Tellme Networks 的馬特・奧什里（Matt Oshry）、布拉德・波特（Brad Porter）與麥克・波德爾（Michael Bodell）於 2004 年 3 月提案將跨來源支援加入 VoiceXML 2.1 以支援 VoiceXML 瀏覽器的跨來源資料請求。W3C 認為這不應該限制在 VoiceXML 而是一般的機制，因此將提案移到另一份實作備忘錄。幾個主要的瀏覽器廠商透過 W3C 的 Web 應用程式工作小組正式的將該備忘錄改寫為 W3C 工作草案並以推動成為 W3C 推薦標準為目標。跨來源資源共享（CORS）是 JSONP 模式的現代版。與 JSONP 不同，CORS 除了 GET 請求方法以外也支援其他的 HTTP 請求。用 CORS 可以讓網頁設計師用一般的 XMLHttpRequest，這種方式的錯誤處理比 JSONP 要來的好。另一方面，JSONP 可以在不支援 CORS 的老舊瀏覽器上運作。現代的瀏覽器都支援 CORS。

跨來源資源共用（Cross-Origin Resource Sharing (CORS)）是一種使用額外 HTTP 標頭令目前瀏覽網站的使用者代理 (en-US)取得存取其他來源（網域）伺服器特定資源權限的機制。當使用者代理請求一個不是目前文件來源——例如來自於不同網域（domain）、通訊協定（protocol）或通訊埠（port）的資源時，會建立一個跨來源 HTTP 請求（cross-origin HTTP request）。簡單地說，CORS (Cross-Origin Resource Sharing) 是針對不同源的請求而定的規範，透過 JavaScript 存取非同源資源時，server 必須明確告知瀏覽器允許何種請求，只有 server 允許的請求能夠被瀏覽器實際發送，否則會失敗。

有三種方式可以啟用 CORS:第一種是使用「具名原則」 或 「預設原則」。第二種是使用 「端點路由」。第三種是使用 [EnableCors] 屬性。AllowAnyOrigin：允許來自所有來源的 CORS 要求，其中包含任何配置 (http 或 https) 。 AllowAnyOrigin 不安全，因為 任何網站 都可以對應用程式提出跨原始來源要求。指定 AllowAnyOrigin 和 AllowCredentials 是不安全的設定，而且可能會導致跨網站偽造要求。 當應用程式使用這兩種方法設定時，CORS 服務會傳回不正確 CORS 回應。XMLHttpRequest 或 Fetch 在 CORS 中最有趣的功能為傳送基於 HTTP cookies 和 HTTP 認證（Authentication）資訊的「身分驗證（credentialed）」請求。預設情況下，在跨站 XMLHttpRequest 或 Fetch 呼叫時，瀏覽器不會送出身分驗證。必須要於 XMLHttpRequest 物件中或是在呼叫 Request (en-US) 建構式時設置一個特定的旗標。

再查完資料之後我也了解到 CROS 的重要性，以下有三點，第一點是 CORS 是瀏覽器的安全機制，不是 Server 的安全機制，但是 CORS 的啟用/關閉者是 Server 指使。第二點 CORS 是避免「瀏覽器上的不當操作」造成安全性資料「跨域亂傳」，不是白名單機制，這也解釋了為什麼用非瀏覽器如 Postman 就可以呼叫 API 的原因。第三點則是承第 2 點，代理 Server 即是透過非瀏覽器的角色的關係來收發 API 請求，就可避開 CORS。這謝對我們平常的生活也是息息相關的，尤其是購物網站，如果被惡意人士透過設立假的網站，雖然外表煞有其事，但是只要下定東西便很有可能發生被他人盜刷的情況發生。由此可知這對所有人

都是極其相關的。