來驗證連線。但是，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。</p> <p>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。</p>
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	選取標記為 [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。

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
作業事件隔絕	使您可以為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。
受保證隔絕層級	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 <code>con.setTransactionIsolation</code>) 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。

JDBC 資源和連線池如何協同工作

為了儲存、組織和擷取資料，大多數應用程式均使用關聯式資料庫。Java EE 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：

- 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。

應用程式伺服器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 呼叫翻譯為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。

應用程式伺服器將連線傳回連線池。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，您必須先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。接著您必須安裝資料庫產品的 JDBC 驅動程式，讓網域的伺服器實例能夠存取驅動程式的 JAR 檔案。然後，建立資料庫、資料庫連線池以及指向連線池的 JDBC 資源。

安裝 JDBC 驅動程式之後，還必須加以整合，讓驅動程式將應用程式的 JDBC 呼叫轉譯為資料庫伺服器的協定。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：

- 使通用類別載入器可以存取該驅動程式。
 1. 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. 重新啟動該網域。
- 使系統類別載入器可以存取該驅動程式。
 1. 在 Administration Console 上，選取所需配置的 [JVM 設定]。
 2. 指出驅動程式 JAR 檔案的完全合格路徑名稱。
 3. 儲存設定並重新啟動伺服器。

持續性管理程式資源

向下相容性需要此功能。若要在 Application Server 7 版本上執行，使用容器管理的持續性 Bean (一種 EJB 元件) 的應用程式需要持續性管理員資源。建議改為使用 JDBC 資源。

若要使用 Admin Console 建立持續性管理程式資源，請選取 [資源] 節點。[持續性管理程式] 節點 > [持續性管理程式] 頁面。若要使用命令行公用程式，請使用 `create-persistence-resource` 指令。

配置 Java 訊息服務資源

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

- 第 55 頁的「關於 JMS 資源」
- 第 57 頁的「JMS 連線工廠」
- 第 57 頁的「JMS 目標資源」
- 第 58 頁的「JMS 實體目標」
- 第 58 頁的「JMS 提供者」
- 第 59 頁的「外來的 JMS 提供者」

關於 JMS 資源

- 第 55 頁的「Application Server 中的 JMS 提供者」
- 第 56 頁的「JMS 資源」
- 第 57 頁的「JMS 資源和連接器資源的關係」

Application Server 中的 JMS 提供者

Application Server 透過將 Sun Java System Message Queue (原來稱為 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 Administration Console。對於進階作業 (包括管理 Message Queue 叢集)，請使用 *MQ-install-dir/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」。

JMS 資源

Java 訊息服務 (JMS) API 使用兩種受管理物件：

- 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。

連線工廠分為三種類型：

- QueueConnectionFactory 物件，用於點對點通訊
- TopicConnectionFactory 物件，用於出版訂閱通訊
- ConnectionFactory 物件，可用於點對點通訊和出版訂閱通訊；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

「J2EE 1.4 Tutorial」中有關 JMS 的章節提供了有關此兩種通訊和 JMS 其他方面的詳細資訊(請參閱 <http://java.sun.com/j2ee/1.4/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。
- 在訊息導引 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 透過使用名為 `jsra` 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 Administration Console 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 Administration Console 的 [連接器] 節點 (而非 [JMS 資源] 節點) 為 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 連線工廠為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 的 Admin Console 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 目標可做為訊息的儲存庫。您可以使用 Admin Console 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。建議的作業是使用 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。如需有關這些設定和其他特性的詳細資訊，請參閱 Admin Console 線上說明。

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

提示 – 若要為 `asadmin create-jms-resource` 指令指定 `addresslist` 特性 (以 `host:mqport,host2:mqport,host3:mqport` 的格式)，請使用 `\\` 以退出 `:`。例如，`host1\\:mqport,host2\\:mqport,host3\\:mqport`。

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 Admin Console 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 `topic` 或 `queue`。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 Admin Console 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。

JMS 提供者

配置 JMS 提供者的一般特性

使用 Admin Console 的 [JMS 服務] 頁面來配置所有 JMS 連線將使用的特性。於 Admin Console，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)。

- 以秒為單位指定 [重新連線間隔]。此屬性適用於對 AddressList 中每個位址的重新連線嘗試，及對該清單中連續位址的重新連線嘗試。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。在欄位中，鍵入用戶端執行階段嘗試連線 (或重新連線) AddressList 清單中每個位址的次數，到達這個值後，用戶端執行階段將嘗試連線清單中的下一個位址。
- 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 AddressList 中的位址順序 (priority)，還是按隨機順序 (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 中循環的次數。
- 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，必須重新啟動應用程式伺服器。預設的 JMS 主機是唯一無須重新啟動應用程式伺服器就能更新的特性。

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為遠端系統會使所有 JMS 應用程式都在遠端伺服器上執行。若要在使用的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

JMS 通用資源配接卡為 Java EE Connector 1.5 資源配接卡，可包裝外部 JMS 提供者 (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 應用程式伺服器 (如 Sun Java System Application Server) 整合起來。該配接卡是 .rar 歸檔，可藉由 Java EE 1.4 應用程式伺服器的管理工具加以部署和配置。

配置 JMS 的通用資源配接卡

應用程式伺服器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 做為整合方式。在這種情況下，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 `javabeans` 類別的 Java 反射來做為整合模式。您可以使用 Sun Java System Application Server 的 Administration Console 或 CLI 來配置資源配接卡。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，應用程式伺服器應該能夠使用 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，應用程式伺服器 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。

如需執行這項作業的步驟，請參閱 Admin Console 線上說明。在部署期間，請確實將通用資源配接卡的位置指定為

`install-dir/lib/addons/resourceadapters/genericjmsra/genericra.rar`。另外，您也必須指定第 61 頁的「資源配接卡特性」小節中說明的特性。

2. 建立連接器連線池

如需執行這項作業的步驟，請參閱 Admin Console 線上說明。在 [新建連接器連線池] 頁面上，從 [資源配接卡] 組合方塊中選取 `genericra`。另外，請在 [連線定義] 組合方塊中，選取 `javax.jms.QueueConnectionFactory`，並指定在第 63 頁的「ManagedConnectionFactory 特性」小節中所說明的特性。

3. 建立連接器資源。

如需執行這項作業的詳細程序，可以參照 Admin Console 線上說明。在 [新建連接器資源] 頁面上，選取您在先前的步驟中建立的池。

4. 建立管理物件資源。

如需執行這項作業的詳細程序，可以參照 Admin Console 線上說明。在 [新建管理物件資源] 頁面上，選取 `genericra` 做為 [資源配接卡]，以及 `javax.jms.Queue` 做為 [資源類型]。按 [下一步]，並在第二頁按一下 [增加特性]。在 [其他特性] 表格中，指定帶有 `Name\\=clientQueue` 值且名為 `DestinationProperties` 的新特性。如需有關更多特性的資訊，請參閱第 64 頁的「受管理物件資源特性」小節的說明。

5. 在 Sun Java System Application Server 上，對安全性策略進行如下變更。

- 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	javabeans/jndi	javabeans	決定資源配接卡和 JMS 用戶端之間的整合模式。
ConnectionFactoryClassName	應用程式伺服器類別路徑中可供使用的類別名稱，例如： <code>com.sun.messaging.ConnectionFactory</code>	無	JMS 用戶端的 <code>javax.jms.ConnectionFactory</code> 實作的類別名稱。會在 <code>ProviderIntegrationMode</code> 為 <code>javabeans</code> 時加以使用。
QueueConnectionFactoryClassName	應用程式伺服器類別路徑中可供使用的類別名稱，例如： <code>com.sun.messaging.QueueConnectionFactory</code>	無	JMS 用戶端的 <code>javax.jms.QueueConnectionFactory</code> 實作的類別名稱。會在 <code>ProviderIntegrationMode</code> 為 <code>javabeans</code> 時加以使用。
TopicConnectionFactoryClassName	應用程式伺服器類別路徑中可供使用的類別名稱，例如： <code>com.sun.messaging.TopicConnectionFactory</code>	無	JMS 用戶端的 <code>javax.jms.TopicConnectionFactory</code> 實作的類別名稱。會在 <code>ProviderIntegrationMode</code> 指定為 <code>javabeans</code> 時加以使用。
XAConnectionFactoryClassName	應用程式伺服器類別路徑中可供使用的類別名稱，例如： <code>com.sun.messaging.XAConnectionFactory</code>	無	JMS 用戶端的 <code>javax.jms.ConnectionFactory</code> 實作的類別名稱。會在 <code>ProviderIntegrationMode</code> 指定為 <code>javabeans</code> 時加以使用。
XAQueueConnectionFactoryClassName	應用程式伺服器類別路徑中可供使用的類別名稱，例如： <code>com.sun.messaging.XAQueueConnectionFactory</code>	無	JMS 用戶端的 <code>javax.jms.XAQueueConnectionFactory</code> 實作的類別名稱。會在 <code>ProviderIntegrationMode</code> 指定為 <code>javabeans</code> 時加以使用。
XATopicConnectionFactoryClassName	應用程式伺服器類別路徑中可供使用的類別名稱，例如： <code>com.sun.messaging.XATopicConnectionFactory</code>	無	JMS 用戶端的 <code>javax.jms.XATopicConnectionFactory</code> 實作的類別名稱。會在 <code>ProviderIntegrationMode</code> 是 <code>javabeans</code> 時加以使用。
TopicClassName	應用程式伺服器類別路徑中可供使用的類別名稱，例如： <code>com.sun.messaging.Topic</code>	無	JMS 用戶端的 <code>javax.jms.Topic</code> 實作的類別名稱。會在 <code>ProviderIntegrationMode</code> 是 <code>javabeans</code> 時加以使用。

特性名稱	有效值	預設值	說明
QueueClassName	應用程式伺服器類別路徑中可供使用的類別名稱，例如： <code>com.sun.messaging.Queue</code>	無	JMS 用戶端的 <code>javax.jms.Queue</code> 實作的類別名稱。會在 <code>ProviderIntegrationMode</code> 指定為 <code>javabeans</code> 時加以使用。
SupportsXA	True/false	FALSE	指定 JMS 用戶端是否支援 XA。
ConnectionFactoryProperties	以逗號分隔的「名稱-值」對。	無	這指定 JMS 用戶端的 <code>javabeans</code> 特性名稱以及 <code>ConnectionFactory</code> 的值。只有當 <code>ProviderIntegrationMode</code> 為 <code>javabeans</code> 時才會需要。
JndiProperties	以逗號分隔的「名稱-值」對。	無	這指定用來連線至 JMS 提供者之 JNDI 的 JNDI 提供者特性。只有當 <code>ProviderIntegrationMode</code> 為 <code>jndi</code> 時才會用到。
CommonSetterMethodName	方法名稱	無	這指定某些 JMS 供應商在設定其受管理物件的特性時，所使用的一般 <code>setter</code> 方法名稱。只有當 <code>ProviderIntegrationMode</code> 為 <code>javabeans</code> 時，才會用到這個特性。如果是 Sun Java System Message Queue，則此特性稱為 <code>setProperty</code> 。
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	<p>作業事件管理員使用 XAResource 上的 isSameRM 方法，來判斷由兩個 XAResources 所呈現的資源管理員實例是否相同。</p> <p>當 RMPolicy 設定為 ProviderManaged (預設值) 時，由 JMS 提供者負責決定通用資源配接卡中的 RMPolicy 和 XAResource 包裝程式，是否僅將 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。</p> <p>有些 XAResource 實作 (如 IBM MQ Series) 在進行每個實體連線時，都要仰賴資源管理員。單一作業事件中同一個佇列管理員進行內送和外送通訊時，會造成問題 (例如，當 MBD 傳送回應至目標時)。</p> <p>當 RMPolicy 設定為 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。如需有關此特性的其他資訊，請參閱 Issue Tracker 資料庫中的問題 #5 (位於 Glassfish 網站)。</p>

ManagedConnectionFactory 特性

ManagedConnectionFactory 特性是在 connector-connection-pool 建立時所指定。建立資源配接卡時指定的所有特性，都可在 ManagedConnectionFactory 中置換。其他只由 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。

特性名稱	有效值	預設值	說明
ConnectionFactoryJndiName	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 <code>clientId</code> 之外)。只有當 <code>ProviderIntegratinMode</code> 為 <code>jndi</code> 時，才會用到此特性名稱。
ConnectionValidationEnabled	true/false	FALSE	若設定為 <code>true</code> ，資源配接卡會使用異常偵聽程式來擷取任何連線異常，並傳送 <code>CONNECTION_ERROR_OCCURED</code> 事件至應用程式伺服器。

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 <code>ProviderIntegrationMode</code> 為 <code>jndi</code> 時，才會使用這個特性名稱。
DestinationProperties	以逗號分隔的「名稱-值」對	無	這指定 JMS 用戶端的 <code>javabean</code> 特性名稱以及目標值。只有當 <code>ProviderIntegrationMode</code> 為 <code>javabean</code> 時才會需要。

啓動規格特性

本小節所述特性在 MDB 的 Sun 專用部署描述元中，指定為 `activation-config-properties`。所有資源配接卡特性都可在 `ActivationSpec` 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	資源配接卡為同步傳遞訊息，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。
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。只有在 ProviderIntegrationMode 配置為 jndi 時才會使用。
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。只有在 ProviderIntegrationMode 配置為 jndi 時使用。
DestinationType	javax.jms.Queue 或 javax.jms.Topic	空	MDB 會偵聽的目標類型。
DestinationProperties	以逗號分隔的「名稱-值」對	無	這將指定 JMS 用戶端的 javabean 特性名稱以及目標值。只有當 ProviderIntegrationMode 為 javabean 時才會需要。
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 (以秒計)		當訊息在 MDB 中造成執行階段異常時，重複傳遞的間隔。

特性名稱	有效值	預設值	說明
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 <code>ProviderIntegrationMode</code> 為 <code>jndi</code> 時才加以使用。
DeadMessageDestinationClassName	目標物件的類別名稱。	無	只有當 <code>ProviderIntegrationMode</code> 為 <code>javabean</code> 時才加以使用。
DeadMessageDestinationProperties	以逗號分隔的「名稱-值」對	無	這指定 JMS 用戶端的 <code>javabean</code> 特性名稱以及目標值。只有當 <code>ProviderIntegrationMode</code> 為 <code>javabean</code> 時才會需要。
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。
ReconnectInterval	時間 (以秒計)		重新連線之間的間隔。

配置 JavaMail 資源

Application Server 包含 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、撰寫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 為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。

如需瞭解有關 JavaMail API 的更多資訊，請參閱 JavaMail 網站，網址為 <http://java.sun.com/products/javamail/index.html>

本章包括下列小節：

- 第 67 頁的「建立 JavaMail 階段作業」

建立 JavaMail 階段作業

若要配置 JavaMail 以在 Application Server 中使用，請在 Application Server 的 Admin Console 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 (利用您為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 Admin Console 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，因為 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 Admin Console 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。名稱必須可以解析為實際的主機名稱。

- 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：如果您不希望此時啟用郵件階段作業，請取消選取 [啟用] 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：選取此核取方塊可為此郵件階段作業啟用額外的除錯輸出功能，包括協定追蹤。如果將 JavaMail 記錄層級設為 `FINE` 或更精細，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。請參閱第 138 頁的「設定自訂記錄層級」，以取得有關設定記錄層級的資訊。
- 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。

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 資源有時被稱為資料庫。)每個資源物件都是由專屬的易懂名稱識別，稱為 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。

本章包括下列小節：

J2EE 命名服務

JNDI 名稱是易懂的物件名稱。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 名稱為 jdbc/Pointbase。當 Application Server 啟動時，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 J2EE 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。

J2EE 容器實作應用程式元件的環境，並將該環境做為 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，可以為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，資源參照名稱為 `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 查詢及其關聯的參照。

表 6-1 JNDI 查找及其關聯的參考

JNDI 查找名稱	關聯的參考
<code>java:comp/env</code>	應用程式環境項目
<code>java:comp/env/jdbc</code>	JDBC DataSource 資源管理程式連線 Factory
<code>java:comp/env/ejb</code>	EJB 參照
<code>java:comp/UserTransaction</code>	UserTransaction 參考
<code>java:comp/env/mail</code>	JavaMail 階段作業連線 Factory
<code>java:comp/env/url</code>	URL 連線 Factory
<code>java:comp/env/jms</code>	JMS 連線 Factory 與目標
<code>java:comp/ORB</code>	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，我們將討論如何為 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 資源的範例為：

```
<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>
```


連接器資源

連接器模組 (也稱為資源配接卡) 是一種 Java 元件，它可讓應用程式與企業資訊系統 (EIS) 互動。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。若要安裝連接器模組，請依照其他 Java 模組的部署方式來部署該模組。

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 (資源介面)。

連接器資源是為應用程式提供到 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。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 9 透過使用連接器模組 (資源配接卡) 來實作 JMS。

本章包含以下主題：

- [第 73 頁的「連接器連線池」](#)
- [第 75 頁的「連接器資源」](#)
- [第 75 頁的「受管理物件資源」](#)

連接器連線池

下表說明連線池設定：

參數	說明
池的初始大小和最小大小	池中連線的最小數目。該值還確定了首次建立池或應用程式伺服器啟動時置於池中的連線的數目。
池的最大大小	池中連線的最大數目。

參數	說明
池設定大小數量	當池向最小池大小收縮時，將成批調整大小。此值確定批次中的連線數目。將該值設置過大會延遲連線循環；而將該值設置過小則會導致效率太低。
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。
最長等待時間	已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。
一旦失敗	選取標記為 [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，則僅在使用個別連線時才會重新建立這些連線。
作業事件支援	<p>使用 [作業事件支援] 清單可以為連線池選取作業事件支援類型。選擇的作業事件支援以向下相容方式置換與此連線池關聯的資源介面中的作業事件支援屬性。也就是說，它可以支援比資源介面中指定的作業事件層級低或與其相同的作業事件層級，但它不能指定更高的層級。</p> <p>[作業事件支援] 功能表中的 [無] 選項表示資源介面不支援本機資源管理員或 JTA 作業事件，並且不實作 XAResource 或 LocalTransaction 介面。對於 JAXR 資源配接卡，您必須從 [作業事件支援] 功能表中選擇 [無]。JAXR 資源配接卡不支援本機或 JTA 作業事件。</p> <p>[本機] 作業事件支援表示資源介面將透過實作 LocalTransaction 介面來支援本機作業事件。本機作業事件的管理在資源管理員內部進行，不涉及任何外部作業事件管理員。</p> <p>XA 作業事件支援表示資源配接卡將透過實作 LocalTransaction 和 XAResource 介面來支援本機資源管理員和 JTA 作業事件。XA 作業事件由作業事件管理員在資源管理員外部進行控制和協調。本機作業事件的管理在資源管理員內部進行，不涉及任何外部作業事件管理員。</p>
連接器驗證	若要讓連線池在傳送到應用程式之前經過驗證，請選取 [已啓用] 核取方塊。

建立連線池之前，您必須先部署與該池關聯的連接器模組 (資源配接卡)。您可以使用 Admin Console 或 `asadmin` 指令來部署連接器模組。如需有關 `asadmin` 指令的資訊，請參閱 `asadmin(1M)`。

若要在 Admin Console 中檢視、建立、編輯或刪除連線池，請按一下 [資源] > [連接器] > [連接器連線池]。您可以將特性 (「名稱-值」對) 增加到連接器連線池。或者，您可以使用以下 `asadmin` 指令來建立以及刪除連線池：

- `create-connector-connection-pool(1)`
- `delete-connector-connection-pool(1)`

連接器資源

連接器資源可為應用程式提供企業資訊系統 (EIS) 連線。每個連接器資源都與連線池相關聯。若要檢視、建立、編輯或刪除連接器資源，請在 Admin Console 中按一下 [資源] > [連接器] > [連接器資源]。或者，您可以使用以下 `asadmin` 指令來建立以及刪除連線資源：

- `create-connector-resource(1)`
- `delete-connector-resource(1)`

受管理物件資源

受管理物件可為應用程式提供專用功能，例如存取資源配接卡及其關聯的 EIS 專屬的剖析器。若要檢視、建立、編輯或刪除受管理物件，請在 Admin Console 中按一下 [資源] > [連接器] > [管理物件資源]。

- `create-admin-object(1)`
- `delete-admin-object-1(1)`

J2EE 容器

本章說明如何配置伺服器中包含的 J2EE 容器。它包含以下小節：

- 第 77 頁的「J2EE 容器的類型」
- 第 78 頁的「配置 J2EE 容器」

J2EE 容器的類型

J2EE 容器為 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 26 頁的「Application Server 架構」。

Web 容器

Web 容器是容納 Web 應用程式的 J2EE 容器。Web 容器為開發者提供執行 servlet 和 JavaServer Pages (JSP 檔案) 的環境，以延伸 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 分為三種類型：階段作業 Bean、實體 Bean 和 訊息導引 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。訊息導引 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體中、快取 Bean 的狀態以及在必要時啟動或鈍化 Bean。

配置 J2EE 容器

- 第 78 頁的「配置一般 Web 容器設定」
- 第 80 頁的「配置一般 EJB 設定」
- 第 81 頁的「配置訊息導引 Bean 設定」
- 第 81 頁的「配置 EJB 計時器服務設定」

配置一般 Web 容器設定

在此發行版本中，Administration Console 中沒有用於 Web 容器的容器範圍的設定。

配置 Web 容器階段作業

本小節說明 Web 容器中的 HTTP 階段作業設定。HTTP 階段作業是唯一將狀態資料寫入持續性儲存的 Web 階段作業。

- 第 78 頁的「配置階段作業逾時值」
- 第 78 頁的「配置管理程式特性」
- 第 79 頁的「配置儲存特性」

配置階段作業逾時值

使用 Administration Console 來設定 HTTP 階段作業逾時值。階段作業逾時值表示 HTTP 階段作業有效的持續時間。

在 Administration Console 中，請移至 [配置] > [Web 容器] > [階段作業特性]。在 [階段作業逾時] 欄位中，輸入階段作業有效的秒數。

如需有關設定階段作業逾時值的詳細指示，請按一下 Administration Console 中的 [說明]。

配置管理程式特性

階段作業管理程式提供配置建立和銷毀階段作業的方式、儲存階段作業狀態的位置以及最大階段作業數的方法。

若要在 Administration Console 中變更階段作業管理程式設定，請移至 [配置] > [Web 容器] > [管理程式特性]。

- 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。

在 [管理程式特性] 標籤中，設定下列特性：

- [清除間隔] 的值。[清除間隔] 欄位是指存放區內非使用中的階段作業資料將在幾秒後刪除。
- [最大階段作業數] 的值。[最大階段作業數] 欄位是允許的最大階段作業數目。
- 設定 [階段作業檔案名稱] 的值。[階段作業檔案名稱] 欄位是包含階段作業資料的檔案。
- [階段作業 ID 產生器類別名稱] 的值。

[階段作業 ID 產生器類別名稱] 欄位使您可以指定用於產生唯一的階段作業 ID 的自訂類別。每個伺服器實例只允許有一個階段作業 ID 產生器類別，並且叢集中的所有實例必須使用同一階段作業 ID 產生器，以防止階段作業金鑰衝突。

自訂階段作業 ID 產生器類別必須實作

`com.sun.enterprise.util.uuid.UuidGenerator` 介面：

```
package com.sun.enterprise.util.uuid;

public interface UuidGenerator {

    public String generateUuid();
    public String generateUuid(Object obj); //obj is the session object
}
```

類別必須位於 `Application Server` 類別路徑中。

如需有關設定管理程式特性的詳細指示，請按一下 `Administration Console` 中的 [說明]。

配置儲存特性

若要指定階段作業存放區資料的儲存位置，請在 `Administration Console` 中，移至 [配置] > [Web 容器] > [儲存特性]。

- 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。
- 設定 [清除間隔] 欄位。
 - [清除間隔] 欄位是指存放區內非使用中的階段作業資料將在幾秒後刪除。
- 指定將用來儲存階段作業資料的目錄。

如需有關設定階段作業存放區特性的詳細指示，請按一下 `Administration Console` 中的 [說明]。

配置虛擬伺服器設定

當您安裝 Application Server 時，會為 Application Server 實例建立預設虛擬伺服器。此虛擬伺服器的預設 docroot 會建立在 *instance-dir*/domains/domain1/docroot 中，且與 *instance_name* /docroot 同步化。對於您所建立的每個額外 Application Server 實例，都會建立虛擬伺服器。

配置一般 EJB 設定

本小節描述以下適用於伺服器上所有企業 Bean 容器的設定：

- 第 80 頁的「階段作業存放區位置」
- 第 80 頁的「配置 EJB 池設定」
- 第 81 頁的「配置 EJB 快取設定」

若要置換每個容器的預設值，請調整企業 Bean 的 *sun-ejb-jar.xml* 檔案中的值。如需詳細資訊，請參閱「Application Server Developer's Guide」。

階段作業存放區位置

[階段作業存放區位置] 欄位指定在檔案系統上儲存鈍化 Bean 和持續的 HTTP 階段作業所在的目錄。

鈍化 Bean 是已將其狀態寫入到檔案系統上的檔案中的企業 Bean。通常，鈍化的 Bean 已經閒置了一段時間並且目前未被用戶端存取。

與鈍化 Bean 類似，持續的 HTTP 階段作業是已將其狀態寫入到檔案系統上的檔案中的各個 Web 階段作業。

[確定選項] 欄位用於指定容器如何快取兩次作業事件之間的鈍化實體 Bean 實例。

[選項 B] 用於快取作業事件之間的實體 Bean 實例，並且是預設選項。[選項 C] 用於停用快取。

配置 EJB 池設定

容器維護了一個企業 Bean 池，以便在不建立 Bean 來實現效能的情況下回應用戶端請求。這些設定僅適用於無狀態階段作業 Bean 和實體 Bean。

當您在使用已部署之企業 Bean 的應用程式中遇到效能問題時，建立池或增加現有池所維護的 Bean 的數目會有助於提高應用程式的效能。

依預設，容器維護企業 Bean 池。

配置 EJB 快取設定

容器維護最常用企業 Bean 的企業 Bean 資料快取。這樣，容器可以更快回應來自其他應用程式模組的企業 Bean 資料請求。本小節只適用於有狀態階段作業 Bean 和實體 Bean。

被快取的企業 Bean 處於以下三種狀態之一：使用中、閒置或已鈍化。使用中企業 Bean 是目前正被用戶端存取的企業 Bean。閒置企業 Bean 的資料目前儲存在快取中，但無用戶端存取 Bean。鈍化 Bean 的資料是暫時儲存的，如果用戶端請求此 Bean，則會將其資料讀回快取中。

配置訊息導引 Bean 設定

訊息引導 Bean 的池與第 80 頁的「[配置 EJB 池設定](#)」中說明的階段作業 Bean 的池類似。依預設，容器維護訊息導引 Bean 的池。

若要調整該池的配置，請執行以下步驟：

配置 EJB 計時器服務設定

計時器服務是由企業 Bean 容器提供的用於排程企業 Bean 使用的通知或事件的持續性和作業事件通知服務。所有企業 Bean (有狀態階段作業 Bean 除外) 均可接收計時器服務的通知。關閉或重新啟動伺服器時，不會銷毀服務設定的計時器。

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紐a。Application Server 具有基於 J2EE 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。本節論述以下主題：

- 第 83 頁的「瞭解應用程式和系統安全性」
- 第 84 頁的「管理安全性的工具」
- 第 85 頁的「管理密碼安全性」
- 第 87 頁的「關於認證與授權」
- 第 89 頁的「瞭解使用者、群組、角色和範圍」
- 第 92 頁的「憑證和 SSL 簡介」
- 第 94 頁的「關於防火牆」
- 第 95 頁的「使用 Administration Console 管理安全性」
- 第 97 頁的「使用憑證和 SSL」
- 第 110 頁的「詳細資訊」

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。做為管理員，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論述；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，因此主要論述系統安全性。

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：

- Administration Console，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 Administration Console 的一般介紹，請參閱第 28 頁的「管理工具」。如需有關使用 Administration Console 可以執行之安全性作業的簡介，請參閱第 95 頁的「使用 Administration Console 管理安全性」。
- asadmin，命令行工具，可以執行 Administration Console 能夠執行的許多作業。您還可以使用 asadmin 執行使用 Administration Console 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 28 頁的「管理工具」。
- deploytool 是一種圖形化封裝和部署工具，用於編輯應用程式部署描述元，從而控制個別應用程式的安全性。由於 deploytool 主要針對應用程式開發者，因此本文件未對該工具的使用做詳細說明。如需有關使用 deploytool 的說明，請參閱此工具的線上說明以及位於以下位置的「The J2EE 1.4 Tutorial」：
<http://java.sun.com/j2ee/1.4/docs/tutorial/doc/index.html>。

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

- keytool，命令行公用程式，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。做為管理員，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 Enterprise Edition 中，還可以使用另外兩個實作網路安全服務 (NSS) 的工具來管理安全性。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，命令行公用程式，用於管理憑證和金鑰資料庫。
- pk12util，命令行公用程式，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 的此發行版本中，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為 `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 Administration Console 增加使用者和資源，並為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：

- [第 85 頁的「對 domain.xml 檔案中的密碼進行加密」](#)
- [第 86 頁的「保護具有編碼密碼的檔案」](#)
- [第 86 頁的「變更主密碼」](#)
- [第 86 頁的「使用主密碼和金鑰庫」](#)
- [第 87 頁的「變更管理員密碼」](#)

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，此目錄為 `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 (在本範例中為 `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. 重新啟動相關網域的 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。此密碼是全局安全性的要塞點；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。使用者可以透過移除此檔案強制系統提示輸入 MP。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. 重新啟動 Application Server。



注意 – 此時，不能啟動正在執行的伺服器實例並且不能重新啟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 之前重新啟動了該伺服器實例，它將無法啟動。

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，然後重新啟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，網域就會啟動。

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 85 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
```

還可以使用 Administration Console 變更管理員密碼，如以下程序所示。

若要使用 Admin Console 變更管理員密碼，請選取 [配置] 節點 > 要配置的實例 > [安全性] 節點 > [範圍] 節點 > [admin-realm] 節點，並視需要編輯範圍頁面。

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：

- 第 87 頁的「認證實體」
- 第 88 頁的「對使用者進行授權」
- 第 88 頁的「指定 JACC 提供者」
- 第 89 頁的「稽核認證與授權決策」
- 第 89 頁的「配置訊息安全性」

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、數位憑證或其他憑證。

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，則稱為相互認證。

當實體嘗試存取受保護的資源時，Application Server 將使用為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 支援四種類型的認證，如第 87 頁的「認證實體」所示。應用程式在其部署描述元中指定所使用的認證類型。如需有關使用 `deploytool` 配置應用程式認證方法的更多資訊，請參閱位於以下位置的「The J2EE 1.4 Tutorial」：
<http://java.sun.com/j2ee/1.4/docs/tutorial/doc/index.html>。

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
基本	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。
基於表單	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。
使用者端憑證	HTTPS (基於 SSL 的 HTTP)	伺服器使用公開金鑰憑證認證使用者端。	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。其部署描述元可定義相同的群組並可使用相同認證方法 (基本、表單、摘要、憑證) 的所有 Web 應用程式均共用單次登入。

對於為 Application Server 定義的虛擬伺服器，依預設已啟用單次登入。

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。使用者的授權基於其角色。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料，但每個員工僅能檢視自己的個人資料。如需有關角色的更多資訊，請參閱第 89 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 J2EE 1.4 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。如需有關開發自訂稽核模組的資訊，請參閱「Application Server Developer's Guide」中的「*Configuring an Audit Module*」小節。

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。可以為 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。

您可以為整個 Application Server 或者為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。「Developer's Guide」中的「*Securing Applications*」一章論述了如何配置應用程式層級的訊息安全性。

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 90 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為**主體**。使用者有時被稱為**個人**。
- [第 90 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 91 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 91 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。範圍也稱為安全策略網域。

備註 – 儘管使用者和群組是為整個 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

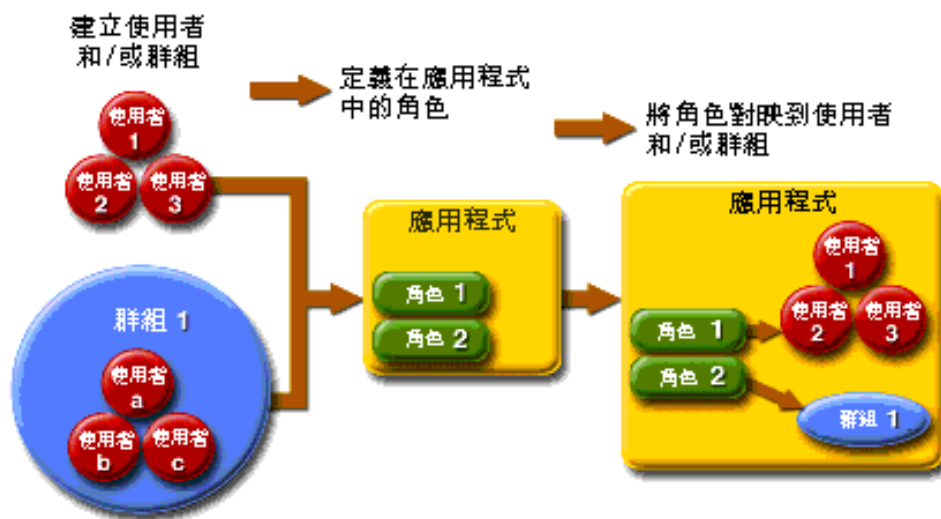


圖 9-1 角色對映

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地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 (也稱為**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`、`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為 `keyfile` 的檔案中。您可以使用 Administration Console 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 92 頁的「憑證和 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為 `admin-keyfile` 的檔案中。您可以使用 Administration Console 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 `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 Admin Console 線上說明。

憑證和 SSL 簡介

本節論述以下主題：

- [第 92 頁的「關於數位憑證」](#)
- [第 93 頁的「關於安全套接字層」](#)

關於數位憑證

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。憑證還可以使兩個實體之間能夠進行安全、機密的通訊。

憑證有很多種類型，例如個人憑證 (由個人使用) 和伺服器憑證 (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 93 頁的「關於安全套接字層」](#)。

憑證以**公開金鑰密碼學**為基礎，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。

憑證就好像一本護照：它可以識別持有者並提供其他重要資訊。憑證由稱為**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，以使他人無法偽造或竄改憑證。CA 對憑證進行簽署之後，持有者可以提供該憑證做為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，憑證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，憑證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。
- 發行憑證的 CA 的名稱。
- 過期日期。

數位憑證受 x.509 格式的技術規格約束。若要驗證 certificate 範圍中某個使用者的身份，認證服務將使用 X.509 憑證的一般名稱欄位做為主體名稱對 X.509 憑證進行驗證。

關於憑證鏈

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**自簽署憑證**。自簽署憑證是其發行者(簽署者)與主旨(其公開金鑰由該憑證進行認證的實體)相同的憑證。如果所有者向 CA 傳送憑證簽署請求(CSR)，然後輸入回應，自簽署憑證將被憑證鏈取代。鏈的底部是由 CA 發行的、用於認證主旨的公開金鑰憑證(回覆)。鏈中的下一個憑證是認證 CA 公開金鑰的憑證。通常，這是一個自簽署憑證(即，來自 CA、用於認證其自身公開金鑰的憑證)，並且是鏈中的最後一個憑證。

在其他情況下，CA 可能會傳回一個憑證鏈。在此情況下，鏈的底部憑證是相同的(由 CA 簽署的憑證，用於認證金鑰項目的公開金鑰)，但是鏈中的第二個憑證是由其他 CA 簽署的憑證，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，直至到達自簽署的**根憑證**。因此，鏈中的每個憑證(第一個憑證之後的憑證)都需要認證鏈中前一個憑證的簽署者的公開金鑰。

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 使用數位憑證來確保在伺服器和使用端之間進行安全、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(通常請求為以 https 開頭的 URL，而非 http 開頭的 URL)。
- 伺服器透過傳送其憑證(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 憑證是否已過期。
- 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。

交換之後，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 標準的最新版本稱為 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。只有安裝了數位憑證之後，大多數 Web 伺服器的 HTTPS 服務才能夠執行。請使用第 99 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、傳輸憑證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憑證)，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，因此也不能將多個憑證指定給單一 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憑證進行認證，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示的網域名稱 (如果有，且主要適用於官方 CA 簽署憑證) 進行對照。如果網域名稱不相符，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，則將防火牆配置為僅允許處理連接埠 8080 上的 HTTP 請求。同樣地，如果為連接埠 8181 設定了 HTTPS 請求，則必須將防火牆配置為允許處理連接埠 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 (全稱為 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，因為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為允許處理 HTTP 和 HTTPS 作業事件。您必須將內部防火牆配置為允許 HTTP 伺服器外掛程式與防火牆後面的 Application Server 進行通訊。

使用 Administration Console 管理安全性

Administration Console 提供了對安全性的以下方面進行管理的方法：

- [第 95 頁的「伺服器安全性設定」](#)
- [第 95 頁的「範圍和 file 範圍使用者」](#)
- [第 95 頁的「JACC 提供者」](#)
- [第 96 頁的「稽核模組」](#)
- [第 96 頁的「訊息安全性」](#)
- [第 96 頁的「HTTP 和 IIOP 偵聽程式安全性」](#)
- [第 96 頁的「管理服務安全性」](#)
- [第 97 頁的「安全性對映」](#)

伺服器安全性設定

在 [安全性設定] 頁面中，設定整個伺服器的特性，包括指定預設範圍、匿名角色和預設的主體使用者名稱和密碼。

範圍和 file 範圍使用者

[第 89 頁的「瞭解使用者、群組、角色和範圍」](#) 中介紹了範圍的概念。

- 建立新範圍
- 刪除現有範圍
- 修改現有範圍的配置
- 增加、修改和刪除 file 範圍中的使用者
- 設定預設範圍

JACC 提供者

[第 88 頁的「指定 JACC 提供者」](#) 中介紹了 JACC 提供者。使用 Administration Console 執行以下作業：

- 增加新的 JACC 提供者
- 刪除或修改現有 JACC 提供者

稽核模組

第 89 頁的「稽核認證與授權決策」中介紹了稽核模組。稽核是記錄重要事件 (例如錯誤或安全性漏洞) 以進行後續檢查的方法。所有認證事件都被記錄到 Application Server 記錄中。完整的存取記錄提供了 Application Server 存取事件的循序追蹤。

使用 Administration Console 執行以下作業：

- 增加新的稽核模組
- 刪除或修改現有稽核模組

訊息安全性

第 89 頁的「配置訊息安全性」中介紹了訊息安全性的概念。使用 Administration Console 執行以下作業：

- 啓用訊息安全性
- 配置訊息安全性提供者
- 刪除或配置現有訊息安全性配置或提供者

有關這些作業的詳細資訊，請參閱 Administration Console 線上說明。

HTTP 和 IIOP 偵聽程式安全性

HTTP 服務中的每個虛擬伺服器都透過一個或多個 **HTTP 偵聽程式** 提供網路連線。

Application Server 支援 CORBA (共用物件請求代理程式架構) 物件，這類物件使用網際網路 Orb 交換協定 (IIOP) 在網路上進行通訊。**IIOP 偵聽程式** 接受來自 EJB 的遠端用戶端和其他基於 CORBA 之用戶端的內送連線。如需有關 IIOP 偵聽程式的一般資訊，請參閱第 134 頁的「IIOP 偵聽程式」。

使用 Administration Console 執行以下作業：

- 建立新的 HTTP 或 IIOP 偵聽程式，並指定該偵聽程式所使用的安全性。
- 修改現有 HTTP 或 IIOP 偵聽程式的安全性設定。

管理服務安全性

管理服務決定伺服器實例是一般實例、網域管理伺服器 (DAS)，還是兼具兩者。使用管理服務可以配置 JSR-160 相容遠端 JMX 連接器，該連接器處理網域管理伺服器與遠端伺服器實例的節點代理程式 (管理主機電腦上的伺服器實例) 之間的通訊。

使用 Administration Console 執行以下作業：

- 管理管理服務

- 編輯 JMX 連接器
- 修改 JMX 連接器的安全性設定

安全性對映

請使用 Administration Console 來執行下列安全性對映作業：

- 將安全性對映增加至現有連接器連線池中
- 刪除或配置現有安全性對映

使用憑證和 SSL

- [第 97 頁的「關於憑證檔案」](#)
- [第 98 頁的「使用 Java 安全套接字延伸 \(JSSE\) 工具」](#)
- [第 101 頁的「使用網路安全服務 \(NSS\) 工具」](#)
- [第 104 頁的「將 Application Server 與硬體加密加速器搭配使用」](#)

關於憑證檔案

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 (網路安全性服務) 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 102 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為 *slas* 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。受信任的憑證通常包括那些憑證授權單位 (CA) 的憑證。

在 Platform Edition 的伺服器端，Application Server 使用 JSSE 格式，該格式使用 `keytool` 來管理憑證和金鑰庫。在 Enterprise Edition 的伺服器端，Application Server 使用 NSS 格式，該格式使用 `certutil` 來管理儲存私密金鑰和憑證的 NSS 資料庫。在兩種版本中，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 配置為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 Admin Console，展開 [server-config] 節點 > [JVM 設定] > [JVM 選項] 標籤，以增加或修改憑證檔案新位置的值欄位。

```
-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 Platform Edition 和 Enterprise Edition 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 (以憑證形式)。

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 金鑰演算法在類型為 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 99 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為 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 100 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。

- 顯示類型為 JKS 的金鑰庫中的可用憑證。

```
keytool -list -v -keystore ${keystore.file} -storepass ${keystore.pass}
```

- 顯示類型為 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 (也稱為 *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}
```

- 從類型為 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}
```

- 從類型為 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}
```

- 從類型為 JKS 的金鑰庫中刪除憑證。

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

[第 101 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，該目錄為 *domain-dir/config*。如需有關變更這些檔案位置的資訊，請參閱 [第 97 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為 *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*。

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。

- 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
```

- 如需由憑證授權單位簽署的憑證，請參閱第 100 頁的「使用 `keytool` 公用程式簽署數位憑證」。
- 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
```

- 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`。
- 工具將顯示有關憑證的資訊並提示您是否要信任該憑證。
- 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]
```

- 重新啟動 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偽造。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。

- 請依照 CA 網站上的說明產生憑證金鑰對。
- 下載產生的憑證金鑰對。
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，該目錄為 `domain-dir/config`。請參閱第 97 頁的「變更憑證檔案的位置」。
- 在 `shell` 中，變更至包含憑證的目錄。
- 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. 重新啟動 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
```

使用網路安全服務 (NSS) 工具

在 Enterprise Edition 的伺服器端，使用網路安全服務 (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 98 頁的「使用 Java 安全套接字延伸 (JSSE) 工具」中所述的 JSSE 格式。

使用網路安全服務 (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 102 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 103 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 104 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 `install-dir/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 = ${install-dir}/lib`
- `${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，為方便起見，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，而不必使用其真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 102 頁的「使用 `certutil` 公用程式」
- 第 103 頁的「使用 `pk12util` 公用程式匯入和匯出憑證」
- 第 104 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 and 用戶端憑證。在此範例中，CN 必須為 `hostname.domain.[com|org|net|...]` 格式。
在此範例中，目錄為 `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為 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 為公開金鑰加密標準 (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 憑證轉換為 JKS 格式 (需要 Java 來源)：

```
<target name="convert-pkcs12-to-jks" depends="init-common">
  <delete file="{jks.file}" failonerror="false"/>
  <java classname="com.sun.enterprise.security.KeyTool">
    <arg line="-pkcs12"/>
    <arg line="-pkcsFile ${pkcs12.file}"/>
    <arg line="-pkcsKeyStorePass ${pkcs12.pass}"/>
    <arg line="-pkcsKeyPass ${pkcs12.pass}"/>
    <arg line="-jksFile ${jks.file}"/>
    <arg line="-jksKeyStorePass ${jks.pass}"/>
  </classpath>
    <path element path="{slas.classpath}"/>
    <path element path="{env.JAVA_HOME}/jre/lib/jsse.jar"/>
  </classpath>
</java>
</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、啟用或停用槽、啟用或停用 FIPS-140-1 規範遵循，以及為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，為一般使用者提供行動式安全金鑰儲存。

Sun Java System Application Server 8.1 和 8.2 Standard Edition 或 Enterprise Edition 在 Java 2 Platform, Standard Edition (J2SE 平台) 5.0 上執行時，支援對 SSL 或 TLS 通訊使用 PKCS#11 記號，並支援使用網路安全性服務 (NSS) 工具來管理金鑰和 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：

- [第 105 頁的「關於配置硬體加密加速器」](#)
- [第 106 頁的「配置 PKCS#11 記號」](#)
- [第 107 頁的「管理金鑰和憑證」](#)
- [第 108 頁的「配置 J2SE 5.0 PKCS#11 提供者」](#)

關於配置硬體加密加速器

Sun Java System Application Server 8.1 和 8.2 Standard Edition 或 Enterprise Edition 已通過 Sun Crypto Accelerator 1000 (SCA-1000) 和 SCA-4000 的測試。

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，通常稱為 NSS 軟體記號) 以及 NSS 命令行管理工具。如需更多詳細資訊，請參閱[第 101 頁的「使用網路安全服務 \(NSS\) 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 提供者為加密服務的提供者，可做為原生 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。為簡化說明，本小節所描述的程序僅使用 UNIX 指令。必要時請將 UNIX 變數替代為 Windows 變數。

硬體加密加速器的配置可分為兩個主要程序：

- [第 106 頁的「配置 PKCS#11 記號」](#)
- [第 108 頁的「配置 J2SE 5.0 PKCS#11 提供者」](#)

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為 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 108 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為 `mytoken`。

為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 101 頁的「使用網路安全服務 (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。如需有關使用 `keytool` 的詳細資訊，請參閱「Java PKCS#11 Reference Guide」，網址為

<http://java.sun.com/j2se/1.5.0/docs/guide/secuirty/pllguide.html>。

本小節說明下列主題：

- 第 107 頁的「列出金鑰和憑證」
- 第 108 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
slas	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 101 頁的「使用網路安全服務 (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 會為 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，會為每個 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為唯一。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為 *install-dir/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：

```
<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. 重新啟動 Application Server。
- 自訂配置將在重新啟動後生效。

詳細資訊

- 您可透過以下 URL 檢視 Java 2 Standard Edition 的安全性論述：<http://java.sun.com/j2se/1.5.0/docs/guide/security/index.html>。
- 您可透過以下 URL 檢視「J2EE 1.4 Tutorial」中的「Security」一章：<http://java.sun.com/j2ee/1.4/docs/tutorial/doc/index.html>。
- 「Administration Guide」中的第 10 章。
- 「Developer's Guide」中的「Securing Applications」一章。

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。若要瞭解有關這些概念的更多資訊，請在開始閱讀本章之前先仔細閱讀第 110 頁的「詳細資訊」中列出的資源。

本章說明如何在 Application Server 中為 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「J2EE 1.4 Tutorial」中的「Security」一章中論述) 不同，因為訊息安全性可用於將訊息保護從訊息傳輸中分離開，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) 為互通 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 憑證、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 112 頁的「系統管理員」](#)和 [第 112 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 113 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：

- [第 112 頁的「系統管理員」](#)
- [第 112 頁的「應用程式部署人員」](#)
- [第 113 頁的「應用程式開發人員」](#)

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 Administration Console 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 116 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應用程式特定的訊息保護策略，則 (在應用程式組譯時) 指定這些策略。
- 修改 Sun 特定的部署描述元，以向 Web 服務端點和服務參照指定應用程式特定的訊息保護策略資訊 (即 message-security-binding 元素)。

「Developers' Guide」中的「Securing Applications」一章將論述這些安全性作業。如需有關指向該章的連結，請參閱第 110 頁的「詳細資訊」。

應用程式開發人員

應用程式開發者可以啟用訊息安全性，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應用程式特定的訊息保護策略。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 規格為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為使用者 (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 用戶端憑證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 97 頁的「關於憑證檔案」。

關於加密

加密的目的是將資料修改為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 118 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。應用程式特定的訊息保護策略(細緻程度為 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 Sun 特定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 116 頁的「[配置 Application Server 以實現訊息安全性](#)」中也論述了這些概念。

- **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為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為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**用於將其容器建立為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為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**有時被稱為**預設提供者**。

- 預設用戶端提供者

預設用戶端提供者用於為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 121 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 116 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 120 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。

配置應用程式特定的 Web 服務安全性

透過在應用程式之 Sun 特定的部署描述元中定義 `message-security-binding` 元素，來配置應用程式特定的 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。[第 110 頁的「詳細資訊」](#) 包含指向該章的連結。

確保範例應用程式的安全

Application Server 隨附名為 `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 116 頁的「配置應用程式特定的 Web 服務安全性」](#) 中所述)。

`xms` 範例應用程式安裝在以下目錄中：`install-dir/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

- [第 116 頁的「請求策略配置和回應策略配置的動作」](#)
- [第 117 頁的「配置其他安全性功能」](#)
- [第 118 頁的「配置 JCE 提供者」](#)

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署。此訊息包含 wsse:Security 標頭，此標頭包含以 ds:Signature。
auth-source="sender" auth-recipient="before-content" OR 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" 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 已簽署。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" auth-recipient="after-content"	SOAP 訊息內文的內容已簽署和加密，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" OR 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 118 頁的「配置 JCE 提供者」中論述了如何配置 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，並為該範圍配置適當的使用者資料庫。

4. 管理憑證和私密金鑰 (如有必要)。

完成之後

如果已將 Application Server 的功能配置為供訊息安全性提供者使用，則可能會啟用隨 Application Server 一起安裝的提供者，如第 120 頁的「啟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 加密擴展 (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 提供者增加為 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/jce14_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啟動，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* 設定為 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，其值為 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.rsa.jca.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. 重新啟動 Application Server。

訊息安全性設定

為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 Administration Console、`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，因為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 Administration Console 來配置 Application Server，然後選擇使用 `asadmin` 工具指令。僅在無 Administration Console 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。以下小節提供用於啟用、建立、編輯和刪除訊息安全性配置和提供者的詳細資訊。

- 第 120 頁的「啟用訊息安全性的提供者」
- 第 120 頁的「配置訊息安全性提供者」
- 第 121 頁的「建立訊息安全性提供者」
- 第 121 頁的「啟用應用程式用戶端的訊息安全性」
- 第 121 頁的「設定應用程式用戶端配置的請求策略和回應策略」
- 第 122 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啟動 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 121 頁的「啓用應用程式用戶端的訊息安全性」](#)中論述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 Admin Console 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 Admin Console 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符。

若要設定應用程式用戶端配置的請求策略，請依循第 121 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="install-dir/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 116 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，請將該元素保留為空白，例如：

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

詳細資訊

- 您可透過以下 URL 檢視 Java 2 Standard Edition 的安全性論述：<http://java.sun.com/j2se/1.4.2/docs/guide/security/index.html>。
- 您可透過以下 URL 檢視「J2EE 1.4 Tutorial」中的「Security」一章：<http://java.sun.com/j2ee/1.4/docs/tutorial/doc/index.html>。
- 「Administration Guide」中的第 9 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/xmlsig-core/>。
- 您可透過以下 URL 檢視「XML Encryption Syntax and Processing」文件：<http://www.w3.org/TR/xmlenc-core/>。

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：

- 第 125 頁的「何為作業事件？」
- 第 126 頁的「J2EE 技術中的作業事件」
- 第 126 頁的「恢復作業事件」
- 第 127 頁的「作業事件逾時值」
- 第 127 頁的「作業事件記錄」
- 第 127 頁的「關鍵點間隔」

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。該事件稱為**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。

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。資源管理員參與分散式作業事件，方法為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。
- 資源介面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。資源介面通常專用於資源管理員。它可以做為程式庫，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。

恢復作業事件

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 可在伺服器啟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，則伺服器重新啟動作業可能被延遲，因為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，啟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 [啟發式決策] 欄位來確定結果。

可以使用 Admin Console 配置 Application Server，以回復作業事件。有關執行這項作業的詳細程序，請參閱 Admin Console 線上說明。

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。在為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。有關執行這項作業的詳細步驟，請參閱 Admin Console 線上說明。

作業事件記錄

為了保持被呼叫資源的資料完整性，同時為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為讀取。

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為頻繁，效能會略微降低。

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：

- 第 129 頁的「虛擬伺服器」
- 第 130 頁的「HTTP 偵聽程式」

虛擬伺服器

虛擬伺服器 (有時稱為虛擬主機) 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` 均可解析為實體伺服器的 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：

- 名為 `server` 的虛擬伺服器，用於託管所有使用者定義的 Web 模組
- 名為 `_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 Administration Console)。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，其他虛擬伺服器可為使用者和用戶提供託管功能，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、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和 IP 位址組合。例如，透過將 IP 位址指定為 `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 偵聽程式可以為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 位址和連接埠號的組合，因此您可以擁有多個 IP 位址相同但連接埠號不同 (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 (如果已將機器配置為可以回應這些位址)。

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，其中每個連接埠號用於不同目的。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 啟動時，它具有以下 HTTP 偵聽程式：

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

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 期間指定為 HTTP 伺服器連接埠號的連接埠號。如果 Application Server 使用預設連接埠號值，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 4849。

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 (稱為連線佇列) 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。連線佇列除接收器執行緒剛剛接受的新連線外，也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為已關閉 (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為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，則該名稱應為此別名。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，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，還應指定是否為 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/`) 和為此 Web 應用程式指定的環境根目錄。若要存取 Administration Console，請使用 URL `https://localhost:4849/` 或 `https://localhost:4849/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。

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：

- 第 133 頁的「CORBA」
- 第 133 頁的「什麼是 ORB？」
- 第 134 頁的「IIOP 偵聽程式」
- 第 134 頁的「使用 ORB」

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為基礎，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 (稱為物件參考) 和參數 (如果有)。CORBA 自動處理物件註冊、物件位置、物件啟動、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 (縮寫為 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插槽，它接受來自企業 Bean 的遠端用戶端和其他基於 CORBA 的用戶端的進來的連線。可以為 Application Server 配置多個 IIOP 偵聽程式。為每個偵聽程式指定一個連接埠號、一個網路位址和 (選擇性地) 多個安全性屬性。

使用 ORB

若要建立 IIOP 偵聽程式，請選取 Admin Console 中的 [配置] > [ORB] > [IIOP 偵聽程式]，並按一下 [新增]。或者，您也可以使用以下的 `asadmin` 指令來建立 IIOP 偵聽程式：`create-iiop-listener(1)` and `create-ssl(1)`。

若要編輯 IIOP 偵聽程式，請在 Admin Console 中選取 [配置] > [ORB] > [IIOP 偵聽程式]，然後選取您要修改的偵聽程式。修改設定。若您變更了連接埠號，請重新啟動伺服器。ORB 使用執行緒池回應來自通過 RMI-IIOP 進行通訊的企業 Bean 的遠端用戶端和其他用戶端的請求。

若要刪除 IIOP 偵聽程式，請在 Admin Console 中選取 [配置] > [ORB] > [IIOP 偵聽程式]，然後選取您要刪除的偵聽程式。或者，您也可以使用 `delete-iiop-listener(1)` 指令。

`ORBCommunicationsRetryTimeout` 特性會指定 ORB 用戶端嘗試與無法連線之 ORB 後端建立連線的秒數。預設值是 60 秒。如果使用這個預設值，當無法連線 ORB 後端時，您會看到記錄中出現大量的 CORBA 異常，以及很高的網路使用量。

在這種情況下，請將 `ORBCommunicationsRetryTimeout` 設定為較低的值。

協力廠商的 ORB

Sun Java System Application Server 可以和協力廠商的 ORB 軟體一起使用。為支援上述協力廠商的 ORB，您必須修訂伺服器端的設定。

若您要在 Application Server 中實作對協力廠商 ORB 的支援，必須編輯 `domain.xml` 和 `server.policy` 檔案。如需有關配置範例協力廠商 ORB 之方式的詳細說明，請參閱 [Configuring Sun Java System Application Server for Third-Party ORBs](http://developers.sun.com/prodtech/appserver/reference/techart/orb.html) (<http://developers.sun.com/prodtech/appserver/reference/techart/orb.html>)

執行緒池

Java 虛擬機器 (JVM) 可以支援一次執行多個執行緒。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閒執行緒。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 中所有的執行緒時，出現執行緒不足的情況。

本章包含以下主題：

- [第 135 頁的「配置執行緒池」](#)

配置執行緒池

若要使用 Administration Console 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 (以秒為單位)，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，輸入由此執行緒池處理的工作佇列總數。
- 重新啟動 Application Server。

如需有關建立執行緒池的詳細資訊，請按一下 Administration Console 的「說明」。

還可以使用 `asadmin` 指令 `create-threadpool(1)` 從指令行建立執行緒池。

若要使用 Administration Console 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，儲存並重新啟動 Application Server。

如需有關編輯執行緒池的詳細資訊，請按一下 Administration Console 的「說明」。

若要使用 Administration Console 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]。

重新啟動 Application Server。

還可以使用 `asadmin` 指令 `delete-threadpool(1)` 從指令行刪除執行緒池。

配置記錄

本章簡要說明如何配置記錄並檢視伺服器記錄。本章包含以下小節：

- 第 137 頁的「記錄檔的記錄」
- 第 138 頁的「設定自訂記錄層級」
- 第 139 頁的「記錄程式名稱空間階層結構」

記錄檔的記錄

Application Server 使用 JSR 047 中指定的 Java 2 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* 中。

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，請查閱 Administration Console 線上說明。

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：

```
[#|2004-10-21T13:25:53.852-0400|INFO|sun-appserver-e8.1|javax.enterprise.  
system.core|_ThreadID=13;|CORE5004: Resource Deployed:  
[cr:jms/DurableConnectionFactory].|#]
```

在本範例中，

- [# 和 #] 標示該記錄的開頭和結尾。

- 垂直線 (|) 用於分隔記錄欄位。
- 2004-10-21T13:25:53.852-0400 指定了日期和時間。
- Log Level 為 INFO。此層級可以是以下任何值：SEVERE、WARNING、INFO、CONFIG、FINE、FINER 和 FINEST。
- ProductName-Version 為 sun-appserver-ee8.1。
- 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，在此例中為 javax.enterprise.system.core。
- Key Value Pairs 為鍵名和鍵值，通常為執行緒 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 (在此範例中為 CORE5004)。

以後的版本中，可能會變更或增強記錄檔的記錄格式。

設定自訂記錄層級

本小節說明如何為使用 `java.util.logging` 套裝軟體的應用程式配置自訂記錄層級，並存取 Application Server 的記錄子系統。

`java.util.logging` 套裝軟體提供階層式名稱空間，以供建立記錄程式實例。特定的記錄記錄是否要輸出至 Application Server 實例的記錄檔案，要看 [記錄記錄] 的記錄層級+H248，以及所指定的記錄層級而定。

Application Server 記錄程式設定配置提供超過二十種的記錄模組，允許對 Application Server 的內部記錄進行精密控制。另外還有一個可用來建立其他自訂記錄模組的選項，只要指定模組所要使用的模組名稱和記錄層級即可。

這裡的重點是，記錄程式為靜態名稱，且未提供繼承性。因此，若自訂記錄程式是以 `com.someorg.app` 名稱進行配置，而應用程式嘗試查找 `com.someorg.app.submodule` 記錄程式，則不提供繼承 `com.someorg.app` 設定的記錄程式。相反，`com.someorg.app.submodule` 會有預設的記錄程式，且其記錄層級會設定為 INFO 層級或更高。

若應用程式需要使用記錄程式繼承性，只要編輯用來執行 Application Server 之 Java 執行階段的 `logging.properties` 檔案，即可進行配置。例如，若將下列項目增加到 `logging.properties` 檔案，則會導致 `com.someorg.app.submodule` 在建立時即繼承相同的 FINE 層級：

```
com.someorg.app.level = FINE
```

如需有關 Java 記錄 API 的更多詳細資訊，請參閱 Java 文件，網址是 <http://java.sun.com/j2se/1.5.0/docs/api/java/util/logging/package-summary.html>，以及其他 `java.util.logging` 類別。

記錄程式名稱空間階層結構

Application Server 為它的每個模組都提供了記錄程式。下表按照每個記錄程式的模組名稱和名稱空間之字母順序列出，如 Administration Console 的 [記錄層級] 頁面中所示。[記錄層級] 頁面中未顯示表中最後三個模組。

表 15-1 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
Classloader	javax.enterprise.system.core.classloading
CMP	javax.enterprise.system.container.cmp
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
JavaMail	javax.enterprise.resource.javamail
JAXR	javax.enterprise.resource.webservices .registry
JAX-RPC	javax.enterprise.resource.webservices.rpc
JDO	javax.enterprise.resource.jdo
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
節點代理程式 (僅限於 Enterprise Edition)	javax.ee.enterprise.system.nodeagent
根	javax.enterprise
SAAJ	javax.enterprise.resource.webservices.saaaj
安全性	javax.enterprise.system.core.security
伺服器	javax.enterprise.system
同步 (僅限於 Enterprise Edition)	javax.ee.enterprise.system.tools .synchronization

表 15-1 Application Server 記錄程式名稱空間 (續)

模組名稱	名稱空間
Util	javax.enterprise.system.util
檢驗器	javax.enterprise.system.tools.verifier
Web 容器	javax.enterprise.system.container.web
核心	javax.enterprise.system.core
系統輸出 (System.out.println)	javax.enterprise.system.stream.out
系統錯誤 (System.err.println)	javax.enterprise.system.stream.err

監視元件和服務

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。

本章包括下列小節：

- 第 141 頁的「監視概述」
- 第 145 頁的「受監視的元件和服務的統計資訊」
- 第 163 頁的「啓用與停用監視」
- 第 164 頁的「檢視監視資料」
- 第 178 頁的「使用 JConsole」

監視概述

若要監視 Application Server，請執行以下步驟：

1. 使用 Administration Console 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 163 頁的「啓用與停用監視」。
2. 使用 Administration Console 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 164 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 142 頁的「應用程式樹」
- 第 143 頁的「HTTP 服務樹」
- 第 143 頁的「資源樹」
- 第 144 頁的「連接器服務樹」
- 第 144 頁的「JMS 服務樹」
- 第 144 頁的「ORB 樹」
- 第 145 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 145 頁的「EJB 容器統計資訊」。

範例 16-1 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 slsb/entity/mdb) *
|   |--- web-module-1
|   |   |--- virtual-server-1 *
|   |       |---servlet1 *
|   |       |---servlet2 *
|--- standalone-web-module-1
|   |--- virtual-server-2 *
|       |---servlet3 *
|       |---servlet4 *
```

範例 16-1 應用程式節點樹狀結構 (續)

```

|      |      |----- 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 150 頁的「[HTTP 服務統計資訊](#)」。

範例 16-2 HTTP 服務示意圖 (Platform Edition 版)

```

http-service
|--- virtual-server-1
|   |--- http-listener-1 *
|   |--- http-listener-2 *
|--- virtual-server-2
|   |--- http-listener-1 *
|   |--- http-listener-2 *

```

範例 16-3 HTTP 服務示意圖 (Enterprise Edition 版)

```

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 151 頁的「[JDBC 連線池統計資訊](#)」。

範例 16-4 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 152 頁的「JMS/連接器服務統計資訊」。

範例 16-5 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 16-6 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 153 頁的「ORB 中連線管理程式的統計資訊」。

範例 16-7 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 153 頁的「執行緒池統計資訊」。

範例 16-8 執行緒池示意圖

```
thread-pools
|   |--- thread-pool-1 *
|   |--- thread-pool-2 *
```

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 145 頁的「EJB 容器統計資訊」
- 第 149 頁的「Web 容器統計資訊」
- 第 150 頁的「HTTP 服務統計資訊」
- 第 151 頁的「JDBC 連線池統計資訊」
- 第 152 頁的「JMS/連接器服務統計資訊」
- 第 153 頁的「ORB 中連線管理程式的統計資訊」
- 第 153 頁的「執行緒池統計資訊」
- 第 154 頁的「作業事件服務統計資訊」
- 第 154 頁的「Java 虛擬機器 (JVM) 統計資訊」
- 第 154 頁的「J2SE 5.0 中的 JVM 統計資訊」
- 第 158 頁的「生產 Web 容器 (PWC) 統計資訊」

EJB 容器統計資訊

下表說明 EJB 統計。

表 16-1 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	處於匯集狀態的實體 Bean 的數目。
readycount	範圍統計	處於就緒狀態的實體 Bean 的數目。
messagecount	計數統計資訊	訊息導引 Bean 收到的訊息數。

表 16-1 EJB 統計資訊 (續)

屬性名稱	資料類型	說明
methodreadycount	範圍統計	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	處於 Passive 狀態的有狀態階段作業 Bean 的數目。

下表中列出了可用於 EJB 方法呼叫的統計。

表 16-2 EJB 方法統計資訊

屬性名稱	資料類型	說明
methodstatistic	時間統計	作業被呼叫的次數；呼叫期間所花費的總時間等資訊。
totalnumerrors	計數統計資訊	執行方法導致異常的次數。如果為 EJB 容器啓用了監視功能，則此統計資訊是為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。如果為 EJB 容器啓用了監視功能，則此統計資訊是為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，則此統計資訊是為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。

表 16-3 EJB 階段作業儲存統計

屬性名稱	資料類型	說明
currentSize	範圍統計	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

表 16-3 EJB 階段作業儲存統計 (續)

屬性名稱	資料類型	說明
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，則此統計資訊是為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 (取消啟動) 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。
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 (僅限於 EE)	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 (僅限於 EE)	計數統計資訊	成功進行檢查點操作的階段作業數。
checkpointErrorCount (僅限於 EE)	計數統計資訊	無法進行檢查點操作的階段作業數。
checkpointedBeanSize (僅限於 EE)	值統計	由該儲存進行檢查點操作的 Bean 的總數。
checkpointTime (僅限於 EE)	時間統計	通過檢查點操作將 Bean 放入儲存中所花費的時間。

下表中列出了可用於 EJB 池的統計。

表 16-4 EJB 池統計資訊

屬性名稱	資料類型	說明
numbeansinpool	限制範圍統計	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	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。

表 16-4 EJB 池統計資訊 (續)

屬性名稱	資料類型	說明
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	針對要服務的訊息導引 Bean 而一次載入 JMS 階段作業的最大訊息數。預設值為 1。僅套用於訊息導引 Bean 的池。

下表中列出了可用於 EJB 快取的統計。

表 16-5 EJB 快取統計

屬性名稱	資料類型	說明
cachemisses	限制範圍統計	使用者請求未在快取中找到 Bean 的次數。
cachehits	限制範圍統計	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	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。

下表中列出了可用於計時器的統計。

表 16-6 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 142 頁的「應用程式樹」所示。系統為每個單獨的 Web 應用程式都顯示了 Web 容器統計資訊。第 149 頁的「Web 容器統計資訊」中列出了可用於 Servlet 的 Web 容器的統計，第 149 頁的「Web 容器統計資訊」中列出了可用於 Web 模組的統計。

表 16-7 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。
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	到目前為止所處理的請求總數。

第 149 頁的「Web 容器統計資訊」中列出了可用於 Web 模組的統計。

表 16-8 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	已為 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 (僅限於 EE)	平均範圍統計	Web 模組的階段作業大小。值可以為大、小或平均值，或以位元組為單位 (用於序列化階段作業)。

表 16-8 Web 容器 (Web 模組) 統計資訊 (續)

統計資訊	資料類型	注釋
containerlatency (僅限於 EE)	平均範圍統計	整個延時請求中 Web 容器部分的延時。值可以是長、短或平均值。
sessionpersisttime (僅限於 EE)	平均範圍統計	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 為單位，短、長或平均值)。
cachedsessionscurrent (僅限於 EE)	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。
passivatedsessionscurrent (僅限於 EE)	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

第 150 頁的「[HTTP 服務統計資訊](#)」中列出了可用於 HTTP 服務的統計。這些統計資訊僅適用於 Platform Edition。如需有關 Enterprise Edition 上的 HTTP 服務的統計，請參閱第 158 頁的「[生產 Web 容器 \(PWC\) 統計資訊](#)」。

表 16-9 HTTP 服務統計資訊 (僅適用於 Platform Edition)

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。

表 16-9 HTTP 服務統計資訊 (僅適用於 Platform Edition) (續)

統計資訊	單位	資料類型	注釋
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	到目前為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。

下表中列出了可用於 JDBC 連線池的統計。

表 16-10 JDBC 連線池統計

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為止在連線池中驗證失敗的連線總數。
numconnused	數目	範圍統計	提供連線使用統計資訊。目前正在使用的連線總數，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	上次取樣時池中的自由連線總數。
numconnexpired	數目	限制範圍統計	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	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。

表 16-10 JDBC 連線池統計 (續)

統計資訊	單位	資料類型	說明
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

第 152 頁的「JMS/連接器服務統計資訊」中列出了可用於連接器連線池的統計。第 152 頁的「JMS/連接器服務統計資訊」中列出了連接器工作管理的統計。

表 16-11 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為止在連線池中驗證失敗的連線總數。
numconnused	數目	範圍統計	提供連線使用統計資訊。目前正在使用的連線總數，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	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。
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	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。

第 152 頁的「JMS/連接器服務統計資訊」中列出了可用於連接器工作管理的統計。

表 16-12 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	由連接器執行的工作物件數。
waitqueuelength	範圍統計	執行前在佇列中等待的工作物件數。
workrequestwaittime	範圍統計	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

第 153 頁的「ORB 中連線管理程式的統計資訊」中列出了可用於 ORB 中連線管理員的統計。

表 16-13 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	與 ORB 的連線總數。

執行緒池統計資訊

下表中列出了可用於執行緒池的統計。

表 16-14 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	在得到處理之前請求在佇列中等待的平均時間 (以毫秒為單位)。
averageworkcompletion-time	毫秒	範圍統計	完成指定所花費的平均時間 (以毫秒為單位)。
currentnumberofthreads	數目	限制範圍統計	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。

表 16-14 執行緒池統計資訊 (續)

統計資訊	單位	資料類型	說明
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	到目前為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。下表中列出了可用於作業事件服務的統計。

表 16-15 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結作業事件服務後轉返。
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。下表中列出了可用於 JVM 的統計。

表 16-16 JVM 統計資訊

統計資訊	資料類型	說明
heapsize	限制範圍統計	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。

J2SE 5.0 中的 JVM 統計資訊

如果將 Application Server 配置為在 J2SE 5.0 版或更高版本上執行，則可以從 JVM 中取得附加監視資訊。將監視層級設定為 [低] 以啓用這些附加資訊的顯示。將監視層級設定為 [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 J2SE 5.0 中可用的附加監視功能的更多資訊，請參閱以下位置中標題為「*Monitoring and Management for the Java Platform*」的文件：<http://java.sun.com/j2se/1.5.0/docs/guide/management/>。

以下位置中論述了 J2SE 5.0 監視工

具：<http://java.sun.com/j2se/1.5.0/docs/tooldocs/#manage>。

第 154 頁的「J2SE 5.0 中的 JVM 統計資訊」中列出了可用於在 J2SE 5.0 的 JVM 中進行類別載入的統計。

表 16-17 J2SE 5.0 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。

第 154 頁的「J2SE 5.0 中的 JVM 統計資訊」中列出了可用於在 J2SE 5.0 的 JVM 中進行編譯的統計。

表 16-18 J2SE 5.0 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 (以毫秒為單位)。

第 154 頁的「J2SE 5.0 中的 JVM 統計資訊」中列出了可用於在 J2SE 5.0 的 JVM 中進行資源回收的統計。

表 16-19 J2SE 5.0 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 (以毫秒為單位)。

第 154 頁的「J2SE 5.0 中的 JVM 統計資訊」中列出了可用於 J2SE 5.0 的 JVM 中的記憶體

表 16-20 J2SE 5.0 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。

表 16-20 J2SE 5.0 的 JVM 統計資訊 - 記憶體 (續)

統計資訊	資料類型	說明
maxheapsize	計數統計資訊	可用於記憶體管理的最大記憶體容量 (以位元組為單位)。
committedheapsize	計數統計資訊	確定可由 JVM 使用的記憶體容量 (以位元組為單位)。
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 (以位元組為單位)。
committednonheapsize	計數統計資訊	確定可由 JVM 使用的記憶體容量 (以位元組為單位)。

第 154 頁的「J2SE 5.0 中的 JVM 統計資訊」中列出了可用於 J2SE 5.0 的 JVM 中的作業系統的統計。

表 16-21 J2SE 5.0 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。

第 154 頁的「J2SE 5.0 中的 JVM 統計資訊」中顯示了可用於 J2SE 5.0 的 JVM 中的執行階段的統計。

表 16-22 J2SE 5.0 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。

表 16-22 J2SE 5.0 的 JVM 統計資訊 - 執行階段 (續)

統計資訊	資料類型	說明
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	啟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 (以毫秒為單位)。

第 154 頁的「J2SE 5.0 中的 JVM 統計資訊」中列出了可用於 J2SE 5.0 的 JVM 中的 ThreadInfo 的統計。

表 16-23 J2SE 5.0 的 JVM 統計 — 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 (以毫秒為單位) 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 (以毫秒為單位) 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段化。
stacktrace	字串統計資訊	與該執行緒相關聯的堆疊追蹤。

第 154 頁的「J2SE 5.0 中的 JVM 統計資訊」中列出了可用於 J2SE 5.0 的 JVM 中的執行緒的統計。

表 16-24 J2SE 5.0 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 啟動或峰重設以來的峰活性執行緒計數。
totalstartedthreadcount	計數統計資訊	JVM 啟動以來建立和/或啟動的執行緒總數。
daemonthreadcount	計數統計資訊	目前的活性常駐程式執行緒數。
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 (以納秒為單位)。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

可用於 Application Server Enterprise Edition (EE) 上的以下 PWC 元件和服務的統計：

- [第 158 頁的「生產 Web 容器 \(PWC\) 統計資訊」](#)，PWC 虛擬伺服器
- [第 158 頁的「生產 Web 容器 \(PWC\) 統計資訊」](#)，PWC 請求
- [第 158 頁的「生產 Web 容器 \(PWC\) 統計資訊」](#)，PWC 檔案快取
- [第 158 頁的「生產 Web 容器 \(PWC\) 統計資訊」](#)，PWC 持續作用
- [第 158 頁的「生產 Web 容器 \(PWC\) 統計資訊」](#)，PWC DNS
- [第 158 頁的「生產 Web 容器 \(PWC\) 統計資訊」](#)，PWC 執行緒池
- [第 158 頁的「生產 Web 容器 \(PWC\) 統計資訊」](#)，PWC 連線佇列
- [第 158 頁的「生產 Web 容器 \(PWC\) 統計資訊」](#)，PWC HTTP 服務

[第 158 頁的「生產 Web 容器 \(PWC\) 統計資訊」](#) 中列出了 PWC 虛擬伺服器的統計。

表 16-25 PWC 虛擬伺服器統計資訊 (僅限於 EE)

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 <code>unknown</code> 或 <code>active</code> 。
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。

下表中列出了可用於 PWC 請求的統計。

表 16-26 PWC 請求統計 (僅限於 EE)

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，則值為 0。
countbytesreceived	計數統計資訊	接收的位元組數，如果此資訊不可用，則值為 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，則值為 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，則值為 0。
countopenconnections	計數統計資訊	目前開啓的連線數，如果此資訊不可用，則值為 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，則值為 0。
count2xx	計數統計資訊	代碼為 2XX 的回應的總數。
count3xx	計數統計資訊	代碼為 3XX 的回應的總數。
count4xx	計數統計資訊	代碼為 4XX 的回應的總數。
count5xx	計數統計資訊	代碼為 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	代碼為 200 的回應的總數。
count302	計數統計資訊	代碼為 302 的回應的總數。
count304	計數統計資訊	代碼為 304 的回應的總數。
count400	計數統計資訊	代碼為 400 的回應的總數。
count401	計數統計資訊	代碼為 401 的回應的總數。
count403	計數統計資訊	代碼為 403 的回應的總數。
count404	計數統計資訊	代碼為 404 的回應的總數。
count503	計數統計資訊	代碼為 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。

表 16-27 PWC 檔案快取統計資訊 (僅限於 EE)

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。
sizemapcache	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。
maxmapcachesize	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。下表中列出了可用於 PWC 持續作用的統計。

表 16-28 PWC 持續作用統計 (僅限於 EE)

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。
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。

表 16-28 PWC 持續作用統計 (僅限於 EE) (續)

屬性名稱	資料類型	說明
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 (以秒為單位)。

DNS 快取快取 IP 位址和 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。下表中列出了可用於 PWC DNS 的統計。

表 16-29 PWC DNS 統計 (僅限於 EE)

屬性名稱	資料類型	說明
flagcacheenabled	計數統計資訊	指出 DNS 快取是否被啟用 (開啟)。禁用時為 0，啟用時為 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	指示是否啟用了非同步 DNS 查找。禁用時為 0，啟用時為 1。
countasyncnamelookups	計數統計資訊	非同步 DNS 名稱查找的總數。
countasynccaddrlookups	計數統計資訊	非同步 DNS 位址查找的總數。
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。

表 16-30 PWC 執行緒池統計資訊 (僅限於 EE)

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。

表 16-30 PWC 執行緒池統計資訊 (僅限於 EE) (續)

屬性名稱	資料類型	說明
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。

表 16-31 PWC 連線佇列統計資訊 (僅限於 EE)

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。
ticktotalqueued	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。

表 16-32 PWC HTTP 服務統計資訊 (僅限於 EE)

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。
flagvirtualserveroverload	計數統計資訊	表示是否配置了多於 maxvirtualservers 的虛擬伺服器。如果設定不會爲所有的虛擬伺服器追蹤統計資料。
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，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，結果爲 0。

啓用與停用監視

您可以在 Admin Console 中選取 [配置] 節點 > [伺服器實例] 節點 > [監視] 頁面並選擇監視層級正在變更之服務的值，以配置監視層級。依預設，所有元件和服務的監視功能均處於關閉狀態。

如需有關配置監視層級的詳細步驟，請參閱 Admin Console 線上說明。

從指令行使用 `asadmin set` 開啓對各種 Application Server 元件的監視。例如，執行以下指令以開啓對 HTTP 服務的監視：

```
asadmin> set --user admin-user
server.monitoring-service.module-monitoring-levels.http-service=HIGH
```

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 = HIGH
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

```

使用 `set` 指令來關閉監視功能。

例如，執行以下指令以停用對 HTTP 服務的監視：

```

asadmin> set --user admin-user
server.monitoring-service.module-monitoring-levels.http-service=OFF

```

同樣地，若要停用監視其他元件的功能，請使用 `set` 指令並指定 `OFF` 為監視層級。

檢視監視資料

您可在 Admin Console 中檢視監視資料，方式是瀏覽至獨立伺服器實例的 [監視] 頁面，然後選取已部署在伺服器實例上並啓用了監視功能的元件或服務。[檢視] 欄位下顯示所選元件或服務的監視資料，並對可監視的特性進行了說明。

使用 Administration Console 監視遠端應用程式和實例。遠端實例必須正在執行並已設定配置才能執行此操作。如需有關檢視監視資料的詳細步驟，請參閱 Admin Console 線上說明。

若要使用指令行公用程式檢視監視資料，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
```

- 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，請使用 `asadmin get` 指令。

若要取得統計，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
```

瞭解和指定帶點名稱

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 (.) 字元做為分隔符來命名，因此這些子物件稱為**帶點名稱**。如果子節點為單一型，則僅需監視物件類型即可命名物件；否則，需要形式為 `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，並且為單一型。若要命名表示實例 `server` 之 `http-service` 的單一型子節點，則帶點名稱為：

```
server.http-service
```

另一個範例，`application` 為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
```

帶點名稱 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 172 頁的「[list 和 get 指令在所有層級上的預期輸出](#)」。

list 指令的範例

`list` 指令可針對指定伺服器實例名稱，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 172 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
```

還可以列示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
```

get 指令的範例

`get` 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 172 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 J2EE 應用程式中取得所有屬性：

```
asadmin> get --user admin-user --monitor server.applications.myJ2eeApp.*
```

傳回：

```
No matches resulted from the wildcard expression.
CLI137 Command get failed.
```

該 J2EE 應用程式層級上沒有提供可監視的屬性，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。

1. 啟動 `Application Server` 和 `asadmin` 工具。
2. 設定一些有用的環境變數，以避免爲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=4849
```

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 (為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。

```
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 occurred 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，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、帶點名稱 及相應的輸出。

表 16-33 頂層

指令	帶點名稱	輸出
list -m	server	server.applicationsserver.thread-poolserver.resourcesserver.http-servicesserver.transaction-servicesserver.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 16-34 應用程式層級

指令	帶點名稱	輸出
list -m	server.applications 或 *applications	appllapp2web-module1_warejb-module2_jar...

表 16-34 應用程式層級 (續)

指令	帶點名稱	輸出
list -m	server.applications.* 或 *applications.*	此節點下的子節點的階層結構。
get -m	server.applications.* 或 *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 16-35 應用程式 - 企業應用程式和獨立模組

指令	帶點名稱	輸出
list -m	server.applications.app1 或 *app1 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...
list -m	server.applications.app1.* 或 *app1.*	此節點下的子節點的階層結構。
get -m	server.applications.app1.* 或 *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 或 *ejb-module1_jar 或 server.applications.ejb-module1_jar	bean1bean2bean3...

表 16-35 應用程式 - 企業應用程式和獨立模組 (續)

指令	帶點名稱	輸出
list -m	server.applications.app1.ejb-module1_jar 或 *ejb-module1_jar 或 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.* 或 *ejb-module1_jar.* 或 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 備註：在獨立模組中，不會顯示包含應用程式名稱 (此範例中為 app1) 的節點。	子節點清單： bean-poolbean-cachebean-method
list -m	server.applications.app1.ejb-module1_jar.bean1 備註：在獨立模組中，不會顯示包含應用程式名稱 (此範例中為 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。
get -m	server.applications.app1.ejb-module1_jar.bean1.* 備註：在獨立模組中，不會顯示包含應用程式名稱 (此範例中為 app1) 的節點。	以下屬性及其關聯值： CreateCount_CountCreateCount_ DescriptionCreateCount_ LastSampleTimeCreateCount_ NameCreateCount_ StartTimeCreateCount_ UnitMethodReadyCount_ CurrentMethodReadyCount_ DescriptionMethodReadyCount_ HighWaterMarkMethodReadyCount_ LastSampleTimeMethodReadyCount_ LowWaterMarkMethodReadyCount_ NameMethodReadyCount_ StartTimeMethodReadyCount_ UnitRemoveCount_CountRemoveCount_ DescriptionRemoveCount_ LastSampleTimeRemoveCount_ NameRemoveCount_StartTimeAttribute RemoveCount_Unit

表 16-35 應用程式 - 企業應用程式和獨立模組 (續)

指令	帶點名稱	輸出
list -m	server.applications.appl.ejb-module1_jar.bean1.bean-pool 備註：在獨立模組中，不會顯示包含應用程式名稱 (此範例中為 appl) 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.appl.ejb-module1_jar.bean1.bean-pool.* 備註：在獨立模組中，不會顯示包含應用程式名稱 (此範例中為 appl) 的節點。	與 EJB 池屬性對應的屬性和值清單，如表 1-4 中所示。
list -m	server.applications.appl.ejb-module1_jar.bean1.bean-cache 備註：在獨立模組中，不會顯示包含應用程式名稱 (此範例中為 appl) 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.appl.ejb-module1_jar.bean1.bean-cache.* 備註：在獨立模組中，不會顯示包含應用程式名稱 (此範例中為 appl) 的節點。	與 EJB 快取記憶體屬性對應的屬性和值清單，如表 1-5 中所示。
list -m	server.applications.appl.ejb-module1_jar.bean1.bean-method.method1 備註：在獨立模組中，不會顯示包含應用程式名稱 (此範例中為 appl) 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.appl.ejb-module1_jar.bean1.bean-method.method1.* 備註：在獨立模組中，不會顯示包含應用程式名稱 (此範例中為 appl) 的節點。	與 EJB 方法屬性對應的屬性和值清單，如表 1-2 中所示。
list -m	server.applications.appl.web-module1_war	顯示指定給模組的虛擬伺服器。
get -m	server.applications.appl.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.appl.web-module1_war.virtual_server	顯示已註冊的 Servlet 清單。
get -m	server.applications.appl.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.appl.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.appl.web-module1_war.virtual_server.servlet1.*	與 Web 容器 (Servlet) 屬性對應的屬性和值清單，如表 1-7 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 16-36 HTTP 服務層級

指令	帶點名稱	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 Service 屬性對應的屬性和值清單。

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 16-37 執行緒池層級

指令	帶點名稱	輸出
list -m	server.thread-pools	thread-pool 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 Thread Pool 屬性對應的屬性和值清單。

下表列出了資源層級的指令、帶點名稱及對應的輸出。

表 16-38 資源層級

指令	帶點名稱	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。

表 16-38 資源層級 (續)

指令	帶點名稱	輸出
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 Connection Pool 屬性對應的屬性和值清單。

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 16-39 作業事件服務層級

指令	帶點名稱	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 Transaction Service 屬性對應的屬性和值清單。

下表列出了 ORB 層級的指令、帶點名稱及對應的輸出。

表 16-40 ORB 層級

指令	帶點名稱	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 Connection Manager 屬性對應的屬性和值清單。

下表列出了 JVM 層級的指令、帶點名稱及對應的輸出。

表 16-41 JVM 層級

指令	帶點名稱	輸出
list -m	server.jvm	無屬性，但顯示如下訊息：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。

使用 JConsole

本小節包含下列主題：

- 第 178 頁的「保護 JConsole 與 Application Server 間連線的安全」
- 第 179 頁的「將 JConsole 連線至 Application Server 的必要條件」
- 第 179 頁的「將 JConsole 連線至 Application Server」
- 第 180 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 為基礎。這表示受管理的元件以 MBean 表示。您可以使用 Java 2 Standard Edition (J2SE) 5.0 監視 JVM，並檢視 JVM MBean 以瞭解狀況。為公開此設備，Application Server 提供一個名為 System JMX Connector Server 的標準 JMX Connector Server 配置。當 Application Server 啟動時，會啟動此 JMX Connector Server 的實例，以將此設備公開給信任的用戶端。

Java Monitoring and Management Console (JConsole) 是常見的 JMX 連接器，可用來管理 JMX 後端。從 J2SE 5.0 開始，標準 JDK 發行軟體中就包含 JConsole (<http://java.sun.com/j2se/1.5.0/docs/tooldocs/share/jconsole.html>)。如需有關 JConsole 的更多資訊，請參閱

<http://java.sun.com/developer/technicalArticles/J2SE/jconsole.html>

當您配置 JConsole 搭配 Application Server 使用時，Application Server 會做為 JMX 連接器的伺服器端，而 JConsole 會做為 JMX 連接器慣用的客戶機端。

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 Platform Edition 之系統 JMX 連接器伺服器端是不安全的。
- 依預設，Application Server 的 Enterprise Edition 之系統 JMX 連接器伺服器端是安全的。
- 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。

例如，當您在諸如 `appserver.sun.com` 的機器上安裝 Platform Edition 網域時，將會在 DAS (Domain Administration Server，或稱管理伺服器，或簡稱網域) 的 `domain.xml` 中看到以下項目：

```
<!-- 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` 旗標為 *false*。若您是執行 Enterprise Edition，或已在 Platform Edition 開啓 JMX 連接器的安全性，則此旗標會設定為 *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 設定分為兩個部分：伺服器端與客戶機端。Application Server 網域會安裝在名為 `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，我們假設客戶機端是已安裝 Java SE 5.0 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 (Enterprise Edition 預設會啓用安全性)，才需要在客戶機端安裝 Application Server。若只是要在上述 Solaris 機器管理 Application Server Platform Edition 網域，則不需要在用戶端機器上安裝 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

本小節說明在 JMX 連接器上未啓用安全性的情況下，如何將 JConsole 連線至 Application Server。依預設，Application Server Platform Edition 上不會啓用安全性。

1. 啓動 `appserver.sun.com` 上的網域。

2. 執行 `JDK_HOME/bin/jconsole` 以啟動 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員的使用者名稱，密碼指的則是網域管理員的密碼。
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

本小節說明在 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，Application Server Enterprise Edition 上會啓用安全性。若已在 Platform Edition 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 (為呼叫此指令，您必須在本機安裝 Application Server)。
2. 在 `appserver.sun.com` 上啟動 Application Server Enterprise Edition。
由於這是 Enterprise Edition 網域，所以系統 JMX 連接器伺服器是安全的。
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 4849` (其中，4849 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。此時系統會提示您接受 `appserver.sun.com` 之 DAS 所傳送的憑證。
4. 按下 [y] 接受 `appserver.sun.com` 上之 Domain Administration Server 所傳送的憑證。
伺服器憑證會儲存在名為 `.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 DAS 的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"`
此時 JConsole 會自動信任此憑證。
6. 執行 `JDK_HOME/bin/jconsole` 以啟動 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員的使用者名稱，密碼指的則是網域管理員的密碼。

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。

Java 虛擬機器和進階設定

Java 虛擬機器 (JVM) 為解譯運算引擎，負責執行已編譯 Java 程式中的位元碼。JVM 會將 Java 位元碼轉譯為主機電腦的本機指令。做為 Java 程序的應用程式伺服器需要 JVM 才能執行，並支援在其上執行的 Java 應用程式。JVM 設定為應用程式伺服器配置的一部分。

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：

- [第 183 頁的「調校 JVM 設定」](#)
- [第 184 頁的「配置進階選項」](#)

調校 JVM 設定

定義相關設定以強化 Java 虛擬機器的使用，也屬於應用程式伺服器配置的一部分。若使用 Admin Console 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 將無法啟動。

- Javac 選項：為 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 [已啟用] 核取方塊。
JPDA 由應用程式開發者使用。
- 除錯選項：指定在啟用除錯功能的情況下，要傳遞至 JVM 的 JPDA 選項。
- RMI 編譯選項：為 `rmi` 編譯器輸入指令行選項。部署 EJB 元件後，Application Server 將執行 `rmi` 編譯器。

- 位元碼預處理程式：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。

配置進階選項

若要使用 Admin Console 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，以啟用動態重新載入應用程式。
啟用動態重新載入之後 (預設為啟用)，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，因為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。用戶端必須重新啟動階段作業。
- 重新載入輪詢間隔：為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。預設值為 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，以啟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：針對應用程式和模組的程式碼變更，定義檢查及動態重新載入的間隔。預設值為 2。
- 檢驗器：核取 [啟用檢驗器] 方塊，以驗證您的部署描述元檔案。此為選擇性選項。
- 預編譯：核取 [啟用預編譯] 方塊，以預編譯所有 JSP 檔案。

domain.xml 的帶點名稱屬性

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點號來分隔名稱，所以這些名稱稱為「帶點名稱」。

* (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。使用萬用字元符號的好處是它可以折疊帶點名稱的所有部分。例如，您可以將冗長的帶點名稱 `this.is.really.long.hierarchy` 縮寫為 `th*.hierarchy`。但是，必須一律使用 `.` 來分隔名稱的每個部分。`*` 可取得整個帶點名稱清單。

本附錄包含下列主題：

- [第 185 頁的「頂層元素」](#)
- [第 187 頁的「不能別名化的元素」](#)

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為 `domain`。
- 不能將伺服器實例命名為 `agent`。

下表列出了頂層元素及其對應的帶點名稱字首。

元素名稱	帶點名稱字首
應用程式	domain.applications
resources	domain.resources
配置	domain.configs

元素名稱	帶點名稱字首
伺服器	domain.servers 此元素中包含的每個伺服器均可做為 <i>server-name</i> 進行存取。其中， <i>server-name</i> 是伺服器子元素的名稱屬性值。
叢集	domain.clusters 此元素中包含的每個叢集均可做為 <i>cluster-name</i> 進行存取。其中， <i>cluster-name</i> 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，形式為 server1 格式的帶點名稱與帶點名稱 domain.servers.server1 (其中 server1 為伺服器實例) 對映。
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開頭的帶點名稱，這些帶點名稱在伺服器或叢集所參照的配置中被別名化為頂層名稱。

帶點名稱	別名化為	註釋
target.applications.*	domain.applications.*	該別名解析為僅由 目標 參照的應用程式。
target.resources.*	domain.resources.*	該別名解析為由 目標 參照的所有 jdbc-connection-pool、connector-connection-pool、resource-adapter-config 和所有其他資源。

下表列出了以伺服器名稱或叢集名稱開頭的帶點名稱，這些帶點名稱在伺服器或叢集所參照的配置中被別名化為頂層名稱。

帶點名稱	別名化為
target.http-service	config-name.http-service
target.iiop-service	config-name.iiop-service

帶點名稱	別名化為
<i>target.admin-service</i>	<i>config-name.admin-service</i>
<i>target.web-container</i>	<i>config-name.web-container</i>
<i>target.ejb-container</i>	<i>config-name.ejb-container</i>
<i>target.mdb-container</i>	<i>config-name.mdb-container</i>
<i>target.jms-service</i>	<i>config-name.jms-service</i>
<i>target.log-service</i>	<i>config-name.log-service</i>
<i>target.security-service</i>	<i>config-name.security-service</i>
<i>target.transaction-service</i>	<i>config-name.transaction-service</i>
<i>target.monitoring-service</i>	<i>config-name.monitoring-service</i>
<i>target.java-config</i>	<i>config-name.java-config</i>
<i>target.availability-service</i>	<i>config-name.availability-service</i>
<i>target.thread-pools</i>	<i>config-name.thread-pools</i>

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`。

asadmin 指令

`asadmin get`、`set` 與 `list` 指令共同為 Application Server 的抽象階層提供瀏覽機制。共有兩種階層：配置和監視，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。

配置階層提供可修改的屬性；而來自監視階層的管理元件只有唯讀屬性。配置階層並非嚴格基於網域的模式文件。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帶點名稱中的所有項目。如需使用 `get`、`set` 與 `list` 指令的範例，請參閱以下線上手冊：

```
get(1)
set(1)
list(1)
```


asadmin 公用程式

Application Server 包含一個名為 `asadmin` 的指令行管理公用程式。您可以使用 `asadmin` 公用程式來啟動與停止 Application Server，以及管理使用者、資源和應用程式。

本章包括下列小節：

- 第 190 頁的「`asadmin` 指令用法」
- 第 193 頁的「`Multimode` 指令」
- 第 193 頁的「`List`、`Get` 與 `Set` 指令」
- 第 194 頁的「伺服器生命週期指令」
- 第 195 頁的「`List` 與 `Status` 指令」
- 第 196 頁的「部署指令」
- 第 196 頁的「訊息佇列管理指令」
- 第 197 頁的「資源管理指令」
- 第 198 頁的「Application Server 配置指令」
- 第 202 頁的「使用者管理指令」
- 第 203 頁的「規則指令」
- 第 203 頁的「資料庫指令」
- 第 204 頁的「診斷與記錄指令」
- 第 204 頁的「Web 服務指令」
- 第 205 頁的「安全性服務指令」
- 第 205 頁的「密碼指令」
- 第 206 頁的「驗證 `domain.xml` 指令」
- 第 206 頁的「自訂 MBean 指令」
- 第 207 頁的「其他指令」

asadmin 指令用法

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` 公用程式來取代使用 Administration Console。

`asadmin` 公用程式可呼叫其他指令，以便識別您想要執行的作業或操作。這些指令須大小寫相符。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指令。選項也須大小寫相符。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 [開啟] 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為運算元。

範例 19-1 語法範例

```
asadmin command [-short_option] [short_option_argument]* [--long_option  
[long_option_argument]* [operand]*
```

```
asadmin create-profiler -u admin --passwordfile password.txt myprofiler
```

若要在 Solaris 平台上存取 Application Server `asadmin` 公用程式指令的線上手冊，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` 公用程式指令的完整用法資訊。若指定了一個指令，便會顯示該指令的使用資訊。使用 `--help` 選項但不指定指令，則會顯示所有可用指令的清單。

範例 19-2 help 指令範例

`asadmin --help` 可顯示一般說明

`asadmin command --help` 可顯示指定之指令的說明。

本小節包含下列主題：

- 第 190 頁的「多重模式與互動式模式」
- 第 191 頁的「本機指令」
- 第 191 頁的「遠端指令」
- 第 192 頁的「密碼檔案」

多重模式與互動式模式

您可以在指令 shell 呼叫或多重指令模式 (亦稱為 `multimode` 指令) 中使用 `asadmin` 公用程式。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。

您也可以在此互動式或非互動模式中執行 `asadmin` 公用程式。依預設，已啟用互動式模式選項。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` 呼叫中使用互動式模式選項。當您從指令提示符號一次執行一個指令，或是從檔案執行 `multimode` 時，可以在 `multimode` 中使用互動式模式選項。`multimode` 中的指令 (由輸入串流與其建立管道) 以及從其他程式呼叫的指令，都無法以互動式模式執行。

本機指令

本機指令可在沒有管理伺服器的情況下執行。然而，使用者必須登入至託管網域的機器，才能執行指令並擁有該安裝與網域目錄的存取 (權限)。

對於可在本機或遠端執行的指令，若已在環境或指令行中設定 `--host`、`--port`、`--user` 與 `--passwordfile` 選項中的任何一個選項，則該指令會以遠端模式執行。此外，若未在指令行或環境中設定任何本機選項，則該指令預設會在本機執行。

遠端指令

遠端指令的執行方式永遠是先連線到管理伺服器，然後在該處執行指令。這時一定要有執行中的管理伺服器。所有遠端指令都需要以下共用選項：

表 19-1 遠端指令必需的選項

短選項	選項	定義
-H	--host	正在執行網域管理伺服器的機器名稱。預設值是 <code>localhost</code> 。
-p	--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， <code>http://localhost:4848</code> 。Platform Edition 的預設連接埠號是 4848。
-u	--user	授權的網域管理伺服器管理使用者名稱。若已使用 <code>asadmin login</code> 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 <code>--user</code> 選項。

表 19-1 遠端指令必需的選項 (續)

短選項	選項	定義
	--passwordfile	<p>--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。</p> <p>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含 AS_ADMIN_PASSWORD、AS_ADMIN_USERPASSWORD 與 AS_ADMIN_ALIASPASSWORD、AS_ADMIN_MAPPEDPASSWORD。</p> <p>所有遠端指令都必須指定管理密碼以向網域管理伺服器認證 (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。對於必須為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。</p> <p>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)。</p> <p>為了安全性考量，asadmin 將不會讀取指定為環境變數的密碼。</p>
-s	--secure	若設定為 true，將使用 SSL/TLS 與網域管理伺服器通訊。
-I	--interactive	若設定為 true (預設)，將只提示您輸入必需的密碼與使用者選項。
-t	--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。
-e	--echo	將此選項設定為 true，則指令行敘述會回應在標準輸出上。預設是 false。
-h	--help	顯示指令的說明文字。

密碼檔案

為了安全性考量，您可以從檔案設定指令的密碼，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 19-3 密碼檔案內容

```
AS_ADMIN_PASSWORD=value
AS_ADMIN_ADMINPASSWORD=value
AS_ADMIN_USERPASSWORD=value
```


範例 19-3 密碼檔案內容 (續)

```
AS_ADMIN_MASTERPASSWORD=value
```

Multimode 指令

您可以使用 `multimode` 指令來處理 `asadmin` 指令。指令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。

若要呼叫 `multimode`，請輸入 `asadmin multimode`。

List、Get 與 Set 指令

`asadmin list`、`get` 與 `set` 指令共同為 Application Server 的帶點命名階層提供瀏覽機制。共有兩種階層：**configuration** 與 **monitoring**，且上述指令在兩者上都可執行。`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** 階層則有些不同。

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。

Application Server 帶點名稱使用「.」(點號)做為分隔完整名稱的分隔符。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。
- 名稱的一部分通常用來識別應用程式伺服器子系統和/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為「<classname>th*.hierarchy</classname>」。但請注意，必須一律使用 . 來分隔名稱的每個部分。

- 在 Solaris 上，執行使用 * 做為選項值或運算元的指令時，必須加上引號。
- 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)。此切換的有無，代表進行應用程式伺服器管理時，是選取兩個階層中的哪一個：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為階層中父系節點的完整名稱。為 list 指令提供此名稱時，該指令只會傳回該層級之下一層子項的名稱。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為節點 (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，則上述情況為真。您永遠不會遇到此情況，因為為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 來設定或取得特定應用程式伺服器子系統的屬性，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 (開頭為 /) 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 Application Server 中之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，以識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/刪除或啟動/停止網域、服務 (DAS) 或實例的指令。

表 19-2 伺服器生命週期指令

指令	定義
create-service	在自動啟動環境中配置 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，使用者必須啟動、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。

表 19-2 伺服器生命週期指令 (續)

指令	定義
<code>create-domain</code>	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 (其中每個網域都各自具有明確的管理身份)。網域可獨立存在，不需依賴其他網域。可存取指定系統上 <code>asadmin</code> 公用程式的任何使用者，都可以建立網域並將其配置儲存在想要的資料夾中。依預設，網域配置會建立在 <code>install_dir/domains</code> 目錄中。您可以置換此位置，以便將配置儲存在其他地方。。
<code>delete-domain</code>	刪除指定的網域。該網域必須已經存在，且已停止使用。
<code>start-domain</code>	啟動網域。若未指定網域目錄，將啟動預設 <code>install_dir/domains</code> 目錄中的網域。若其中有兩個以上的網域，則必須指定 <code>domain_name</code> 運算元。
<code>stop-domain</code>	停止指定網域的 Domain Administration Server。
<code>restore-domain</code>	從備份目錄復原網域下的檔案。
<code>list-domains</code>	列出網域。若未指定網域目錄，會列出預設 <code>install_dir/domains</code> 目錄中的網域。若有一個以上的網域，則必須指定 <code>domain_name</code> 運算元。
<code>backup-domain</code>	備份指定網域下的檔案。
<code>list-backups</code>	顯示備份儲存庫中所有備份的狀態資訊。
<code>shutdown</code>	以正常方式關閉管理伺服器與所有執行中的實例。若要重新啟動管理伺服器，必須以手動方式重新啟動。

List 與 Status 指令

`list` 與 `status` 指令可顯示已部署之元件的狀態。

表 19-3 List 與 Status 指令

指令	定義
<code>show-component-status</code>	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「 <code>status of app-name is enabled</code> 」或「 <code>status of app-name is disabled</code> 」。
<code>list-components</code>	列出所有已部署的 Java EE 5 元件。若未指定 <code>--type</code> 選項，將列出所有元件。
<code>list-sub-components</code>	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。

部署指令

部署指令可部署應用程式或取得用戶端 stub。

表 19-4 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 <code>--force</code> 選項已設定為 <code>true</code> ，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 <code>AppClient</code> 獨立模組或包含 <code>AppClient</code> 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。您也可以在此指令中使用 <code>--retrieve</code> 選項，以取得用戶端 stub。
undeploy	移除指定的已部署元件。

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 19-5 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 <code>create-jms-resource</code> 指令來建立具有指定實體目標之 <code>Name</code> 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 (又稱為 JMS 提供者) 是否已啟動並正在執行。當您啟動 <code>Application Server</code> 時，預設會啟動 JMS 服務。此外，它只會使用 <code>ping</code> 指令偵測 JMS 服務中的預設 JMS 主機。若使用 <code>ping</code> 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 19-6 資源管理指令

指令	定義
<code>create-jdbc-connection-pool</code>	以指定的 JDBC 連線池名稱註冊新的 JDBC 連線池。
<code>delete-jdbc-connection-pool</code>	刪除 JDBC 連線池。運算元可識別要刪除的 JDBC 連線池。
<code>list-jdbc-connection-pools</code>	取得已建立的 JDBC 連線池。
<code>create-jdbc-resource</code>	建立新的 JDBC 資源。
<code>delete-jdbc-resource</code>	移除具有指定之 JNDI 名稱的 JDBC 資源。
<code>list-jdbc-resources</code>	顯示已建立之 JDBC 資源的清單。
<code>create-jms-resource</code>	建立 Java 訊息服務 (JMS) 連線工廠資源或 JMS 目標資源。
<code>delete-jms-resource</code>	移除指定的 JMS 資源。
<code>list-jms-resources</code>	列出現有 JMS 資源 (目標與連線工廠資源)。
<code>create-jndi-resource</code>	註冊 JNDI 資源。
<code>delete-jndi-resource</code>	移除具有指定之 JNDI 名稱的 JNDI 資源。
<code>list-jndi-resources</code>	識別所有現有 JNDI 資源。
<code>list-jndi-entries</code>	瀏覽並查詢 JNDI 樹狀結構。
<code>create-javamail-resource</code>	建立 JavaMail 階段作業資源。
<code>delete-javamail-resource</code>	移除指定的 JavaMail 階段作業資源。
<code>list-javamail-resources</code>	列出現有 JavaMail 階段作業資源。
<code>create-persistence-resource</code>	註冊持續性資源。
<code>delete-persistence-resource</code>	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 <code>create-persistence-resource</code> 指令所建立)。
<code>list-persistence-resources</code>	顯示所有持續性資源。
<code>create-custom-resource</code>	建立自訂資源。自訂資源指定實作 <code>javax.naming.spi.ObjectFactory</code> 介面的自訂全伺服器資源物件工廠。
<code>delete-custom-resource</code>	移除自訂資源。
<code>list-custom-resources</code>	列出自訂資源。

表 19-6 資源管理指令 (續)

指令	定義
<code>create-connector-connection-pool</code>	增加具有指定之連線池名稱的新連接器連線池。
<code>delete-connector-connection-pool</code>	移除使用運算元 <code>connector_connection_pool_name</code> 指定的連接器連線池。
<code>list-connector-connection-pools</code>	列出已建立的連接器連線池。
<code>create-connector-resource</code>	註冊具有指定之 JNDI 名稱的連接器資源。
<code>delete-connector-resource</code>	移除具有指定之 JNDI 名稱的連接器資源。
<code>list-connector-resources</code>	取得所有連接器資源。
<code>create-admin-object</code>	增加具有指定之 JNDI 名稱的受管理物件。
<code>delete-admin-object</code>	移除具有指定之 JNDI 名稱的受管理物件。
<code>list-admin-objects</code>	列出所有受管理物件。
<code>create-resource-adapter-config</code>	為連接器模組建立配置資訊。
<code>delete-resource-adapter-config</code>	刪除為連接器模組在 <code>domain.xml</code> 中建立的配置資訊。
<code>list-resource-adapter-configs</code>	列出為連接器模組在 <code>domain.xml</code> 中建立的配置資訊
<code>add-resources</code>	在指定的 XML 檔案中建立指定資源。 <i>xml_file_path</i> 是包含要建立之資源的 XML 檔案的路徑。DOCTYPE 應該指定為 <code>resources.xml</code> 檔案中的 <i>install_dir/lib/dtds/sun-resources_1_2.dtd</i> 。
<code>ping-connection-pool</code>	測試 JDBC 連線池與連接器連線池中的連線池是否可用。例如，若為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 <code>ping</code> 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 Application Server 或資料庫已啟動。

Application Server 配置指令

您可以使用配置指令來配置 Application Server 的作業。本小節包含下列主題：

- 第 199 頁的「一般配置指令」
- 第 199 頁的「HTTP、IIOP 與 SSL 偵聽程式指令」
- 第 200 頁的「生命週期與稽核模組指令」
- 第 200 頁的「效能評測器和 JVM 選項指令」
- 第 201 頁的「虛擬伺服器指令」
- 第 201 頁的「執行緒池指令」
- 第 201 頁的「作業事件和計時器指令」

一般配置指令

您可以使用這些指令來管理 Application Server 元件的配置。

表 19-7 一般配置指令

指令	定義
<code>enable</code>	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。
<code>disable</code>	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。
<code>export</code>	將變數名稱標記成要自動匯出至後續指令的環境。除非使用 <code>unset</code> 指令取消設定數或結束 <code>multimode</code> ，否則所有後續指令都會使用指定的變數名稱值。
<code>get</code>	取得屬性的名稱與值。
<code>set</code>	設定一個或多個可配置屬性的值。
<code>list</code>	列出可配置的元素。在 Solaris 上，執行使用 * 做為選項值或運算元的指令時，必須加上引號。
<code>unset</code>	移除針對 <code>multimode</code> 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。

HTTP、IIOP 與 SSL 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 19-8 IIOP 偵聽程式指令

指令	定義
<code>create-http-listener</code>	增加新的 HTTP 偵聽程式。
<code>delete-http-listener</code>	移除指定的 HTTP 偵聽程式。
<code>list-http-listeners</code>	列出現有 HTTP 偵聽程式。
<code>create-iiop-listener</code>	建立 IIOP 偵聽程式。
<code>delete-iiop-listener</code>	移除指定的 IIOP 偵聽程式。
<code>list-iiop-listeners</code>	列出現有 IIOP 偵聽程式。
<code>create-ssl</code>	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。

表 19-8 IIOP 偵聽程式指令 (續)

指令	定義
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。

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 19-9 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 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	為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 JVM 選項指令

您可以使用效能評測器和 JVM 選項指令來管理效能評測器並控制這些元素。只有遠端模式支援這些指令。

表 19-10 效能評測器和 JVM 選項指令

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啟動伺服器。
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啟動伺服器。

表 19-10 效能評測器和 JVM 選項指令 (續)

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。若為效能評測器建立了 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。您必須重新啟動伺服器，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。

虛擬伺服器指令

您可以使用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 19-11 虛擬伺服器指令

指令	定義
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。
list-virtual-server	列出現有虛擬伺服器。

執行緒池指令

您可以使用執行緒池指令來控制這些元素。只有遠端模式支援這些指令。

表 19-12 執行緒池指令

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。

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 19-13 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。
list-timers	列出特定伺服器實例所擁有的計時器

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 19-14 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	列出檔案範圍認證支援的使用者與群組。此指令可列出檔案使用者中的可用群組。

監視資料指令

您可以使用監視資料指令來監視伺服器。只有遠端模式支援這些指令。

表 19-15 監視資料指令

指令	定義
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 設定為 ON 時才會收集資料。
stop-callflow-monitoring	停止收集請求呼叫流程資訊。

規則指令

您可以使用規則指令來管理伺服器的規則。只有遠端模式支援這些指令。

表 19-16 規則指令

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。
list-management-rules	列出可用的管理規則。

資料庫指令

您可以使用資料庫指令來啟動和停止 Java DB 資料庫 (基於 Apache Derby)。只有本機模式支援這些指令。

表 19-17 資料庫指令

指令	定義
start-database	啟動 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 19-18 診斷與記錄指令

指令	定義
generate-diagnostic-report	產生包含應用程式伺服器安裝詳細資訊 (例如，應用程式伺服器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。
generate-jvm-report	顯示指定之目標實例的執行緒 (堆疊追蹤的傾印)、類別與記憶體，包括 Domain Administration Service。此指令只適用於應用程式伺服器實例程序。此指令將取代傳統技術，如傳送 <code>ctrl+break</code> 或 <code>kill -3</code> 訊號到應用程式伺服器程序。若目標伺服器實例並未在執行中，則此指令將無法運作。
display-error-statistics	顯示自上次伺服器重新啟動後， <code>server.log</code> 中記錄的嚴重性與警告的摘要清單。
display-error-distribution	顯示實例 <code>server.log</code> 中整個模組的錯誤分佈狀況。 TM
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 19-19 Web 服務指令

指令	定義
configure-webservice-management	配置已部署之 Web 服務端點的監視或 <code>maxhistorysize</code> 屬性。
create-transformation-rule	建立可套用至 Web 服務作業的 XSLT 變換規則。您可以將規則套用至請求或回應 (或兩者)。
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伺服器。
unpublish-from-registry	從登錄伺服器取消發佈 Web 服務工件。
list-registry-locations	顯示已配置之 Web 服務項目存取點的清單。

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 19-20 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。為順利執行此指令，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-maps	列出屬於指定之連接器連線池的安全對映。
create-message-security-provider	可供管理員為指定的訊息層 (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-provider	可供管理員為指定的訊息層 (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)。
create-auth-realm	增加指定的認證範圍。
delete-auth-realm	移除指定的認證範圍。
list-auth-realms	列出現有認證範圍。

密碼指令

您可以使用密碼指令來管理密碼以確保 Application Server 安全性。

表 19-21 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。別名是格式為 <code>\${ALIAS=password-alias-password}</code> 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 domain.xml 指令

XML 驗證指令可驗證 domain.xml 檔案的內容。

表 19-22 驗證 domain.xml 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 19-23 自訂 MBean 指令

指令	定義
create-mbean	建立並註冊自訂 MBean。若目標 MBeanServer (DAS) 並未執行，則不會註冊 MBean。
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。

其他指令

您可以使用其他指令來管理 Application Server 的各個層面。

表 19-24 其他指令

指令	定義
login	可讓您登入網域。若已在不同的機器 (本機) 建立不同的應用程式伺服器網域，從這些機器中的任何一部呼叫 <code>asadmin</code> 即可管理位於其他地方 (遠端) 的網域。當您選擇特定機器做為管理用戶端，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 <code>asadmin</code> 指令稱為遠端指令。 <code>asadmin login</code> 指令可簡化管理此類遠端網域的作業。 <code>login</code> 指令只會以互動式模式執行。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 <code>.asadminpass</code> 檔案。該檔案即為使用 <code>--saveLogin</code> 選項時 <code>create-domain</code> 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。
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 <code>asadmin</code> 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。

索引

A

ACC
 請參閱容器
 應用程式用戶端, 77
Admin Console, 28
applet, 77
Application Server
 重新啟動, 81
 關閉, 81
asadmin command, 136
 create-threadpool, 136
asadmin 公用程式, 29
asadmin 指令, 136
 delete-threadpool, 136

B

bean-cache, 監視屬性名稱, 148

C

cache-hits, 148
cache-misses, 148
CORBA, 133
create-domain 指令, 34

D

delete-domain 指令, 35

E

EAR 檔案, 46
EJB JAR 檔案, 46
execution-time-millis, 146

G

get 指令, 監視資料, 167

H

HTTP 服務
 HTTP 偵聽程式, 130-132
 持續作用子系統, 131
 虛擬伺服器, 129-130
 請求處理執行緒, 131
HTTP 階段作業, 78
HTTP 偵聽程式
 接收器執行緒, 131
 預設虛擬伺服器, 131
 簡介, 130-132

I

IIOP 偵聽程式, 134

J

J2SE 軟體, 42

Java 命名和目錄服務, 請參閱 JNDI, 77
Java 訊息服務 (JMS), 請參閱 JMS 資源, 55
JavaMail, 28
JavaServer Pages, 77
JCE 提供者
 配置, 118
JDBC, 28
 驅動程式, 126
JMS
 目標資源, 57-58
 連線工廠, 57
 提供者, 58-59
 實體目標, 58
jms-max-messages-load, 148
JMS 提供者, 55
JMS 資源
 目標資源, 56-57
 主題, 56-57
 佇列, 56-57
 連線工廠資源, 56-57
 實體目標, 56-57
 簡介, 56-57
jmsra 系統資源配接卡, 57
JNDI, 77
 EJB 元件的查詢名稱, 47
 外部儲存庫, 71
 名稱, 69
 自訂資源, 使用, 71
 查詢及其關聯的參照, 70
JSP, 請參閱 JavaServer Page, 77

K
keystore.jks 檔案, 97-98

L
list-domains 指令, 35
list 指令, 監視, 166

M
Message Queue 軟體, 55

N
numbeansinpool, 147
numexpiredsessionsremoved, 148
numpassivationerrors, 148
numpassivations, 148
numpassivationsuccess, 148
numthreadswaiting, 147

O
Oasis Web Services Security, 參閱 WSS
ORB, 133
 IIOP 偵聽程式, 134
 服務, 監視, 153
 配置, 134
 簡介, 133

R
RAR 檔案, 46
RSA 加密, 118

S
servlet, 77
start-domain 指令, 35
stop-domain 指令, 36
Sun Java System Message Queue 軟體, 55

T
total-beans-created, 147
total-beans-destroyed, 148
total-num-errors, 146
total-num-success, 146
truststore.jks 檔案, 97-98

W

- WAR 檔案, 46
- Web 服務, 27
- Web 階段作業, 請參閱 HTTP 階段作業, 78
- Web 應用程式, 46
- 工作佇列, 請參閱執行緒池, 136
- 中央儲存庫, 應用程式部署到, 43
- 用戶端存取, 27
- 用於應用程式的服務, 27
- 目標
 - 已部署的應用程式, 44
 - 資源, JMS, 57-58
 - 實體, JMS, 58
- 目標, JMS, 簡介, 56-57
- 目錄部署, 45
- 外部儲存庫, 存取, 71
- 主題, JMS, 56-57
- 企業 Java Bean
 - 永久性, 77
 - 有狀態階段作業, 81
 - 池儲存, 80, 81
 - 快取, 77, 80, 81
 - 建立, 77
 - 使用中, 81
 - 計時器服務, 81
 - 訊息引導, 81
 - 訊息導引, 77
 - 階段作業, 77
 - 授權, 77
 - 啟動, 77
 - 閒置, 80, 81
 - 無狀態階段作業, 80
 - 鈍化, 77, 80, 81
 - 實體, 77, 80, 81
- 企業應用程式, 46
- 回復
 - 請參閱作業事件
 - 回復, 125
- 安全性, 27
- 有狀態階段作業 Bean, 請參閱企業 Java Bean, 81
- 自訂資源, 使用, 71
- 自動部署應用程式, 44
- 池儲存
 - 企業 Java Bean, 80, 81
- 作業事件, 125
 - 分隔, 126
 - 分散式, 126
 - 企業 Java Bean, 80
 - 回復, 125, 126
 - 完成, 126
 - 管理員, 126
 - 確定, 125
 - 關聯, 126
 - 屬性, 126
- 作業事件服務, 監視, 154
- 作業事件管理, 28
- 作業事件管理員
 - 請參閱作業事件
 - 管理員, 126
- 快取
 - 企業 Java Bean, 80
 - 停用, 80
- 伺服器管理, 28
- 佇列
 - 工作
 - 請參閱執行緒池, 136
- 佇列, JMS, 56-57
- 物件請求代理程式 (ORB), 133
 - 配置, 134
 - 簡介, 133
- 命名, JNDI 和資源參照, 70
- 命名和目錄服務, 27
- 命名服務, 27
- 命名慣例, 針對應用程式, 47
- 服務, 計時器, 81
- 重新啟動伺服器, 36
- 重新部署應用程式, 44
- 持續作用子系統, HTTP 服務, 131
- 計時器
 - 請參閱企業 Java Bean
 - 計時器服務, 81
- 計時器服務
 - 請參閱企業 Java Bean
 - 計時器服務, 81
- 記錄
 - 記錄程式名稱空間, 139-140
 - 簡介, 137-138
- 記錄記錄, 137-138

- 訊息傳送, 28
- 連接埠偵聽程式, 42
- 連接器, 28
- 連接器連線池, JMS 資源和, 57
- 連接器資源, JMS 資源和, 57
- 連線, 工廠, JMS, 57
- 連線工廠, JMS, 簡介, 56-57
- 效能
 - 問題, 80
 - 提高, 80
- 高可用性, 26
- 容器, 27
 - applet, 77
 - J2EE, 77
 - servlet
 - web, 77
 - 請參閱容器, 77
 - web, 77
 - 企業 Java Bean, 77, 80-81
 - 配置, 80-81
 - 應用程式用戶端, 77
- 階段作業
 - HTTP, 78, 80
 - ID, 79
 - 自訂 ID, 79
 - 刪除, 79
 - 刪除資料, 79
 - 非使用中的, 79
 - 配置, 78-79
 - 管理, 78
 - 檔案名稱, 79
 - 儲存, 80
 - 儲存資料, 79
- 階段作業管理程式, 78
- 執行緒, 移除, 136
- 執行緒池
 - 工作佇列, 136
 - 執行緒不足, 135
 - 逾時, 136
 - 閒置, 136
- 接收器執行緒, 在 HTTP 偵聽程式中, 131
- 清除間隔, 79
- 逾時, 執行緒池, 136
- 提供者, JMS, 58-59
- 無狀態階段作業 Bean, 請參閱企業 Java Bean, 80
- 虛擬伺服器, 簡介, 129-130
- 資料庫
 - JNDI 名稱, 69
 - 資源參照, 70
- 資源 RAR 檔案, 46
- 資源配接卡, 126
 - jmsra, 57
- 資源參照, 70
- 資源管理員, 126
- 網域
 - 建立, 34-35
 - 部署應用程式到, 43
- 實體 Bean
 - 請參閱企業 Java Bean
 - 實體, 80
- 實體目標, JMS, 58
- 監視
 - bean-cache 屬性, 148
 - ORB 服務, 153
 - 作業事件服務, 154
 - 使用 get 指令, 167
 - 使用 list 指令, 166
 - 容器子系統, 142-143
- 範圍, 憑證, 92
- 部署規劃, 45
- 線上手冊, 29
- 請求處理執行緒, HTTP 服務, 131
- 應用程式
 - 目錄部署, 45
 - 自動部署, 44
 - 命名慣例, 47
 - 重新部署, 44
 - 效能, 80
 - 部署規劃, 45
- 應用程式用戶端 JAR 檔案, 46
- 叢集, 26
- 關鍵點作業, 127
- 關鍵點間隔, 127