SIP：建立新一代電信應用程式

Session Initiation Protocol 簡化了電信網路的應用程式開發

等級：入門

Nick Williams (dwsip@ubiquity.net)，Ubiquity Software Corporation 開發人員

2003 年 9 月 30 日

現在，電信網路的應用程式開發比以往更加輕鬆，您可使用開放式網際網路標準，例如 Session Initiation Protocol (SIP)，拋棄過去使用的專有通訊協定。若整合運用 SIP Servlet API 形式的 Java 技術功能和簡易性，應用程式開發人員即可大幅縮短爲使用者建立並部署新服務的時間，善用這項變革就表示精通 SIP。在本文中，您將瞭解 SIP 的運作方式，然後根據此知識學習如何使用 Java SIP Servlet API 建置令人振奮的新一代應用程式。SIP 簡介結尾提供程式碼範例，說明實際的 SIP 應用程式開發案例。

未來將不再區分語音及資料網路，而是以單一的整合網路，提供所有通訊基礎。如此一來，除了降低成本之外，這種整合也可以締造全新的服務趨勢，Session Initiation Protocol (SIP) 將是這項變革中最重要的控制機制。隨著網路設備廠商及電信服務業者逐漸採用 SIP 作爲通訊協定，SIP 將越來越普及，此外，第三代合作夥伴計畫 (3rd Generation Partnership Project, 3GPP) 最近也採用 SIP 作爲新一代網路的呼叫控制機制，同時，企業的語音傳輸 (Voice Over IP) 使用率也快速成長。基於以上種種因素，電信應用程式開發速度的大躍進指日可待。

在本文中，您將進一步瞭解 SIP 通訊協定及 SIP Servlet API，並且參考 User Agents (UA) 及 Proxy 伺服器在 SIP 中 所扮演的角色，同時掌握如何建立兩個裝置之間的通訊。本文還會提供 SIP Servlet API 及相關概念的簡介，此外，也會提到 SIP Servlet 的行爲，例如進行代理、扮演啓動與終止 UA 及 Back-to-Back User Agent (B2BUA) 的角色，最後，透過 SIP Servlet 範例程式碼，說明使用 SIP Servlet API 進行程式開發有多麼容易。

何謂 SIP？

SIP 應用程式層通訊協定可建立並管理裝置之間的多媒體通訊。

網際網路工程研究團隊 (IETF) 在 1999 年通過 SIP，成爲 Request For Comment (RFC) 2543 標準，以 Multiparty Multimedia Session Control (MMUSIC) Working Group 較早開發的網路多媒體爲建置基礎，此後，SIP 經歷過數個 RFC 修訂，發展到目前的 RFC 3261。

> 在 SIP 階段作業期間可以交換任何媒體，且使用各種通訊協定。SIP 通常使用：
>
> 用來判斷媒體種類的 Session Description Protocol (SDP)。請參閱 RFC 2327。
>
> 階段作業期間實際傳輸媒體資料的

此通訊協定不會限制呼叫的本質或內容，可讓使用者自由選擇要如何使用階段作業及交換通訊媒體。因此，SIP 階段作業不僅限於語音，建立階段作業後，使用者即可交換任何類型的媒體。

Real Time Protocol (RTP)。請參閱 RFC 1889。

Back to top

SIP 概念

SIP RFC 定義 SIP 網路所需的幾個主要概念和元素：

- *使用者代理程式* (UA) 作爲一個端點，可讓使用者建立並管理通訊階段作業。UA（無論是 SIP 電話或軟體應用程式）可處理階段作業的設定及管理工作，如傳送、終止及服務呼叫等。此外，UA 可說明使用者可用性，並協調階段作業功能。
- 如果某 UA *（發話端）*邀請其他 UA *（受話端）*加入通訊階段作業，即建立*階段作業*。
- *SIP 訊息*是文字型實體。訊息可分爲兩種：*要求*和*回應*。要求是某 UA 傳給另一個 UA 的訊息，然後該 UA 會再傳送回應。（下一節會進一步討論 SIP 訊息的細節。）發話端傳給受話端的訊息會往下移；反之，從相反方向傳送的訊息會往上移。
- *SIP Proxy 伺服器*通常會處理登錄、實作呼叫傳送原則，以及執行驗證和授權。SIP Proxy 伺服器的主要任務，就是確保要求能傳至其他更靠近目標使用者的實體。Proxy 會解譯並重新寫入（必要時）要求訊息的部分內容，然後再轉遞。SIP 訊息傳送給受話 UA 時，可能會經過數個 SIP Proxy 伺服器。UA 的設定功能通常是將本身產生的要求傳送給特定 SIP Proxy 伺服器，在這種情況下，Proxy 伺服器就是*出埠 Proxy*。
- *SIP 位址* (*SIP URL*) 可在建立通訊階段作業期間識別特定使用者。位址就像電子郵件位址，但 SIP 位址的字首有個 sip:。例如，您桌上的電話可能會有下列 SIP 位址：sip:user@194.195.100.20。

  一般而言，您會希望人們使用附有貴公司網域名稱的 SIP 位址聯絡您，如 sip:user@ubiquity.net。因此，SIP 可讓您執行登錄程序，以建立 SIP 位址與一或多個 UA 的關聯，接著，呼叫 SIP 位址後就會傳到 SIP Proxy 伺服器。SIP Proxy 伺服器會執行登錄查閱，以判斷應該將該呼叫導向哪一個 UA。此外，您也可以隨時修改並刪除登錄資訊。

  針對單一 SIP 位址登錄數個 UA，可能會導致通知多個 UA 並開始撥打電話，這就是所謂的「分流」(forking)。分流可分爲兩種：按順序及平行的分流。若使用*按順序分流*，就會在撥話一段特定時間後，按順序提示每個 UA。反之，若使用*平行分流*，就會同時提示所有 UA，而由第一個 UA 建立的階段作業則負責回應，或視計時而定可能由多個 UA 回應。

呼叫/*通話*代表發話端 UA 與受話端 UA 傳送的所有訊息。

*對話*是兩個 UA 之間持續一段時間的 SIP 關係。其中包括 UA 之間傳送的

SIP 訊息，例如透過 Proxy 傳送的訊息。

*交易*發生在用戶端與伺服器之間，是由下列訊息所組成：從用戶端傳給伺服器的第一個要求乃至伺服器傳到用戶端的最終回應所產生的所有訊息。

Back to top

SIP 訊息

SIP 通訊會使用類似 HTTP 通訊協定的文字要求/回應模式，用戶端將要求傳給伺服器，然後伺服器再回應用戶端。SIP 規格可定義六種要求訊息及六種回應訊息，如表 1 所示。

SIP 要求	**說明**
REGISTER	可建立並管理登錄。
INVITE	可啓動通訊階段作業，即由發話端傳給受話端，邀請他們加入階段作業。
ACK	發話端收到 INVITE 要求的最終回應後再送出的要求。
BYE	由發話端或受話端送出，以終止階段作業的要求。
CANCEL	可用來取消 (CANCEL) 及邀請 (INVITE) 尚未收到最終回應的要求。
OPTIONS	要求媒體資訊。

SIP 回應	**說明**
1xx Information	可提供發話端進度回應，如「100 嘗試中」(100 Trying)、「180 撥打中」(180 Ringing)。
2xx Success	確認已接受要求。
3xx Redirect	通知發話端應傳送要求的替代位置。
4xx Request Failure	表示未成功處理要求；如「404 找不到」(404 Not Found)。
5xx Server Failure	表示伺服器本身發生錯誤。例如，503 回應表示伺服器暫時無法處理要求，因爲伺服器本身超載或正在維護中。
6xx Global Failure	表示特定使用者的整體失敗。

這可能是*暫時性*或*最終的回應訊息*；暫時性回應顯示進度且不會終止 SIP 交易。1xx 回應是暫時性的。最終回應則會終止 SIP 交易。所有 2xx、3xx、4xx、5xx 及 6xx 回應皆屬最終回應。

要求訊息的結構如下：要求行、標頭，然後是內文。*要求行*就是訊息的第一行，包含方法名稱、要求 URI、由單一空格 (SP) 字元分隔的通訊協定版本，以及回車換行 (Carriage Return Line Feed, CRLF)。清單 1 是範例要求（不含內文）：

清單 1. INVITE 要求

```
INVITE sip:callee@143.145.52.13 SIP/2.0
Call-ID: 16610637811111548084@143.145.52.134
Via: SIP/2.0/UDP 143.145.52.134:5070;branch=z9hG4bKC1C334860000F6D31DE9
Via: SIP/2.0/UDP app.ubiquity.net
From: sip:caller@ubiquity.net;tag=1738655730
To:sip:callee@143.145.52.13
CSeq:1 INVITE
contact:sip:143.145.52.134:5070
Accept:application/sdp
User-Agent:Ubiquity Third Party Call Control/-7671573430297227200
Max-Forwards: 70
Content-Length: 0
```

回應訊息的結構如下：狀態行、標頭，然後是內文。*狀態行*是由通訊協定版本、數字*狀態碼*及其相關文字片語所組成，每個元素皆由單一 SP 字元分隔，且由 CRLF 結束。清單 2 是清單 1 中 INVITE 要求的相關回應 100 TRYING：

清單 2. 100 TRYING 回應

```
SIP/2.0 100 Trying
Via: SIP/2.0/UDP 143.145.52.134:5070;branch=
z9hG4bKC1C334860000F6D31DE90,SIP/2.0/UDP app.ubiquity.net
From: sip:caller@ubiquity.net;tag=1738655730
To: sip:callee@143.145.52.13;tag=4E880060-97A
Date: Sun, 05 Mar 2000 23:01:39 GMT
Call-ID: 16610637811111548084@193.195.52.134
Server: Cisco-SIPGateway/IOS-12.x
CSeq: 1 INVITE
Allow-Events: telephone-event
```

```
Content-Length: 0
```

所有 SIP 訊息標頭都包含欄位名稱及欄位值。SIP 標頭可分爲五種：

- 要求標頭
- 回應標頭
- 一般標頭
- 實體標頭
- 使用者定義標頭

此外，每個 SIP 訊息都需要數個強制標頭：

- **To:** 包含要求目的地的 SIP 位址。
- **From:** 表示發出要求的人。
- **CSeq:** 包含指令順序，以確保訊息是按照發出的順序來處理。
- **Call-ID:** SIP Proxy 伺服器會使用 Call-ID 標頭識別某階段作業的訊息。Call-ID 是專門識別階段作業的隨機產生字串。
- **Via:** 包含發話端與受話端之間的訊息，其經過的 SIP 裝置相關資訊。此外，Via 標頭可透過與要求相同的 SIP 裝置，以反方向傳送回應。
- **Contact:** 包含受話端的實際位置，可能與 From 標頭中的發送端位址不同。

其他選用標頭可傳送內容類型及內容長度等重要資訊。

Back to top

基本的通話流程

表 2 說明在兩個 UA 之間建立通訊階段作業的 SIP 訊息流程範例。SIP Proxy 伺服器是網路的一部分。

1	從 ua1 傳送 INVITE 到 Proxy
2	從 Proxy 傳送 100 Trying 回應到 ua1，Proxy 由下傳送要求到 ua2。
3	從 Proxy 傳送 INVITE 到 ua2。
4	從 ua2 傳送 100 Trying 到 Proxy。
5	從 ua2 傳送 180 Ringing 回應到 Proxy，表示 ua2 正在提示該使用者。
6	從 Proxy 傳送 180 Ringing 回應到 ua1。
7	從 ua2 傳送 200 OK 回應到 Proxy，表示 ua2 已接受通話。
8	從 Proxy 傳送 200 OK 回應到 ua1。

9	從 ua1 直接傳送 ACK 到 ua2，表示階段作業開始。
	…
	已建立通訊階段作業
	…
10	任一 UA 繞過 Proxy 傳送 BYE 到另一個 UA。
11	另一方繞過 Proxy 傳送 200 OK 回應。

您可在資源段落找到完整的通話日誌。

以上範例只是最簡單的 SIP 通話流程。涉及的 SIP 實體越多，這種通話流程就很容易變得越複雜，比方說，如果單一 SIP 位址有兩個登錄端點，Proxy 伺服器就必須執行通話分流。

其中有幾點需要注意。原本的 INVITE 是*初始要求*，而 BYE 要求則是*後續要求*。此外，BYE 要求和其他後續要求會直接在兩個 UA 之間傳送，不會經過 Proxy。不過，Proxy 沒有辦法確保可查看所有後續要求。

Back to top

記錄傳送

一般而言，您只有在建立通訊階段作業以執行登錄查閱和轉遞訊息時，才會使用 SIP Proxy 伺服器，然後，所有後續要求就會直接在發話端和受話端之間傳送。然而，Proxy 可能要查看該階段作業期間的所有後續 SIP 訊息，使用者可能要 Proxy 伺服器執行此作業，以便產生計費記錄。若要達到此目的，Proxy 伺服器就要查看初始 INVITE 和終止 BYE 要求，在此情況下，Proxy 伺服器會啓用記錄傳送，即將 Record-Route 標頭新增到要求中。每個想要留在傳訊路徑上的 Proxy 伺服器就會在發話端傳訊給受話端時，將 Record-Route 標頭插入初始 INVITE。

受話端 UA 必須保留 Record-Route 標頭中的資訊，然後傳回包含 Record-Route 標頭副本的回應。受話端 UA 最後會收到回應，且必須保留 Record-Route 標頭中的資訊。每個 UA 都會將該資訊存成*傳送路徑集 (route set)*，傳送路徑集的資訊則將 Route 標頭加到後續要求中，然後，Proxy 伺服器就會使用 Route 標頭的資訊，判斷要傳送要求的 SIP 裝置。

Back to top

進階服務

除了支援基本功能外，SIP 伺服器可當作服務建立平台，即讓開發人員擴充 SIP Proxy 伺服器的基本功能，並且建立新的應用程式和服務。

過去爲了提供服務建立平台，往往造就許多專有的應用程式建立模式及環境，雖然達到時間要求，但開發社群希望有更開放的模式。提出 SIP Servlet API，就是爲了解決此問題。

Back to top

SIP servlets：基本概念

慣用電話系統應用程式不僅開發成本昂貴，且相當耗時。

在 Java Community Process (JCP) 下開發的 SIP Servlet API 屬於 Java Specification Request (JSR) 116，可提供標準規格及一系列跨平台介面，以便在統一開放平台上開發及部署可攜式服務。API 進一步延伸了 HTTP Servlet 模式，所以有共同的概念，例如 service 方法、JAR 型檔案格式及部署描述子。

標準 SIP Proxy 伺服器可當作 SIP 應用程式伺服器的基礎。*儲存器*是應用程式伺服器 (SIP A/S) 的元件，可提供 SIP Servlet API 規格所指定的環境，以便 SIP servlet 運作。該儲存器會輸入並起始設定 Servlet，然後在 SIP 訊息抵達時呼叫相關 Servlet。SIP A/S 停止時，儲存器就會刪除所有 Servlet，因此，儲存器可執行許多 HTTP Servlet 儲存器所執行的功能。

SIP servlet 是由類別所組成，可延伸 javax.servlet.sip.Servlet 並執行相關服務方法，然後，編譯過的類別就會與 SIP 部署描述子一起封裝到 Servlet Archive 或 SAR 檔。

XML 型部署描述子包括 servlet 配置資訊、訊息比對原則及一般資訊，例如 Servlet 名稱。SAR 檔會部署到 SIP 應用程式伺服器。

SIP Servlet API 包含一系列物件及介面，可提供許多 SIP 概念的進階擷取，不需要擔心管理交易等 SIP 細節。表 3 詳述其中幾個最重要的項目。

類別 / 介面	**說明**
SipServlet	基本 SIP Servlet 物件應再細分，以建立 SIP Servlet，此類別會透過服務方法接收送入訊息，可分別針對送入要求及回應呼叫 doRequest 或 doResponse。然後，這兩個方法會分派要求方法或狀態碼到下列其中一個方法： • doInvite：適用於 SIP INVITE 要求 • doAck：適用於 SIP ACK 要求 • doOptions：適用於 SIP OPTIONS 要求 • doBye：適用於 SIP BYE 要求 • doCancel：適用於 SIP CANCEL 要求 • doRegister：適用於 SIP REGISTER 要求 • doSubscribe：適用於 SIP SUBSCRIBE 要求 • doNotify：適用於 SIP NOTIFY 要求 • doMessage：適用於 SIP MESSAGE 要求 • doInfo：適用於 SIP INFO 要求 • doProvisionalResponse：適用於 SIP 1xx 參考資訊回應 • doSuccessResponse：適用於 SIP 2xx 回應 • doRedirectResponse：適用於 SIP 3xx 回應 • doErrorResponse：適用於 SIP 4xx、5xx 及 6xx 回應

ServletConfig	Servlet 儲存器會在起始設定期間用此來傳遞資訊給 Servlet。
ServletContext	Servlet 會用此跟其 Servlet 儲存器通訊；例如，儲存屬性、分派要求或寫入日誌檔。
SipServletMessage	定義 SIP 要求和回應的共用部分。
SipServletRequest	提供 SIP 要求訊息的進階存取。
SipServletResponse	提供 SIP 回應訊息的進階存取。
SipFactory	適用於各種 SIP Servlet API 擷取的原廠介面。
SipAddress	表示 SIP From 和 To 標頭。
SipSession	建立相同 SIP 階段作業的 SIP 訊息關聯，亦稱爲通話日誌。如果兩個訊息的 Call-ID（即 RFC 3261 所提及的 From 和 To 標頭）相同，即屬於相同的 Sip 階段作業。
SipApplicationSession	表示應用程式實例，可作爲應用程式資料儲存區並提供所含通訊協定階段作業的存取權。
Proxy	表示代理 SIP 要求的作業，可提供代理執行方式的控制權；如記錄傳送及受監控模式。

執行時，SIP A/S 會收到不同的 SIP 訊息。如果是初始要求（即儲存器事先不知道的要求），該儲存器就會使用部署描述子中的訊息比對原則，以判斷是否要傳送訊息給 Servlet。根據原則呼叫 Servlet 的 service 方法。預設實作服務方法會嘗試識別要求類型，然後呼叫其中一個 doXXX 方法，例如 doInvite()。

處理要求是指轉遞訊息或傳回最終回應，在前述情況下，會在未來某個時間收到相關回應，因爲回應總是會按照要求的相反路徑傳回，收到初始要求的最後回應後，就會建立所謂的*應用程式路徑*實體。應用程式路徑可確保能夠正確傳送後續要求。

Back to top

建立簡單的 Servlet

爲了說明使用 SIP Servlet 模式有多方便，就讓我們來建立一個範例 Servlet。該 Servlet 會接聽 INVITE 要求，且一律將 INVITE 轉遞到目的地 UA。清單 3 是簡單的 Servlet：

清單 3. 範例 Servlet

```
public class SampleServlet extends SipServlet
```

```
{
private static final String SERVLET_INFO = "SampleServlet,  "+
"1.0, Copyright Ubiquity Software Corporation, 2003"

  public void doInvite(SipServletRequest a_Request)
  throws ServletException, java.io.IOException
  {
   try
     {
     a_Request.send();
     }
   catch (IOException e)
     {
     throw new ServletException("Could not send request", e);
     }
  }


public String getServletInfo()
{
return "SampleServlet, 1.0, Copyright   Ubiquity, 2003";
}
}
```

清單 4 是隨附的部署描述子。請注意，指定此 Servlet 的原則比對只處理 INVITE 要求：

清單 4. 部署描述子

```
<sip-app>
    <display-name>Sample Servlet</display-name>
    <servlet>
        <servlet-name>SampleServlet</servlet-name>
        <display-name>SampleServlet</display-name>
        <servlet-class>net.ubiquity.servlet.sample.SampleServlet</servlet-class>
    </servlet>

    <servlet-mapping>
```

```
      <servlet-name>SampleServlet</servlet-name>
      <pattern>
           <equal>
             <var>request.method</var>
             <value>INVITE</value>
           </equal>
      </pattern>
   </servlet-mapping>
</sip-app>
</code>
```

Servlet 會採取的最簡單動作，是直接傳送訊息而不加以修改，就像範例 Servlet 一樣。若只載入範例 Servlet，該要求會從 SIP A/S 傳到指定受話端 UA。誠如前述，受話端傳回發話端的回應是相反路徑，雖然在此情況下 SIP A/S 不會將它們傳給 Servlet。

Back to top

Proxy 行爲

誠如前述，Servlet 可直接傳送訊息，不需要修改。不過，如果必須將要求往下傳且在收到回應後進一步處理，您就需要建立虛擬物件。此外，如果必須將要求傳到數個位置，也需要虛擬物件，就像平行或按順序分流一樣。虛擬物件可讓您指定將所有後續要求傳給 Servlet（前述記錄傳送程序）。

如果 Servlet 必須執行一或多個這類活動，就必須呼叫 SipServletRequest 的 getProxy() 方法，才能取得虛擬物件。然後，透過呼叫虛擬物件的相關方法，才能取得所需行爲，例如，清單 5 中的程式碼會啓動記錄傳送功能及監控模式來代理要求：

清單 5. 記錄傳送

```
public void doInvite(SipServletRequest a_Request) throws ServletException,
IOException
{
   // Required by the Ubiquity Container to establish session state.
   SipApplicationSession appSession = a_Request.getApplicationSession();
   SipSession sipSession = a_Request.getSession();

   Proxy p = a_Request.getProxy(true);
```

```
    // Proxy in supervised mode.  This ensures that we see the response which will
    // be passed to us through the appropriate doResponse method.
    p.setSupervised(true);

    // If you want to see the ACK and the BYE, use RecordRouting.
    p.setRecordRoute(true);

    System.out.println("Proxying request: " + a_Request.getRequestURI());

    p.proxyTo(a_Request.getRequestURI());
}

public void doAck(SipServletRequest a_Request) throws ServletException, IOException
{
    System.out.println("ACK: " + a_Request);
}

public void doBye(SipServletRequest a_Request) throws ServletException, IOException
{
    System.out.println("BYE: " + a_Request);
}

public void doProvisionalResponse(SipServletResponse a_Response) throws
ServletException, IOException
{
    System.out.println("Provisional response: " + a_Response);
}

public void doSuccessResponse(SipServletResponse a_Response) throws
ServletException, IOException
{
    System.out.println("Success response: " + a_Response);
}
```

必須重申的是，上述程式碼會導致 Servlet 透過 doResponse() 方法接收回應，此方法通常會將回應傳回給啓動 UA。階段作業中的後續要求也會呼叫 Servlet 的 service() 方法，例如任一 UA 送出 BYE 要求以終止階段作業。

Back to top

啓動及終止行爲

實作終止 UA 是 SIP Servlet 最簡單的功能，在此情況下，Servlet 會直接執行一些處理，然後再將最終回應傳給該要求，如清單 6 所示：

清單 6. 簡單的 Servlet

```
public void doInvite(SipServletRequest a_Request) throws ServletException,
IOException
{
    // Required by the Ubiquity Container to establish session state.
    SipApplicationSession appSession = a_Request.getApplicationSession();
    SipSession sipSession = a_Request.getSession();

    // Act as a terminating user agent and return a 403 Forbidden response to the request.
    SipServletResponse response =
a_Request.createResponse(SipServletResponse.SC_FORBIDDEN);
    response.send();
}
```

此外，SIP Servlet API 可將 Servlet 當作啓動 UA；即初始及後續要求，B2BUA 通常會使用這種功能。在此情況下，Servlet 可使用 SipFactory 建立新的 SipAddress 和 SipServletRequest 以便傳送，如清單 7 所示：

清單 7. 建立 UAC 要求

```
public void doInvite(SipServletRequest a_Request) throws ServletException,
IOException
{
    // Required by the Ubiquity Container to establish session state.
    SipApplicationSession appSession = a_Request.getApplicationSession();
    SipSession sipSession = a_Request.getSession();
```

```
    // Create a new request and proxy it to a new destination.

    ServletContext sc = getServletContext();

    SipFactory factory =
(SipFactory)sc.getAttribute("javax.servlet.sip.SipFactory");

    Address to = factory.createAddress("sip:callee@company.net:5060");
    Address from = a_Request.getFrom();

    SipServletRequest newRequest = factory.createRequest(appSession, "INVITE", from,
to);

    Proxy p = a_Request.getProxy(true);

    p.setSupervised(true);
    p.setRecordRoute(true);

    System.out.println("Proxying request: " + a_Request.getRequestURI());

    p.proxyTo(newRequest.getRequestURI());
}
```

SIP Servlet API 目前無法讓您建立初始要求以回應非 SIP 事件。這個問題已受到重視，因此若要建立新要求，必須從現有要求取得 SipApplicationSession。SIP Servlet 社群解決此問題之前，SIP A/S 廠商必須實作本身的功能。

Back to top

B2BUA 的角色

Back-to-Back User Agent 在 SIP 期間可當作仲介，方法是接收所有要求，然後轉遞給受話端，也會接收所有回應，再將其傳回受話端。基本上，會有兩個對話方塊，而不是兩個 UA 之間的一個對話方塊：一個是出現在發話端 UA 與 B2BUA 之間，另一個是在 B2BUA 與收接者 UA 之間，

毫無疑問地，這種行爲可能會中斷服務。爲了降低風險，請確定將送入要求的所有不明標頭複製到送出要求。

B2BUA 的另一個問題是他們必須保留的階段作業，以維護對話方塊。一旦發生 SIP A/S 失效，這種階段作業就會遺失，也不會再處理呼叫。

儘管有這些問題，B2BUA 確實在許多情況下都很有用，如實作計費應用程式、第三方呼叫控制及防火牆控制等。

Back to top

SIP：所謂的未來其實已經開始

此 SIP 簡介只談到最基本的 SIP 概念及 SIP Servlet API 可達到的功能。事實上，爲了使本文更加簡潔，我已略過數個重要主題，包括迴圈、持續增加或減少及 Servlet 應用程式組合。

不過，本文應該已清楚說明 SIP 是個相對簡單的通訊協定。同時，SIP Servlet API 可提供進階擷取，簡化開發人員的工作。雖然 SIP Servlet 模式一直改變，但確實提供了卓越的服務建立環境。隨著 SIP 的出現，電信應用程式的開發工作比以往更加輕鬆。

資源

- 請按一下這裡，取得本文的程式碼範例。
- 如需其他程式碼，請取出完整的 SIP 通話日誌。
- Java Community Process 有 JCP SIP Servlet API 的相關資訊。
- SIP 中心可提供開發商業 SIP 的入口網站。
- SIP RFC 3261 包含 SIP 的相關資訊。

關於作者

Nick Williams 是 Ubiquity Software Corporation 的開發人員，向來從事兩項領導業界的 SIP 產品：SIP Application Server 及 Speak C Director。