**SOAP**

SOAP 是在分散的分佈式環境中交換結構化資訊時應遵循的一種規範。因此，它表示在面向服務的體系結構 (SOA) 中三個主要角色：服務提供程式、服務請求程式和服務代理程式之間的主要通信方法。其設計的主要目標是簡單和可擴展。SOAP 消息用於請求 Web Service。

**SOAP 1.1**

WebSphere® Application Server 遵循 SOAP 1.1 中闡述的標準。

包括 IBM® 和 Lotus® 在內的幾個公司已將 SOAP 作為可擴展標記語言 (XML) 協議工作組的基礎提交至萬維網聯盟 (W3C)。 此協議由三部分組成：

* 一個*封包*，定義用於描述消息內容並處理指示資訊的框架。
* 一組*編碼規則*，用於表達應用程式定義的數據類型的實例。
* 一個*常規*，用於表示遠程過程調用和響應。

SOAP 1.1 是獨立於協議的傳輸，可與各種協議組合使用。在用 WebSphere Application Server 開發和實現的 Web Service 中，SOAP 會用在 HTTP、HTTP 擴展框架和 Java 消息服務 (JMS) 的組合中。SOAP 也獨立於操作系統，不會捆綁到任何編程語言或組件技術。

只要客戶機可以發出 XML 消息，它就不在乎使用哪種技術來實現該客戶機。同樣，只要服務可以處理 SOAP 消息，就可以使用任何語言實現它。而且服務器和客戶機都可駐留在任何合適的平臺上。

**SOAP 1.2**

SOAP 1.2 規範也是 W3C 建議，WebSphere Application Server 遵循 SOAP 1.2 中規定的標準。SOAP 1.2 規範包括三個部分和一些聲明以及一個測試集合：

* Part 0：入門
* Part 1：消息傳遞框架
* Part 2：附件
* 規範聲明和測試集合

SOAP 1.2 更明確地定義了 SOAP 處理模型，該模型中移除了許多在缺少 Web Service 互通性 (WS-I) 概要文件的情況下有時會導致互通性問題的二義性。SOAP 1.2 應可降低不同供應商之間 SOAP 1.2 實現發生互通性問題的機率。

SOAP 1.2 規範中一些較重要的更改包括：

* 現在只要供應商遵從 SOAP 1.2 中定義的綁定框架，就能夠正式定義 HTTP 協議以外的其他傳輸協議。雖然 HTTP 普遍存在，但它沒有其他傳輸協議（如 TCP/IP 和 MQ 等等）可靠。
* SOAP 1.2 基於 XML 資訊集（XML 資訊集）。資訊集提供了一種方法來使用 XSD 模式描述 XML 文檔，但並不一定使用 XML 1.0 序列化來序列化該文檔。SOAP 1.1 基於 XML 1.0 序列化。資訊集將使得使用其他序列化格式（例如二進制協議格式）更容易。可以使用二進制協議來將消息壓縮成某一可能不需要某些詳細標記資訊的壓縮度更高的格式。

Java API for XML Web Services (JAX-WS) 標準引入同時支援 SOAP 1.1 和 SOAP 1.2 的功能。

**帶附件的 SOAP Java API 介面**

*帶附件的 SOAP Java API* (SAAJ) 介面用於 SOAP 消息傳遞，它提供從 Java 程式設計模型通過網際網路發送 XML 文檔的標準方法。SAAJ 用於在 SOAP 消息流經運行時環境時將它處理到適當的上下文上。

**最佳實踐：** IBM® WebSphere® Application Server supports the Java API for XML-Based Web Services (JAX-WS) programming model and the Java API for XML-based RPC (JAX-RPC) programming model. JAX-WS is the next generation web services programming model extending the foundation provided by the JAX-RPC programming model. Using the strategic JAX-WS programming model, development of web services and clients is simplified through support of a standards-based annotations model. Although the JAX-RPC programming model and applications are still supported, take advantage of the easy-to-implement JAX-WS programming model to develop new web services applications and clients.

Java API for XML-Based RPC (JAX-RPC) 程式設計模型支援 SAAJ 1.2 並且可以處理 XML。

JAX-WS 程式設計模型支援 SAAJ 1.2 和 1.3。SAAJ 1.3 包含 SOAP 1.2 消息的支援。

SAAJ 1.2 與 SAAJ 1.3 規範之間的差別可在主題“SAAJ 版本之間的差別”中查看。

**如何在 Web Service 中使用消息？**

Web Service 使用 XML 技術來交換消息。這些消息符合 XML 模式。開發 Web Service 應用程式時，只有有限的 XML API 可供使用，例如文檔物件模型 (DOM)。在運行時期間處理 Java 物件並完成序列化和反序列化將更有效。

Web Service 使用 SOAP 消息表示客戶機和伺服器之間的遠端程序呼叫。SOAP 消息通常被反序列化成一系列 Java 值/類型業務物件，這些業務物件表示參數和返回值。另外，Java 程式設計模型提供支援應用程式和處理常式的 API 以直接處理 SOAP 消息。因為該程式設計模型僅支援有限數目的 XML 模式類型，所以，該規範提供 SAAJ 資料模型作為擴展用來處理消息。

要處理 XML 模式類型，需要使用*定制資料綁定程式*將 XML 模式類型映射至 Java 類型。

**SAAJ 介面**

與 SAAJ 相關的類位於 javax.xml.soap 包中。SAAJ 在介面和抽象類別的基礎上構建，並且許多類從調用工廠方法來創建工廠（如 SOAPConnectionFactory 和 SOAPFactory）開始。

**避免故障：** 如果啟用了 Java 安全性，並且未授予讀取 jaxm.properties 檔的許可權，那麼在通過調用 javax.xml.soap.SOAPFactory.newInstance() 來創建 SOAPFactory 實例時，或者通過調用 MessageFactory.newInstance() 來創建 MessageFactory 實例時，會發生 SecurityException 異常，並且會將以下異常寫入系統日誌中：

Permission:

/opt/IBM/WebSphere/AppServer/java/jre/lib/jaxm.properties : access denied

(java.io.FilePermission /opt/IBM/WebSphere/AppServer/java/jre/lib/jaxm.properties

read)

Code:

com.ibm.ws.wsfvt.test.binding.addr1.binder.AddressBinder

in {file:/opt/IBM/WebSphere/AppServer/profiles/AppSrv01/installedApps/

ahp6405Node01Cell/DataBinding.ear/address1.war/WEB-INF/lib

/addressbinder1.jar}

Stack Trace:

java.security.AccessControlException: access denied (java.io.FilePermission

/opt/IBM/WebSphere/AppServer/java/jre/lib/jaxm.properties read)

.

SOAPFactory 會忽略異常，並繼續確定要裝入的實現的下一種方法。因此，可以忽略此安全性異常的日誌條目。

因為本產品使用 SOAPFactory 來支援其他 Web Service 技術（例如 WS-Addressing (WS-A)、WS-Atomic Transaction (WS-AT) 和 WS-Notification），所以在啟用了 Java 安全性的任何應用程式中，您可以忽略此 SecurityException。

最常用的類有：

* SOAPMessage：包含消息，包括 XML 和非 XML 部分
* SOAPHeader：表示 SOAP 頭 XML 元素
* SOAPBody：表示 SOAP 主體 XML 元素
* SOAPElement：表示 SOAP 消息中的其他元素

SAAJ 介面的其他部分包括：

* MessageContext：包含 SOAP 消息和相關的屬性
* AttachmentPart：表示二進位附件
* SOAPPart：表示消息的 XML 部分
* SOAPEnvelope：表示 SOAP 包絡 XML 元素
* SOAPFault：表示 SOAP 故障 XML 元素

SAAJ 模型中的主要介面是 javax.xml.soap.SOAPElement（也稱為 SOAPElement）。通過使用此模型，應用程式可以處理使用預先存在的 DOM 代碼的 SAAJ 模型。將預存在的 DOM 物件轉換為 SAAJ 物件也很容易。

使用 SAAJ 介面創建的消息遵循 SOAP 標準。SOAP 消息在 SAAJ 模型中表示為 javax.xml.soap.SOAPMessage 對象。消息的 XML 內容由 javax.xml.soap.SOAPPart 物件表示。每個 SOAP 部件有一個 SOAP 包絡。此封包由 SAAJ javax.xml.SOAPEnvelope 物件表示。SOAP 規範定義駐留在 SOAP 包絡中的各種元素；SAAJ 為 SOAP 包絡中的各種元素定義物件。

SOAP 消息還可以包含稱為附件的非 XML 資料。這些附件由可從 SOAPMessage 物件訪問的 SAAJ AttachmentPart 物件表示。

關於處理常式和應用程式為何使用一般 SOAPElement API 而不是緊綁定的映射，存在許多原因：

* 一個 Web Service 可能是另一個 Web Service 的同步管道。在這種情況下，僅轉發 SOAP 消息。
* Web Service 可使用不同的資料模型（例如服務資料物件 (SDO)）來操控消息。將消息從 SAAJ DOM 轉換為另一種資料模型是更容易的。
* 處理常式（例如，數位簽章驗證處理常式）可能要對消息進行一般處理。

由於 SOAPElement 介面並非始終都是舊系統的最佳替代者，因此，可能需要進一步映射 XML 模式類型。在這種情況下，您可能需要使用一個通用程式設計模型（例如，SDO），該模型更適合於以資料為中心的應用程式。

可以將 XML 模式配置為包含定制資料綁定，該資料綁定用來使 SDO 或資料物件與 Java 物件配對。例如，運行時將入局 SOAP 消息呈示到 SOAPElement 介面中，並將它傳遞給客戶資料綁定程式以進行更多處理。如果入局消息包含 SDO，那麼運行時識別資料物件代碼，查詢它的類型映射以查找定制綁定程式，並構建用來表示 SDO 代碼的 SOAPElement 介面。SOAPElement 被傳遞給 SDOCustomBinder。

請參閱有關定制資料綁定程式的資訊，以瞭解有關使用 SOAPElement、SDO 和定制綁定程式來開發應用程式的過程的更多資訊。

**SAAJ 版本之間的差別**

帶附件的 SOAP Java API (SAAJ) 介面 V1.3 擴展了對 Web Service 環境下 SOAP 1.2 消息的支援。SAAJ 1.2 與 SAAJ 1.3 之間存在一些差別，本主題將進行討論。

在典型的 Web Service 環境中，用戶依賴基於 Java 標準的底層代碼來轉換一組 Java 物件。 SAAJ 提供了 API 來讀、寫、發送和接收 SOAP 消息，並支援以 SOAP 消息的附件形式發送的二進位內容。

SAAJ 1.3 進行了調整已適合 SOAP 1.1 和 SOAP 1.2 消息並且受到 Java API for XML Web Services (JAX-WS) 程式設計模型和 Java API for XML-Based RPC (JAX-RPC) 程式設計模型的支援。SAAJ 1.2 只能與 SOAP 1.1 消息配合使用。

如果將代碼從 SOAP 1.1 遷移至 SOAP 1.2，若消息是 SOAP 1.2 消息，那麼您可以繼續使用現有的 SOAP 1.1 代碼。如果升級基本代碼以使用 SAAJ 1.3，那麼可以繼續使用處理 SOAP 1.1 消息的現有代碼。這些差別的示例是在 SOAP 1.1 中，故障的符合人們閱讀習慣的文本存儲在 faultString 元素中。在 SOAP 1.2 中，符合人們閱讀習慣的文本存儲在 Reason 元素中。您的代碼看上去可能類似於以下示例：

String text = soapFault.getFaultString();

如果消息基於 SOAP 1.1，那麼 getFaultString () 返回 faultString 值。如果使用的是 SOAP 1.2，那麼 getFaultString () 返回 Reason 值。另外，SAAJ 1.3 介面提供了一個新的方法，即 getReasonText (Locale)，它獲取特定 Reason 值。如果消息基於 SOAP 1.1，那麼 getReasonText (Locale) 方法返回說明的異常。SAAJ 1.3 介面支援現有代碼處理 SOAP 1.1 和 SOAP 1.2 消息。

SAAJ 1.2 與 SAAJ 1.3 之間的其他差別在以下列表中列示：

* SAAJMetaFactory 介面

引入 SAAJMetaFactory SPI 以支持在單個位置中創建 SOAP 工廠類。

* SAAJResult 類

SAAJResult 物件充當 SAAJ 樹中 Java API for XML Processing (JAX-P) 變換或 Java XML 綁定體系結構 (JAXB) 編組的結果的預留位置。引入 SAAJResult 類以提高當期望變換結果是有效的 SAAJ 樹時的可用性。

* 接受 QName 而不是 Name 的超負載方法

首選用 QName 表示 XML 限定名。因此，在所有 SAAJ API 中引入了超負載方法，其中相應的方法接受 javax.xml.soap.Name 名稱作為參數。

* AttachmentPart、SOAPBody 以及 SOAPElement 介面和類中的新方法

使用這些新方法來在使用新的 SOAP 功能部件時提供幫助。

* SOAPPart 現在是 javax.xml.soap.Node 方法。

SOAPPart 物件也看作 SOAP 節點方法。

**消息傳輸優化機制**

SOAP 消息傳輸優化機制 (MTOM) 是萬維網聯盟 (W3C) 開發的標準。MTOM 描述一個機制，該機制用於通過有選擇地重新編碼部分消息但仍向 SOAP 應用程式顯示 XML 資訊集來優化 SOAP 消息的傳輸或有線格式。

可能有很多原因會使您想隨 Web Service 請求發送二進位附件，例如圖像或檔。可通過幾種方法來實現此目的，例如：

* 用 SOAP 有效內容中的基本 64 位元直接插入進行編碼。然而，編碼直接插入會增大 SOAP 消息的大小。請注意，基本 64 位元編碼可能是二進位資料的兩倍大。
* 使用 SOAP with Attachments (SwA) 並遵循 Web Services Interoperability Organization (WS-I) Attachments Profile 對消息進行編碼。WebSphere® Application Server 當前支持此方法。
* 通過使用 XML 二進位優化打包 (XOP) 提供二進位消息傳輸的優化。優化僅可用於二進位資料或內容。MTOM 在 SOAP 和 MIME over HTTP 的上下文中使用 XOP。

XOP 定義了一種序列化機制，該機制用於具有不適合 SOAP 和 MIME 打包、但適合任何 XML 資訊集和任何打包機制的二進位內容的 XML 資訊集。這是 XML 的替代序列化，只是看上去像 MIME 多個部件/相關包，將 XML 文檔用作根部件。除了基本 64 位元編碼的資料替換為對非基本 64 位元編碼的其中一個 MIME 部件的引用之外，該根部件與文檔的 XML 序列化非常相似。此引用使您能夠在與編碼相關聯的處理方面節省大量開銷。編碼是可將二進位資料直接嵌入 XML 世界的唯一方法。

如果禁用 MTOM 映射生成，就會禁用 XOP。如果禁用 XOP，就不能通過使用 MIME 附件發送二進位資料。而是像平時一樣使用基本 64 位元對二進位資料進行編碼。

MTOM 規範以三個不同的部件的形式定義：

* SOAP 消息的優化傳輸或有線格式的抽象功能部件。因為在發送方、接收方和中間方的優化技術的描述和 SOAP 處理器的行為是通用的，並且不包括對 MIME、HTTP 等等的技術的引用；在這個意義上，此功能部件是抽象的。優化技術在有選擇地對可作為 xs:base64Binary 資料類型的規範文字標記法訪問的 SOAP 包絡資訊集的某些內容進行編碼的同時，集中於確保 SOAP 處理器的 SOAP 包絡資訊集視圖。

實現這些抽象功能部件需要兩個方面的具體規範：優化的有線格式以及優化的有線格式在特定傳輸上如何流動。

* MTOM 規範的第二個部件解決序列化方面並標準地取決於 MIME 多個部件/相關 XOP 打包。該序列化方面就是 MTOM 與 XOP 相關之處。
* 作為具體 SOAP HTTP 綁定級別功能部件，MTOM 擴展序列化。此部件描述可如何使用 HTTP 綁定來傳輸存放 SOAP MTOM 消息的 XOP 包。此部件還對可能發生的將 SOAP MTOM 消息序列化為 XOP 包施加了一些限制，例如只能使用 XML 1.0。

Java API for XML Web Services (JAX-WS) 添加了對使用 MTOM 發送二進位資料附件的支援。JAX-WS 是新近重新設計的用於 Web Service 的 API 堆疊（包括 JAX-WS 2.0、JAXB 2.0 和 SAAJ 1.3）的中心部件。API 堆疊有時候稱為集成堆疊。JAX-WS 用於替換 Web Service 和 Web 應用程式中的 JAX-RPC。

**附加方法**

通過值或通過引用附加已成為用於處理 XML 格式化消息中的不透明資料的廣泛接受的技術。

* **通過值**附加：在通過使用基本 64 位元或十六進位文本編碼方法（在 XML 模式中分別以資料類型 xs:base64Binary 和 xs:hexBinary 編碼）以元素或屬性的方式嵌入不透明資料內容時使用該技術。
* **通過引用**附加：在通過使用 URI（在 XML 模式中以資料類型 xs:anyURI 編碼）以元素或屬性的方式在外部引用不透明資料內容時使用該技術。

使用這兩種技術中的任一種都有優點和缺點。MTOM 規範著重於解決這些固有的附加問題。

另一種標準由萬維網聯盟 (W3C) 定義，稱為 SOAP with Attachments (SwA)。SwA 開發為打包帶附件的 SOAP 消息的方法。由於一些供應商不支持 SwA，可以用功能更強大的 MTOM 和 XOP 機制代替 SwA。SwA 與 MTOM 在概念上非常相似，都是將二進位資料編碼為 MIME 文檔中的 MIME 附件。使用 MIME 附件可提高傳輸大量二進位有效內容的性能。

SwA 與 MTOM 的其他差別包括：

* MTOM 使用稱為 XOP 的標準，該標準定義存在於 SOAP 有效內容中的 XOP 引用。此引用指向包含二進位資料的 MIME 附件。
* 對於 MTOM，XOP 引用在邏輯上將二進位資料包括到 XML 資訊集中。對於 SwA，href 指向不僅物理上在 XML 文檔外部，邏輯上也未包括在其資訊集中的資料。
* 對於 MTOM，可對二進位附件進行邏輯簽署，就如它們是 SOAP XML 文檔的一部分一樣。
* 除了 IBM® 之外，Microsoft® .NET 也支持 MTOM, 它消除了使用 SwA 時存在的一些互通性問題。當協作提交者討論建議的修改時，將互通性看作主要目標。

MTOM 附加方法利用 SOAP 基礎結構，同時也可以獲得 SOAP with Attachment (SwA) 解決方案提供的傳輸效率。

**SOAP 1.2 和 SOAP 1.1**

SOAP 1.1 基於 XML 規範。SOAP 1.1 實現可能仍會存在幾年。對於仍然運行 SOAP 1.1 的那些用戶而言，現在有了將 MTOM 用於附件支援的可交互操作方法。SAP、Oracle、Microsoft 和 IBM 已經已向 W3C 提交了用於 MTOM 1.0 的 SOAP 1.1 綁定規範，該規範定義了 MTOM 如何可以與 SOAP 1.1 有效內容配合使用。該規範詳細說明了對 SOAP MTOM 和 XOP 規範進行的必要修改，而這些修改是將這些技術與 SOAP 1.1 成功配合使用所必需的。請參閱該規範以瞭解更多詳細資訊。

MTOM 是 SOAP V1.2 功能部件，它基於資訊集。請參閱 XML 資訊設置文檔以瞭解更多資訊。

**示例**

沒有 MTOM，無論用什麼格式編碼的資料都會用模式（基本 64 位或十六進位）描述，然後包含在 XML 文檔中。以下示例顯示了具有 <xsd:base64Binary> 元素的 SOAP 消息：

... other transport headers ...

Content-Type: text/xml; charset=UTF-8

<?xml version="1.0" encoding="UTF-8"?>

<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">

<soapenv:Header/>

<soapenv:Body>

<sendImage xmlns="http://org/apache/axis2/jaxws/sample/mtom">

<input>

<imageData>R0lGODl ... more base64 encoded data ... KTJk8giAAA7</imageData>

</input>

</sendImage>

</soapenv:Body>

</soapenv:Envelope>

如果啟用了 MTOM，表示附件的二進位資料就會作為 MIME 附件包括到 SOAP 消息中。以下示例顯示具有附件資料的支援 MTOM 的 SOAP 消息：

... other transport headers ...

Content-Type: multipart/related; boundary=MIMEBoundaryurn\_uuid\_0FE43E4D025F0BF3DC11582467646812;

type="application/xop+xml"; start="<0.urn:uuid:0FE43E4D025F0BF3DC11582467646813@apache.org>";

start-info="text/xml"; charset=UTF-8

--MIMEBoundaryurn\_uuid\_0FE43E4D025F0BF3DC11582467646812

content-type: application/xop+xml; charset=UTF-8; type="text/xml";

content-transfer-encoding: binary

content-id:

<0.urn:uuid:0FE43E4D025F0BF3DC11582467646813@apache.org>

<?xml version="1.0" encoding="UTF-8"?>

<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">

<soapenv:Header/>

<soapenv:Body>

<sendImage xmlns="http://org/apache/axis2/jaxws/sample/mtom">

<input>

<imageData>

<xop:Include xmlns:xop="http://www.w3.org/2004/08/xop/include"

href="cid:1.urn:uuid:0FE43E4D025F0BF3DC11582467646811@apache.org"/>

</imageData>

</input>

</sendImage>

</soapenv:Body>

</soapenv:Envelope>

--MIMEBoundaryurn\_uuid\_0FE43E4D025F0BF3DC11582467646812

content-type: text/plain

content-transfer-encoding: binary

content-id:

<1.urn:uuid:0FE43E4D025F0BF3DC11582467646811@apache.org>

... binary data goes here ...

--MIMEBoundaryurn\_uuid\_0FE43E4D025F0BF3DC11582467646812--

**XML 二進位優化打包**

XML 二進位優化打包 (XOP) 規範由萬維網聯盟 (W3C) 在 2005 年 1 月 25 日標準化。SOAP 消息傳輸優化機制 (MTOM) 在 SOAP 和 MIME over HTTP 的上下文中使用 XOP。

XML 廣泛用於資料傳輸。XML 是用於交換格式良好的文檔的流行格式，這是因為它是純文字，符合人類閱讀習慣且是結構化的。例如，Web Service 中的 SOAP 消息傳遞基於 XML（或基於與 SOAP 1.2 配合使用的 XML 資訊集）。人們希望既可利用舊的格式，如 PDF、GIF、JPEG 和類似格式，又可以繼續使用 XML 模型。將 XML 與預先存在的資料格式集成是使用者長久以來的願望，也是 XML community 仍然存在的問題。使用者常常希望既可利用 XML 的結構化可擴展標記約定又不用放棄不符合 XML 1.0 語法的現有資料格式。

隨著 Web Service 中的 SOAP 消息傳遞越來越普遍，下一步是如何將基於非文本的資料（例如圖像和工作流程資料）與消息一起發送。例如，您可能希望在兩個應用程式之間發送 .jpeg 格式的掃描文檔。問題是此資料在各應用程式之間是否能識別。

XML 和 Web Service 的值大都在於使用通用 XML 工具來處理內容的能力。描述和處理 XML 的許多 XML 工具和標準（例如解析器、XPath、XQuery、XSLT、XML 加密、數位簽章和 XML 模式）並非旨在使用非文本資料（例如圖像）。這些 XML 工具不會處理非 XML 內容；這些工具需要文本。難題就是如何用 XML 嵌入或附加非文本的資料（也稱為*二進位資料*）。換句話說，需要將二進位檔案附加至 SOAP 消息的方法。

編碼是可將二進位資料直接嵌入 XML 世界的唯一方法。通常使用基本 64 位元對二進位資料進行編碼來將它作為文本嵌入到 XML 文檔中。基本 64 位是已存在一定時間的一種序列化方法，可現成輕鬆實現，並且具有平臺間互通性。xsi:base64binary 資料類型在 XML 模式中支援此序列化。基本 64 位元將二進位資料編碼為可插入 XML 文檔中的文本標記法。基本 64 位取用您的二進位資料並通過一次編碼三個八位元元來將它轉換為一系列 ASCII 字元。由於每個八位元包括八位，且這八位將它們表示為 ASCII 標準中的四個可列印字元，因此它使用 64 個 ASCII 字元來表示二進位資料。所有平臺都可使用此約定進行解碼和編碼。6 位元 ASCII 廣受支援，不需要處理任何特殊字元。但是，對於較大型消息存在性能影響。

對於需要快速操作的應用程式，基本 64 位可能不適用。如果想要索引到此類內容，對它進行查詢、變換、加密、簽署或描述，那麼您需要使用其他機制。

已開發第一個附件規範，稱為 SOAP with Attachments (SwA)。SwA 的基本概念是將二進位消息部分（附件）看作多用途網際網路郵件擴展 (MIME) 附件。MIME 是用於格式化非 ASCII 郵件消息附件的廣泛實現的規範。SwA 指定只需使用 URI，SOAP 正文就可包含對 MIME 消息部件（附件）的引用。二進位部件通過引用附加。

SwA 具有以下一些缺點：

* SwA 在可使用性與互通性方面存在不足。SOAP 基礎結構是沿 SOAP 包絡創建的，而 SOAP 包絡不善於處理附件。使用 SwA 的附件意味著在一條消息中使用兩個資料模型。這兩種資料模型不能用現有 XML 技術操作。
* SwA 不能處理 SOAP 的可複合字元。WS-Security 之類的標準本質上未編寫為可處理附件。WS-Security 需要處理需要數位簽章或加密的所有資料，這意味著所有資料也都在附件中。但如果 WS-Security 不能訪問資料，那麼 WS-Security 就不能進行處理，簽名也變得無效。

用戶通常希望讓其現有非 XML 格式保持不變，XML 工具和基礎結構將其看作八位元元的不透明順序。這樣就允許 XML 內可以同時存在各種常用格式，例如 .jpeg 和 .wav 格式。XOP 使得使用基本 64 位元編碼資料更為現實。現在，XOP 只允許對基本 64 位元編碼的資料進行優化。

使用 XOP 意味著 XML 類型 base64Binary 的實例（如果啟用）將通過使用 MIME 附件進行傳輸。如果正在使用 XOP，那麼該實現可自動對它進行編碼。XOP 維護 XML 消息的資料模型，這是因為附件被看作基本 64 位元編碼資料。如果 XML 堆疊瞭解 XOP 編碼，那麼根本就不需要更改您的應用程式。例如，當它想要訪問 .jpeg 圖片時，它可以將內容的字元值獲取為基本 64 位元編碼字串。

XOP 為人們提供了考慮在資料交換中使用 MIME 消息的方法，這些消息適合並且已用於許多其他資料。XOP 格式使用多重部件 MIME 將原始二進位資料包含到 XML 1.0 文檔中，無需基本 64 位元的編碼。

然後，結合使用的規範（即 SOAP 消息傳輸優化方法，MTOM）指定如何將此格式綁定到 SOAP。XOP 和 MTOM 標準應增強 SOAP 1.2 性能。XOP 與 MTOM 一起提供將二進位資料與基於文本的 XML 混合在一起的首選方法。 通過 MTOM 和 XOP 二者結合，我們可以選擇消息的哪些部分需要作為二進位連線發送，同時仍保留資訊集。這些標準使包含 SOAP 包絡外部二進位資料的附件可以作為消息部分。 但是，與 SwA 不同，這樣會將二進位資料看作就在 SOAP 包絡中，即只有一個資訊集。

XOP 為具有不適合 SOAP 和 MIME 打包、但適合任何 XML 資訊集和任何打包機制的二進位內容的 XML 資訊集定義序列化機制。另一方面，XML 不是很好的通用打包機制。

通過將一連串 XML 資訊集置於可擴展打包格式（例如 MIME）中來創建 XOP 包。注意，XOP 不會將 MIME 複用於線上的實際打包。然後，將抽取並重新編碼其內容的選定部分（為基本 64 位元編碼二進位資料），表示根據基本 64 位元對資料進行解碼並放入包中。那些選定部分的位置在 XML 中用通過使用 URI 連結至所打包資料的特殊元素標記。

SOAP 處理引擎就在消息命中線路之前執行二進位資料的臨時基本 64 位元編碼。通過這種臨時編碼，SOAP 處理器可以處理基本 64 位元資料；例如，可以獲取資料的 WS-Signature 並將其放置到頭中。無須在另一端進行多餘的解碼操作，該過程以相反的方向工作。

可以在 Java (JAX-WS) 中使用 MTOM 和 XOP 實現。

**示例**

以下示例顯示進行 XOP 處理 (SOAP) 之前的 XML 資訊集：

<soap:Envelope

xmlns:soap='http://www.w3.org/2003/05/soap-envelope'

xmlns:xmlmime='http://www.w3.org/2004/11/xmlmime'>

<soap:Body>

<m:data xmlns:m='http://example.org/stuff'>

<m:photo xmlmime:contentType='image/png'>/aWKKapGGyQ=</m:photo>

<m:sig xmlmime:contentType='application/pkcs7-signature'>Faa7vROi2VQ=</m:sig>

</m:data>

</soap:Body>

</soap:Envelope>

**示例**

以下示例顯示序列化為 XOP 包 (SOAP) 的 XML 資訊集

MIME-Version: 1.0

Content-Type: Multipart/Related;boundary=MIME\_boundary;

type="application/xop+xml";

start="<mymessage.xml@example.org>";

startinfo="application/soap+xml; action=\"ProcessData\""

Content-Description: A SOAP message with my pic and sig in it

--MIME\_boundary

Content-Type: application/xop+xml;

charset=UTF-8;

type="application/soap+xml; action=\"ProcessData\""

Content-Transfer-Encoding: 8bit

Content-ID: <mymessage.xml@example.org>

<soap:Envelope

xmlns:soap='http://www.w3.org/2003/05/soap-envelope'

xmlns:xmlmime='http://www.w3.org/2004/11/xmlmime'>

<soap:Body>

<m:data xmlns:m='http://example.org/stuff'>

<m:photo

xmlmime:contentType='image/png'><xop:Include

xmlns:xop='http://www.w3.org/2004/08/xop/include'

href='cid:http://example.org/me.png'/></m:photo>

<m:sig

xmlmime:contentType='application/pkcs7-signature'><xop:Include

xmlns:xop='http://www.w3.org/2004/08/xop/include'

href='cid:http://example.org/my.hsh'/></m:sig>

</m:data>

</soap:Body>

</soap:Envelope>

--MIME\_boundary

Content-Type: image/png

Content-Transfer-Encoding: binary

Content-ID: <http://example.org/me.png>

// binary octets for png

--MIME\_boundary

Content-Type: application/pkcs7-signature

Content-Transfer-Encoding: binary

Content-ID: <http://example.org/my.hsh>

// binary octets for signature

--MIME\_boundary--

**XML 信息集**

XML 信息集是萬維網聯盟 (W3C) 在 2004 年 2 月 4 日發佈的規範。XML 資訊集是存儲在 XML 文檔中的資訊的抽象模型。資訊集以適合 XML 最習慣使用的方式建立資料與資訊之間的分隔。通過引用 XML 資訊集專案及其屬性，定義了一些具體的 XML 資料模型。

鑒於 *XML 資訊集*是存儲在 XML 文檔中的資訊的抽象模型，*資訊專案*就是 XML 文檔的一些組件的抽象表示。SOAP V1.2 使用此抽象來定義 SOAP 消息中的資訊，而甚至沒有參考 XML V1.*x*。SOAP HTTP 綁定明確地允許使用替代介質類型，這些介質類型至少可以傳輸 SOAP XML 資訊集。

SOAP 消息傳輸優化機制 (MTOM) 根據資訊專案描述 SOAP 1.2 構造，而 SOAP 1.1 根據 XML 元素進行定義。通過有選擇地編碼部分消息但仍向 SOAP 應用程式顯示 XML 資訊集，MTOM 使 SOAP 綁定能夠優化 SOAP 消息的傳輸和/或有線格式。SOAP 1.2 屬性現在處於 SOAP 名稱空間中。XML 資訊集需要 XML 名稱空間支援。核心 XML 建議不需要 XML 名稱空間支援；但是，名稱空間是支援 XML 模式所必需的。

XML 資訊集不需要或不偏愛特定介面或某類介面。為了簡單明瞭起見，XML 資訊集規範將資訊集顯示為樹，但不要求將 XML 資訊集以樹結構的形式提供。其他類型的介面（包括但不限於基於事件和基於查詢的介面）也可以提供符合資訊集的資訊。只要資訊集中的資訊能夠以某種方式可用于 XML 應用程式，就滿足了 XML 資訊集的要求。

XML 資訊集提供了一組定義，以在引用格式良好的 XML 文檔中資訊的其他規範中使用。對於任何給定 XML 文檔，有許多相應的資訊集。

* 唯一的最小資訊集僅包含核心專案的核心屬性，不包含其他內容。
* 唯一的最大資訊集包含所有核心專案和所有週邊專案以及所有週邊屬性，且存在和不存在的週邊項及其之間屬性的每個組合都對應一個唯一的最大資訊集。當中包含的資訊集必須與最大資訊集完全一致。

**資訊集專案**

XML 資訊集是在格式良好的 XML 文檔中可用的資訊的描述，它以根據一組資訊集專案描述 XML 文檔的抽象資料模型。資訊專案是 XML 文檔中某些部分的抽象描述，而每個資訊專案都包含一組相關聯的指定屬性。所有其他資訊專案都可直接或間接地通過其他資訊專案的屬性從文檔資訊專案的屬性訪問。

使用資訊集專案的準則包括：

* 不要求 XML 文檔必須有效才能包含資訊集。
* XML 文檔如果滿足名稱空間約束就可包含資訊集。
* XML 文檔如果格式良好就可以包含資訊集。
* 資訊集中只允許使用一個文檔資訊專案。
* XML 文檔的資訊集由兩個或多個資訊專案組成。
* 任何格式良好的 XML 文檔的資訊集至少包含以下最少的資訊專案：一個文檔資訊專案和一個元素資訊專案。
* 每個資訊專案都包含一組相關聯的屬性，部分屬性是核心屬性，部分屬性是週邊屬性。

資訊集可包含多達 11 不同類型的資訊專案：

* 文檔資訊專案
* 元素資訊專案
* 屬性資訊專案
* 處理指示資訊專案
* 未擴展的實體引用資訊專案
* 字元資訊專案
* 注釋資訊專案
* 文檔型別宣告 (DTD) 資訊專案
* 未解析的實體資訊專案
* 標記法資訊專案
* 名稱空間資訊專案

請注意，XML 文檔的資訊集可能不是所有資訊專案的完整清單。

某些種類的無效性會影響指定給某些屬性的值。可以取消對實體、標記法、元素和屬性的聲明。可以多次聲明標記法和元素。多次聲明對於實體和屬性有效。可能未定義或多次定義標識。在 XML 資訊集規範中與資訊專案定義相關的部分中記錄了此類情況。

**語法**

XML 資訊集使用方括號語法，表示屬性名稱顯示在方括號中。例如，文檔資訊專案具有以下屬性：

| *表 1. XML 資訊語法. 為 XML 文檔資訊專案指定屬性名的語法。* | |
| --- | --- |
| **屬性** | **描述** |
| [children] | 按文檔順序列示的子資訊專案的有序列表。 |
| [document element] | 對應于文檔元素的元素資訊專案。 |
| [notations] | 一組無序的標記法資訊專案，DTD 中聲明的每個標記法對應一組該資訊專案。如果多次聲明了任何標記法，那麼此屬性不包含任何值。 |
| [unparsed entities] | 一組無序的未解析實體資訊專案，DTD 中聲明的每個未解析實體對應一組該資訊專案。 |
| [base URI] | 文檔實體的基本 URI。 |
| [character encoding scheme] | 用來表示文檔實體的字元編碼方案的名稱。 |
| [standalone] | 表示文檔的獨立狀態，值為 yes 或 no。此屬性來自文檔實體開頭 XML 聲明中的可選獨立文檔聲明，如果不存在獨立文檔聲明，那麼它不包含任何值。 |
| [version] | 表示文檔的 XML 版本的字串。此屬性來自可選存在于文檔實體開頭的 XML 聲明，如果沒有 XML 聲明，那麼就不包含任何值。 |
| [all declarations processed] | 嚴格地講，此屬性不是文檔的資訊集的一部分。而是處理器是否已讀取整個 DTD 的指示。其值是布林值。如果它為 false，那麼某些屬性（在其下方的描述中表明）可能是未知的。如果它為 true，那麼就是已瞭解這些屬性。 |

已知所有資訊集用於描述已展開（即用其對應替換文本的資訊專案表示）所有實體引用的 XML 文檔。在實體引用不能展開的情況下，由於 XML 處理器尚未讀取其聲明或其值，將創建顯式位置來表示資訊集中的此類引用。

**SOAP 版本之間的差別**

SOAP V1.1 和 SOAP V1.2 都是萬維網聯盟 (W3C) 標準。可以部署不但支援 SOAP 1.1，而且支援 SOAP 1.2 的 Web Service。從 SOAP 1.0 到 SOAP 1.2 規範所做的一些更改很重要，而其他更改則不太重要。

SOAP 1.2 規範中包含對 SOAP 1.1 的一些更改。本資訊無意深入描述 SOAP 1.1 和 SOAP 1.2 的所有新的或已更改的功能部件。本資訊而是著重於 SOAP 的當前版本之間的重要差別。

對 SOAP 1.2 規範所做的重要更改包括下列更新：

* SOAP 1.1 基於 XML 1.0。SOAP 1.2 基於 XML 資訊集。

XML 資訊集提供使用 XSD 模式描述 XML 文檔的方法。然而，資訊集並不一定使用 SOAP 1.1 所基於的 XML 1.0 序列化來序列化該文檔。這種描述 XML 文檔的新方法有助於顯示其他序列化格式，例如二進位協定格式。可以使用二進位協定來將消息壓縮成某一可能不需要某些詳細標記資訊的壓縮格式。

在 SOAP 1.2 中，可以使用綁定至底層協定的規範確定在底層協定資料單元中使用的 XML 序列化。[SOAP 1.2 - Part 2] 中指定的 HTTP 綁定使用 XML 1.0 作為 SOAP 消息資訊集的序列化。

* 只要供應商遵從 SOAP 1.2 中定義的綁定框架，SOAP 1.2 就能夠提供正式定義傳輸協議（而不是使用 HTTP）的功能。雖然 HTTP 普遍存在，但它沒有其他傳輸協議（包括 TCP/IP 和 MQ）可靠。
* SOAP 1.2 更明確地定義了 SOAP 處理模型，該模型中移除了許多在缺少 Web Services-Interoperability (WS-I) 概要檔的情況下可能會導致互通性錯誤的二義性。其目標是極大地減少使用 SOAP 1.2 實現的不同供應商之間出現互通性問題的機會。
* “帶附件的 SOAP Java API” (SAAJ) 還可以作為一種發出 SOAP 請求的簡單機制獨立出來。對 SAAJ 規範所做的主要更改是能夠表示 SOAP 1.1 消息和其他 SOAP 1.2 格式化消息。例如，SAAJ V1.3 在 SOAP 頭元素上引入了一組更有助於 SOAP 1.2 的新常量和方法（如 getRole() 和 getRelay()）。工廠中還有其他方法供 SAAJ 使用以創建適當的 SOAP 1.1 或 SOAP 1.2 消息。
* 對於 SOAP 1.2，包絡和編碼模式的 XML 名稱空間已更改。這些更改使 SOAP 處理器能夠區分 SOAP 1.1 與 SOAP 1.2 消息，並支援 SOAP 模式中的更改，而不會影響現有實現。
* Java Architecture for XML Web Services (JAX-WS) 引入了同時支援 SOAP 1.1 和 SOAP 1.2 的功能。因為 JAX-RPC 要求在運行時遍歷 SOAP 消息時處理該消息，所以需要在其相應 SOAP 上下文中表示該消息。在 JAX-WS 中，由於支持 SAAJ 1.3 而產生了許多其他增強功能。
* Web Service 描述語言 (WSDL) V1.1 規範不討論 SOAP 1.2。SOAP 1.2 在 WSDL 2.0 草稿版中討論。WSDL 1.1 僅定義如何在 WSDL 1.1 文檔中呈示 SOAP 1.1 有效內容。要解決如何表示基於 SOAP 1.2 的服務這一問題，有另一個 W3C 文檔，它定義如何在 WSDL 1.1 文檔中定義 SOAP 1.2 有效內容。有關 SOAP 1.2，請閱讀 WSDL 1.1 綁定擴展。
* SOAP 1.1 是單個文檔。SOAP 1.2 規範分為以下幾部分：
  + Part 0 是 SOAP 非標準簡介。
  + Part 1 描述 SOAP 消息的結構、SOAP 處理模型和用於將 SOAP 綁定至底層協定的框架。一致性 SOAP 實現必須實現 Part 1 中的所有內容。
  + Part 2 描述 SOAP 核心的可選載入件，其中包括資料模型和編碼、RPC 約定以及至 HTTP 的綁定。一致性 SOAP 實現可實現 Part 2 中的所有載入件。然而，如果實現了這些載入件，那麼這些載入件必須符合規範的相關部分。

第四個文檔是規範聲明和測試集合

SOAP 1.2 對語法進行了許多更改，並且提供了 SOAP 1.1 中所描述的語義的附加的、已明確的語義。“SOAP 1.2 入門”文檔列示並描述了這些語法更改。