I556170

(19)中華民國智慧財產局

(12)發明說明書公告本 **(11)證書號數：TW I556170 B**

(45)公告日： 中華民國 105 (2016) 年 11 月 01 日

(21)申請案號：105101130 (22)申請日： 中華民國 100 (2011) 年 10 月 07 日

(51)Int. Cl.： *G06F9/44 (2006.01)* *G06F17/50 (2006.01)*

(30)優先權：2011/08/31 美國 13/223,296

(71)申請人：微軟技術授權有限責任公司 (美國) MICROSOFT TECHNOLOGY LICENSING, LLC (US)

美國

(72)發明人：皮爾斯哈洛德 PIERSON, HAROLD (US)；瑞克特布蘭特 RECTOR, BRENT (US)；拉維爾馬丁 S LOVELL, MARTYN (US)；派克歷亞馬赫什 PRAKRIYA, MAHESH (US)；羅威史蒂芬 C ROWE, STEPHEN (US)；巴蘇塔莎杜克 BASU, TASSADUQ (IN)；羅達札克羅伯特 A WLODARCZYK, ROBERT A. (US)；歐米亞艾略特 H OMIYA, ELLIOT H. (US)；杜耐茲傑瑞 DUNIETZ, JERRY (US)；侯列可艾列斯 HOLECEK, ALES (CZ)；歐斯特曼羅倫斯 OSTERMAN, LAWRENCE W. (US)；曾煒 ZENG, WEI (CN)；華德華尼瑞 WADHWA, NEERAJ (IN)；索爾卡夏吉爾 SOLKAR, SHAKEEL (IN)；亞克雄金麥可 AKSIONKIN, MICHAEL (BY)

(74)代理人：李世章；彭國洋

(56)參考文獻：

US	6446137B1	US	7171539B2
US	20050102129A1	US	20090199220A1

審查人員：高嘉男

申請專利範圍項數：20 項 圖式數：4 共 39 頁

(54)名稱

將作業系統之原始應用程式介面投射至其它程式語言（二）

PROJECTING NATIVE APPLICATION PROGRAMMING INTERFACES OF AN OPERATING SYSTEM INTO OTHER PROGRAMMING LANGUAGES (2)

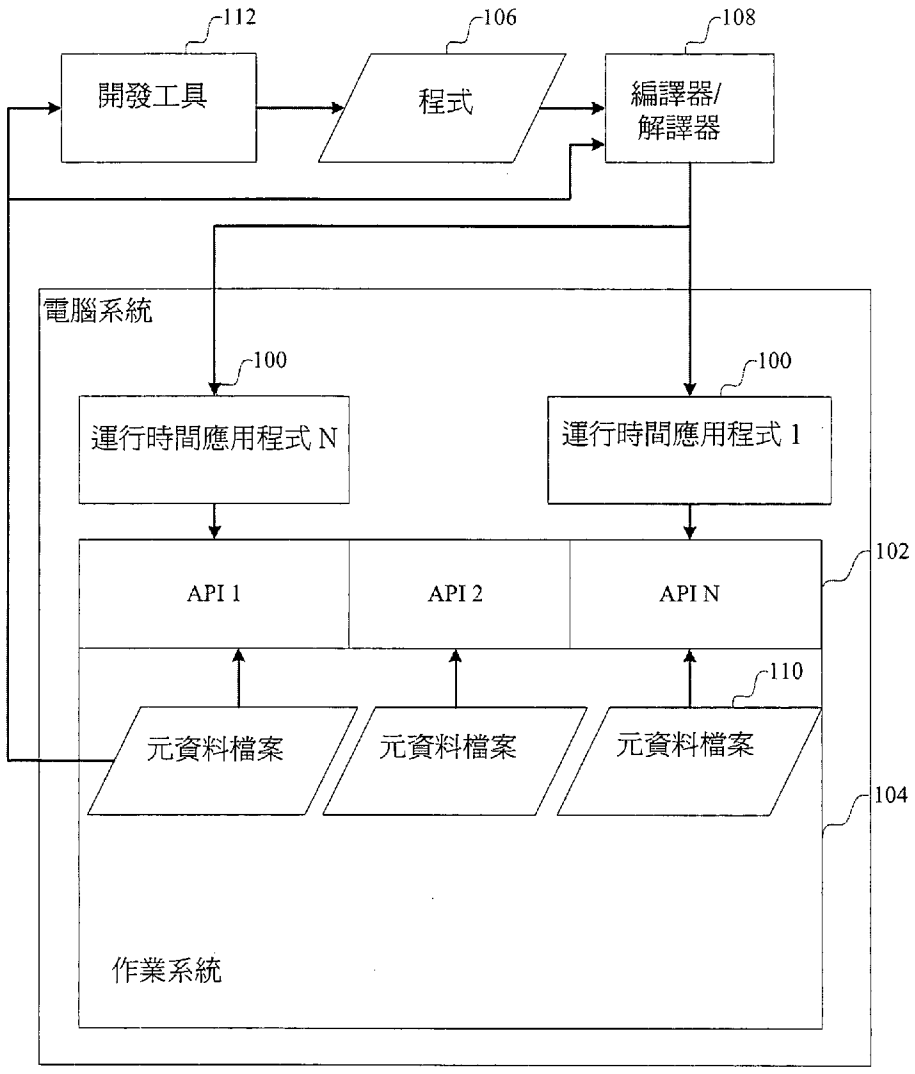
(57)摘要

關於作業系統應用程式介面的資訊係以已知格式產生且儲存於作業系統之中的已知位置。此資訊完整地描述作業系統所暴露的所有 API。此資訊包括但非限於關於多種類型之 API 之經命名的元件的資訊，該等類型的種類例如基本類型、列舉類型、結構、委派、介面、級別、方法、特性及事件。此資訊係儲存於 API 元資料檔案中。語言編譯器或解譯器使用此 API 資訊而以目標語言建立原始系統 API 的自然且熟悉的表現。此表現因語言而有所不同。語言編譯器或解譯器可於編譯時間及/或運行時間讀取 API 資訊。使用元資料以允許應用程式連結至 API 中經命名的元件。建立使用元資料的投射，以將 API 中經命名的元件映射至目標語言中經命名的元件，且定義包裝，該等包裝編列介於目標表現及原始作業系統表現之間的彼等元件的資料。

Information about the operating system application programming interfaces is generated and stored in a known format in a known location within the operating system. This information fully describes all the APIs exposed by the operating system. This includes, but is not limited to, information about named elements

of the API, of a variety of types, such as basic types, enumerated types, structures, delegates, interfaces, classes, methods, properties and events. This information is stored in API metadata files. A language compiler or interpreter uses this API information to build a natural and familiar representation of the native system API in the target language. This representation varies from language to language. The language compiler or interpreter can read the API information at compile time and/or runtime. The metadata is used to allow an application to refer to named elements in the API. Projections are built that use the metadata to map named elements in the API to named elements in the target language, and to define wrappers that marshal data of those elements between the target representation and the native operating system representation.

指定代表圖：



符號簡單說明：

100・・・應用程式

102・・・API

104・・・作業系統

106・・・程式

108・・・編譯器/解譯器

110・・・元資料

112・・・開發工具

第 1 圖

I556170

申請案號：105101130
原申請案號：100136566

公告本

申請日：100年10月7日
IPC分類：G06F 9/44 (2006.01)
G06F 17/50 (2006.01)

【發明摘要】

【中文發明名稱】將作業系統之原始應用程式介面投射至其它程式語言(二)

【英文發明名稱】PROJECTING NATIVE APPLICATION PROGRAMMING INTERFACES OF AN OPERATING SYSTEM INTO OTHER PROGRAMMING LANGUAGES (2)

【中文】

關於作業系統應用程式介面的資訊係以已知格式產生且儲存於作業系統之中的已知位置。此資訊完整地描述作業系統所暴露的所有API。此資訊包括但非限於關於多種類型之API之經命名的元件的資訊，該等類型的種類例如基本類型、列舉類型、結構、委派、介面、級別、方法、特性及事件。此資訊係儲存於API元資料檔案中。語言編譯器或解譯器使用此API資訊而以目標語言建立原始系統API的自然且熟悉的表現。此表現因語言而有所不同。語言編譯器或解譯器可於編譯時間及/或運行時間讀取API資訊。使用元資料以允許應用程式連結至API中經命名的元件。建立使用元資料的投射，以將API中經命名的元件映射至目標語言中經命名的元件，且定義包裝，該等包裝編列介於目標表現及原始作業系統表現之間的彼等元件的資料。

【英文】

申請案號：105101130　　申請日：105年1月14日
原申請案號：100136566　　IPC分類：

Information about the operating system application programming interfaces is generated and stored in a known format in a known location within the operating system. This information fully describes all the APIs exposed by the operating system. This includes, but is not limited to, information about named elements of the API, of a variety of types, such as basic types, enumerated types, structures, delegates, interfaces, classes, methods, properties and events. This information is stored in API metadata files. A language compiler or interpreter uses this API information to build a natural and familiar representation of the native system API in the target language. This representation varies from language to language. The language compiler or interpreter can read the API information at compile time and/or runtime. The metadata is used to allow an application to refer to named elements in the API. Projections are built that use the metadata to map named elements in the API to named elements in the target language, and to define wrappers that marshal data of those elements between the target representation and the native operating system representation.

【指定代表圖】第（ 1 ）圖。

【代表圖之符號簡單說明】

1 0 0　應 用 程 式

1 0 2　A P I

1 0 4　作 業 系 統

1 0 6　程 式

1 0 8　編 譯 器 / 解 譯 器

1 1 0　元 資 料

申請案號：105101130　　申請日：~~105年1月14日~~
原申請案號：100136566　　IPC分類：

1 1 2　開 發 工 具

【特徵化學式】

無

【發明說明書】

【中文發明名稱】將作業系統之原始應用程式介面投射至其它程式語言(二)

【英文發明名稱】PROJECTING NATIVE APPLICATION PROGRAMMING INTERFACES OF AN OPERATING SYSTEM INTO OTHER PROGRAMMING LANGUAGES (2)

【技術領域】

【０００１】　本發明係關於將作業系統之原始應用程式介面投射至其他程式語言。

【先前技術】

【０００２】　作業系統典型地具有數個應用程式介面，該等應用程式介面允許應用程式存取由作業系統所支援的功能。該等ＡＰＩ典型地藉由作業系統，使用電腦程式語言中的經命名的檔案或物件而指定。舉例而言，Ｃ程式語言使用可具有諸如「ｉｎｔｅｒｆａｃｅ.ｈ」名稱的標頭檔案。類似地，在Ｃ#中，稱爲「Ｐ/Ｉｎｖｏｋｅ」簽名的機制係用以存取作業系統ＡＰＩ。撰寫電腦程式而對作業系統ＡＰＩ有使用用途的人典型地在程式中對經命名的檔案或物件包括參考值，或此人典型地使用由程式語言提供的其他機制。舉例而言，該語言接著包括以該ＡＰＩ所使用的語法，呼叫由該ＡＰＩ所定義的功能。

【0003】 以此方式定義的API無法由非撰寫語言的其他語言而直接存取。爲了使得以其他語言撰寫的程式能夠存取，API被「包裝」。此包裝典型地必須對每一個API及每一個語言以人工的方式完成，且此包裝典型地必須對目標語言及API及作業系統兩者具有深入的瞭解。結果，許多作業系統API無法利用。

【發明內容】

【0004】 提供此「發明內容」以簡化的形式介紹概念之選擇，以下「實施方式」中將進一步描述該概念之選擇。此發明內容並非意欲識別所主張標的之關鍵特徵或重要特徵，亦非意欲用以限制所主張標的之範疇。

【0005】 當建立作業系統時，關於API的資訊係以已知的格式產生且儲存於作業系統之中的已知位置。此資訊完整地描述作業系統所暴露的所有API。此資訊包括但非限於關於多種類型之API之經命名的元件的資訊，該等類型的種類例如基本類型、列舉類型、結構、委派、介面、級別、方法、特性及事件。此資訊係儲存於API元資料檔案中。

【0006】 語言編譯器或解譯器使用此API資訊而以目標語言建立原始系統API的自然且熟悉的表現。此表現因語言而有所不同（正如何者爲自然

且熟悉係因語言而有所不同）。語言編譯器或解譯器可於編譯時間及/或運行時間（任何最適合用於討論中的語言者）讀取API資訊。舉例而言，類似C++的靜態編譯語言會純粹地於編譯時間消耗元資料，而類似Python或JavaScript的動態語言會純粹地於運行時間消耗元資料。使用元資料以允許應用程式連結至API中經命名的元件。建立使用元資料的投射，以將API中經命名的元件映射至目標語言中經命名的元件，且定義包裝，該等包裝編列介於目標表現及原始作業系統表現之間的彼等元件的資料。

【0007】 因此，在一個態樣中，描述作業系統的應用程式介面之元資料係儲存於記憶體中。以程式語言給定在程式中經命名的元件的指示，該指示連結至應用程式介面之一者的元件，則經命名的元件係使用元資料投射至該程式語言之中。投射可於程式的編譯或解譯期間發生。投射可包括以程式語言建立編碼，該編碼用於建立一或更多元件；且根據類型編列用於所建立的元件之資料。包括介面之方法、特性及事件的介面可如此投射。投射亦可包括將例外從作業系統傳播至應用程式。

【0008】 此舉將作業系統API投射至其他語言中係可實現於電腦實施程序、製造物品，或計算機器中，該製造物品包括一或更多電腦儲存媒體。

【0009】 在以下說明書中，參考形成說明書的一部分的隨附圖式，且其中藉由圖示的方式來顯示此技術的特定示例性實施。應瞭解可使用其他實施例，且可在不悖離所揭露的範疇之情況下改變結構。

【圖式簡單說明】

【0010】 第1圖係包括將API投射至其他程式語言的系統的方塊圖。

【0011】 第2圖係圖示開發工具的示例性操作的流程圖。

【0012】 第3圖係圖示編譯器或解譯器的示例性操作的流程圖。

【0013】 第4圖係示例性計算裝置的方塊圖，其中可實施此系統。

【實施方式】

【0014】 以下段落提供示例性作業環境，其中可實施將原始系統API投射至其他語言之中的實例。

【0015】 參看第1圖，運行中的應用程式100在運行時間期間存取作業系統104的原始系統API 102。爲了使得此應用程式具有此功能，典型地使用諸如編輯器的開發工具112撰寫程式

106。該等程式係由語言的編譯器或解譯器108進行編譯或解譯任一者，以提供運行時間應用程式100。開發工具112，及編譯器或解譯器108存取元資料110，該元資料110完整描述作業系統的API 102。開發工具112幫助開發者撰寫程式，但透過元資料告知開發者可取得的原始系統API，且允許使用開發者的程式語言存取彼等API。編譯器或解譯器使用元資料實現將原始系統API投射至開發者的程式語言中。具體而言，在API中經命名的元件係映射至目標語言的經命名的元件，且在彼等元件中的值係編列於由目標語言及作業系統所使用的格式之間。

【0016】 建立此系統係由建立具有元資料所描述的API的作業系統而開始。元資料代表以程式語言獨立形式的API描述的各個經命名的元件。此元資料提供完整的介面描述。結合的系統元資料可以ECMA-335 CLI格式儲存於一連串的元資料檔案中，但特定格式對本發明係無關緊要的。

【0017】 給定此情境，此系統的示例性實施例將更詳細地與第2-4圖連接而描述。

【0018】 在第2圖中，現將描述開發工具的操作實例。當撰寫電腦語言時，開發者通常使用諸如編輯器的開發工具的形式。此編輯器可執行各種任務，例如驗證語法及對由開發者輸入的字串提議完

備化。在此環境中，描述作業系統應用程式介面的元資料可以各種方式使用。具體而言，元資料可用以允許開發者發現可取得的ＡＰＩ。開發工具接收２００字串輸入，例如可用於ＡＰＩ中作爲識別符（例如「滑鼠」）或名稱空間（例如作業系統的名稱）的關鍵字。可搜尋２０２元資料，以識別具有將所接收的輸入作爲識別符或名稱空間的一部分的元件。可收集匹配元件組的識別符且將該等匹配元件組的識別符返回２０４至開發工具。開發工具可將元件呈現２０６至開發者用於選擇，且開發工具可使用來自元資料的資訊，以適合開發者正在使用的電腦語言的格式呈現該等元件。

【００１９】　現參看第３圖，現將描述語言編譯器或解譯器的操作的實例。在處理電腦程式中，不論編譯或解譯電腦程式，均偵測３００電腦程式的一序列元件。決定３０２電腦程式的元件是否爲作業系統的ＡＰＩ的參考值。舉例而言，此可藉由在元資料中查找元件而決定。使用元資料建立３０４投射元件，該元資料允許資料在電腦程式及作業系統之間交換。具體而言，編譯器或解譯器實施投射，該投射將作業系統ＡＰＩ中的經命名的元件投射至目標語言中的經命名的元件。元資料允許建立物件，且元資料允許於程式所使用的資料格式及作業系統所使用的格式之間編列值。於運行時間，此元件允許

程式存取306作業系統的API，且此元件允許編列介於應用程式及作業系統之間的資料。

【0020】 在已經描述此系統的大致處理之後，現將描述特定實例。具體而言，現將描述在描述API的元資料及API的元件的電腦語言特定表現之間更加詳細的示例性投射。

【0021】 以下說明係僅爲一個可能實施，且並非考慮爲限制本發明。具體而言，應瞭解以下僅爲可實施的語言投射的實例，且可能具有對此語言的其他實施，且可能具有其他投射至其他語言之中。

【0022】 在此實例中，JavaScript係爲原始系統API將使用元資料而投射之程式語言。在以下的實例中，給定某些種類的元件如何投射至JavaScript程式語言之中的解釋。

【0023】 當腳本嘗試建立由作業系統API定義的具體物件的實例時，投射器物件合理地用於存取元資料，以建立於不同類型之間轉換資料的編列存根，可能牽涉建立代理物件，且投射器物件用於派遣方法呼叫、管理事件且管理回撥功能。

【0024】 對於基本類型的經命名的物件，例如整數、字串及類似者，以下爲將彼等API元件投射至JavaScript之中的方式。

【0025】 作業系統具有數個簽署的及未簽署的整數類型，例如一個位元組的未簽署整數

(UInt8)、四個位元組的未簽署整數(UInt32)、四個位元組的簽署整數(Int8)及八個位元組的簽署及未簽署整數(UInt64及Int64)。該等類型的經命名的元件係投射成JavaScript Number值。當JavaScript Number被編列成作業系統值時，接著該等值被類型轉換爲JavaScript數，且接著跟隨著由ES5 ToInt32規格所定義的程序。對於UInt8，結果具有應用2^8模數。對於Int32，應用2^16模數。對於UInt32，應用2^32模數。

【0026】 編列成JavaScript值的64位元的整數若經簽署而落入[-2^53, 2^53]之範圍，或若未經簽署而落入[0,2^53]之範圍，則該64位元的整數被表示爲標準的Number值。若落於此範圍之外，則編列成JavaScript值的64個位元的整數被表示爲Number值，該Number值具有維持全部64位元的整數資料的自訂備用儲存。對該等自訂Number值上的數學運算造成值被類型轉換成爲在[-2^53, 2^53]範圍中的標準Number表示，或若未簽署則在[0,2^53]範圍中。若值落於此範圍之外，則將提出類型錯誤。若編列成64位元整數的JavaScript值係本身爲經投射的值，則此JavaScript值被直接地指派；否則，通過在值上應用EC5 “ToInteger” 轉換的結果。

【0027】 係爲字母（由16位元萬國碼表示）、字串或全球唯一識別符（GUID）的作業系統API的經命名的元件，可表示爲JavaScript字串，且該等經命名的元件在JavaScript中被投射成經命名的字串。

【0028】 編列成JavaScript的字母被轉換成JavaScript字串值，該JavaScript字串值含有由萬國碼值所表示的單一字母。編列成字母的JavaScript值透過ES5 ToString操作而類型轉換成爲JavaScript字串，且保持第一字母。單一字母接著以Char16值通過。

【0029】 編列成JavaScript的字串被轉換成JavaScript String。編列成字串的JavaScript值被類型轉換成爲JavaScript字串。

【0030】 編列成JavaScript值的GUID被轉換成字串格式。編列成GUID的JavaScript值被類型轉換成爲字串，且該JavaScript值接著被剖析成由作業系統所使用的格式。

【0031】 作業系統可具有API，該API具有經命名的元件，其中該經命名的元件係爲代表時間中一點的DateTime結構，或該經命名的元件係爲代表時間的量的TimeSpan結構。DateTime結構可投射成JavaScript，成爲具有備用儲存匹配

DateTime結構資料（該DateTime結構資料具有與Date實例不同的範圍及精密度）的Date實例。TimeSpan結構被轉換成毫秒，且返回成爲JavaScript Number。類似地，JavaScript Number可由毫秒轉換成100-十億分之一秒的單位，以通過作爲TimeSpan結構。

【0032】 應瞭解，藉由解譯元資料，在某些情況中投射亦可將來自原始環境的類型以語言投射重新映射至存在的類型。舉例而言，當原始API中的類型及語言投射具有相容的資料佈局時，此舉係可能且所欲的，而可立即以基本資料的類型發生。以此元資料映射，投射可單純地直接將所有操作重新導向原始類型，該等操作例如在語言類型上的方法或特性。此舉對語言開發者更自然且熟悉地使用該等類型。舉例而言，DateTime結構重新映射可以此方式實施。在原始的API中，DateTime結構在元資料中被暴露爲Windows.Foundation.DateTime而無任何額外的操作。在C#投射中，此類型於具有豐富支援的C#中可重新導向至System.DateTimeOffset。

【0033】 如另一實例，若在API中經命名的元件係一方法，則此方法具有HRESULT的值作爲該方法之返回類型，該返回類型係轉換成

JavaScript中的例外。返回的HRESULT藉由JavaScript引擎檢查成功性。若HRESULT表明失敗，則JavaScript引擎在JavaScript側上投擲例外。因此，對於作業系統API的JavaScript調用的方法，HRESULT失敗係浮現爲JavaScript Exception。對於消耗JavaScript方法的API方法（例如以下所述的回撥或委派），JavaScript方法呼叫被包裝於例外方塊中（或由JavaScript主機API提供的均等），使得捕獲的例外可傳播爲HRESULTS。然而，作業系統亦允許HRESULTs位於方法及特性的進或出參數位置。在該等情況下，HRESULTS被編列爲未簽署的32位元值（如以下所述）。

【0034】 對於在API中係列舉類型的經命名的元件（該等經命名的元件係爲一組經命名的常數），該等以JavaScript表示爲對各個經命名的值含有唯讀欄位的物件。

【0035】 對於在API中係爲「結構(structs)」的經命名的元件（經命名的資料欄位的收集），該等在JavaScript中表示爲JavaScript物件。編列成JavaScript的結構被轉換成物件。在結構中各個經命名的欄位變成在JavaScript物件中經命名的特性。在結構中各個經命名的欄位之值被編列成按照欄位的下層類型。係爲物件類型的

JavaScript值可編列成結構。JavaScript物件或JavaScript物件的原型對結構的各個經命名的欄位含有經命名的欄位，且各個經命名的欄位之值係根據下層類型編列。在JavaScript物件中，於作業系統API結構中不具有均等的額外欄位被忽略。若任何結構值的編列失敗，則返回編列錯誤。

【0036】 在此實例中，作業系統並非具有陣列的類型，但作業系統取而代之地允許對方法的引數，跟隨著對陣列的第一元件的指標，成爲代表元件的數目（並非在陣列中之位元組）之一對未簽署的整數長度。

【0037】 當將陣列編列成Javascript時，建立具有以下特徵的物件。物件具有對各個整數值介於0及陣列的長度、負1之間的特性，此特性可列舉、可撰寫但不可配置，且「長度」特性初始設定成向量的長度，此長度係不可撰寫、不可列舉且不可配置。此物件原型係爲Array原型物件。此物件在索引特性上之[[Put]]操作於下層原始陣列上設定特定的索引。此物件在索引特性上之[[GetOwnProperty]]操作於下層原始陣列上編輯索引至下層原始陣列之中。身爲「Array」，此物件不具有[[Class]]。當編列JavaScript物件時，若物件具有[[Class]]「Array」，則複製此陣列爲原始陣列且通過對此陣列的參考

值。若物件係爲經投射的陣列，則通過下層原始陣列。

【0038】 API亦可具有委派或回撥功能的經命名的元件，此係單一可調用方法的參考值。該等可投射至JavaScript中成爲可呼叫物件。投射器將回撥委派包裝於自訂編列物件中。

【0039】 編列成JavaScript的委派被包裝於JavaScript功能物件中。當調用功能物件時，編列引數成爲由委派表明的均等參數類型，且接著調用經包裝的委派物件。若任何引數編列失敗，則委派呼叫失敗。若JavaScript引數比參數式的委派更少，則委派呼叫失敗。超出參數式的委派的額外JavaScript引數被忽略。在調用委派之後，出參數被編列成JavaScript類型。若任何出參數編列失敗，則委派呼叫失敗。出參數接著被編列成JavaScript值且被返回。

【0040】 若原始JavaScript功能物件被編列成委派，則此可呼叫物件被包裝於相對應委派類型的委派中。當調用委派時，入參數被編列成JavaScript類型且接著調用JavaScript功能物件。若失敗編列任何引數，則委派呼叫失敗。在調用委派之後，返回的值被映射至基於以下規則的委派的出參數。首先，若無出參數，則忽略來自JavaScript功能物件的返回值。其次，若委派指

定單一出參數，則來自JavaScript功能物件的返回值被編列成該類型。若編列失敗，則委派呼叫失敗。第三，若委派指定多重出參數，則來自JavaScript功能物件返回值的返回值係對各個出參數具有經命名的特性的物件。各個經命名的特性被編列成相對應出參數的類型。若返回值並非物件，則委派呼叫失敗。若返回的物件對各個出參數不含有經命名的特性，則委派呼叫失敗。若返回的物件含有不相對應於出參數的額外的經命名的特性，則該等物件被忽略。若任何經命名的特性失敗於編列成相對應的出參數類型，則委派呼叫失敗。

【0041】 以下實例說明用於委派經呼叫的IString Collection的元資料，以及建立此委派的實例之擬JavaScript編碼。

【0042】 示例性元資料：

```
[uuid(...)]
delegate HRESULT Comparer([in]
HSTRING s1, [in] HSTRING s2,
    [out, retval] boolean* value)
interface IStringCollection
{
    HRESULT Sort([in] Comparer
compare);
}
```

【0043】 擬JavaScript：

```
//create an instance of a class that implements IStringCollection
strCol = getIStringCollection()
strCol.sort(function (s1, s2) { return s1 > s2; });
```

【0044】 介面並非直接投射至JavaScript成爲物件。然而，介面可爲作業系統API方法的參數及返回類型。

【0045】 爲了提供在目標程式語言中自然的投射，在以上的實例中，投射成JavaScript的成員將該等成員之名稱改變成camelCase，跟隨在JavaScript中對成員使用camelCase名稱的強的慣例之後。近似於JavaScript建造功能的類型係照慣例爲PascalCased，且該等類型被投射成該格式。類似地，enum特性、結構欄位、用於addEventListener樣式的事件名稱，在camelCase中具有名稱。

【0046】 基於靜態類型資訊未知爲運行時間級別的呈現，而被編列成JavaScript的介面實例，歷經以下步驟。首先，對介面進行呼叫，以獲得介面的運行時間級別名稱。若成功，物件被投射成運行時間級別實例物件（如下所述）。否則，物件被

投射成好似物件係恰巧執行物件已知的執行之未命名運行時間級別的實例，及物件的中間所需介面。

【0047】 檢查編列成介面類型的JavaScript值，以檢視該值係爲投射的運行時間級別實例物件或投射的介面實例物件。若該值爲運行時間級別物件，且該物件代理的初始值執行介面類型，則執行該介面的物件被通過至作業系統。否則，提出類型錯誤例外。

【0048】 在作業系統API中的物件可爲運行時間級別的實例。運行時間級別執行一組一或更多個介面（以下定義）。運行中物件的執行的介面的清單，可基於返回運行中物件的方法的元資料，或藉由存取運行時間級別名稱及查找元資料任一者而決定。因爲JavaScript係基於原型、動態的語言，此不具有級別的構造。級別構造在JavaScript中被投射爲物件。

【0049】 因此，作業系統API物件在JavaScript中被投射爲物件。定義於級別的所有執行的介面上的方法、特性及事件的合併，透過該合併之原型，而表示被暴露成在投射的JavaScript物件上可取得的經命名的特性的類型成員。JavaScript語言投射的消費者可直接存

取級別的任何成員，而無須顧慮關於成員實際上定義於何者介面上。

【0050】 JavaScript物件係爲動態的，意謂於任何時間可在物件上添加或移除新的特性。只要預定義的介面成員不被撤銷或刪除，投射的物件可支援添加新的特性及方法。在JavaScript的詞彙中，投射的物件可延伸，但經命名的類型成員的收集不可配置。投射的物件具有原型，該原型具有來自運行時間級別所執行的介面定義於成員的收集上的實例成員。

【0051】 如上所述的該等介面，具有方法、參數及事件。

【0052】 於語言投射層處的方法被執行爲對每一方法具有一儲存槽的vtable。關於介面的元資料提供方法名稱以及參數的名稱、類型及方向（進/出）。方法在JavaScript中投射爲經投射的運行時間級別或介面物件上的可呼叫特性。該等特性係爲 {Writable=false,
Enumerable=true,
Configurable=false}。當被呼叫時，引數根據引數的相對應參數類型而編列，方法以該等值而呼叫。返回值編列爲JavaScript值，而具有JavaScript物件返回作爲該值。

【0053】 於語言投射層處的特性被執行成提取及/或設定方法。存取特性值將呼叫提取方法，而更新特性值將調用設定方法。特性可爲讀取或寫入（亦即，可利用提取及設定方法），或唯讀（亦即，僅可利用提取方法）。特性在JavaScript中投射爲特性。編列特性值如以上所述取決於下層特性類型而作用。

【0054】 於語言投射層處的事件係執行爲添加及移除事件收聽者方法。添加方法採取委派實例且添加方法返回說明收聽者的資料，而移除方法採取說明收聽者且不返回的資料。

【0055】 在JavaScript中，具有至少一個事件投射於任何經投射的運行時間級別或介面物件上的任何經投射的運行時間級別或介面物件，提取兩個額外的特性添加至該任何經投射的運行時間級別或介面物件之原型，addEventListener及removeEventListener。該等特性係爲{Writable: false, Enumerable: true, Configurable: false}，且該等特性被指派爲可呼叫物件。

【0056】 addEventListener功能提取串流引數，該串流引數代表所收聽的事件的名稱、指派作爲收聽者的回撥功能，及被忽略的可選第三參數。此功能呼叫下層的add_Event方法，通過經

編列的回撥功能作爲委派，且此功能將所得到的訊標儲存於映射中，該映射由回撥功能物件的參考識別符所定調。

【0057】 removeEventListener功能採取串流引數，該串流引數代表移除收聽者的事件的名稱、應被移除的回撥功能，及被忽略的可選第三參數。此功能藉由在儲存的訊標的映射中之參考識別符而查找，且若找到訊標，則呼叫下層的remove_Event方法，通過經檢索的訊標。

【0058】 當事件被發射時，將調用以回撥通過的任何JavaScript功能物件。通過至功能物件的引數將爲提供至EventHandler委派的引數之經編列的值。

【0059】 如以上所顯示，藉由具有語言的API Projection層，由儲存於作業系統中的元資料所指定的作業系統API的經命名的元件可自動地被使用，以在作業系統API格式及應用程式語言格式之間建立物件且編列資料。應瞭解以上僅爲在示例性作業系統及示例性程式語言之間投射的實例，且本發明並非限於此實例。

【0060】 在已描述示例性實施之後，現將描述其中被設計成作業此系統的計算環境。以下說明意欲提供適合的計算環境的簡要大致描述，其中可實施此系統。可以大量的通用目的或特別目的計算硬

體配置實施系統。眾所周知可爲適合的計算裝置之實例包括但非限於個人電腦、伺服器電腦、手持或膝上型裝置（舉例而言，媒體播放器、筆記型電腦、手機、個人資料助理、錄音機）、多重處理器系統、基於微處理器的系統、機上盒、遊戲操縱台、可程式消費電子、網路PC、迷你電腦、大型電腦、包括上述系統或裝置之任一者的分散式計算環境，及類似者。

【0061】 第4圖圖示適合的計算系統環境的實例。計算系統環境係僅爲適合的計算環境之一個實例，且計算系統環境並非意欲建議對此計算環境的使用或功能的範疇作任何限制。計算環境亦不應詮釋爲具有關於圖示於示例性作業環境中的組件的任一者或結合之任何依賴或需求。

【0062】 參看第4圖，示例性計算環境包括諸如計算機器400的計算機器。在最基本的配置中，計算機器400典型地包括至少一個處理單元402及記憶體404。計算裝置可包括多重處理單元及/或額外的共同處理單元，例如圖形處理單元420。取決於計算裝置的確切配置及類型，記憶體404可爲揮發性（例如RAM）、非揮發性（例如ROM、快閃記憶體等等）或兩者之某些結合。此最基本的配置係藉由虛線406圖示於第4圖中。此外，計算機器400亦可具有額外的特徵/功能。舉例而言，

計算機器400亦可包括額外的儲存（可移除及/或不可移除），包括但非限於磁碟或光碟或磁帶。此額外的儲存係藉由可移除儲存408及不可移除儲存410圖示於第4圖中。電腦儲存媒體包括以任何方法或技術實施，用於儲存資訊的可揮發及不可揮發、可移除及不可移除媒體，該等資訊例如電腦程式指令、資料結構、程式模組或其他資料。記憶體404、可移除儲存408及不可移除儲存410均爲電腦儲存媒體的實例。電腦儲存媒體包括但非限於RAM、ROM、EEPROM、快閃記憶體或其他記憶技術、CD-ROM、數位光碟（DVD）或其他光學儲存、磁性卡匣、磁帶、磁碟儲存或其他磁性儲存裝置，或可用以儲存所欲資訊且可由計算機器400存取的任何其他媒體。任何此電腦儲存媒體可爲計算機器400的部分。

【0063】 計算機器400亦可含有通訊連接412，該通訊連接412允許裝置與其他裝置通訊。通訊連接412係通訊媒體的實例。通訊媒體典型地承載電腦程式指令、資料結構、程式模組或其他資料於調變資料訊號中，調變資料訊號例如載波或其他傳送機制，且包括任何資訊傳遞媒體。「調變資料訊號」一詞意謂具有一或更多特徵組的訊號，或以此方式改變而編碼資訊於訊號中，從而改變訊號的接收裝置的配置或狀態。藉由實例的方式且非限

制，通訊媒體包括諸如有限網路或直接連線的有線媒體，及諸如聽覺、RF、紅外線及其他無線媒體的無線媒體。

【0064】 計算機器400可具有各種輸入裝置414，例如顯示器、鍵盤、滑鼠、筆、相機、觸碰輸入裝置等等。亦可包括諸如喇叭、印表機等等的輸出裝置416。所有該等裝置係本領域中眾所周知的，且所有該等裝置無須在此花篇幅的討論。

【0065】 此系統可實施於一般情境的軟體，包括電腦可執行指令及/或電腦編譯指令，例如由計算機器所處理的程式模組。一般而言，程式模組包括常式、程式、物件、組件、資料結構等等，當該等程式模組由處理單元處理時，指導處理單元實行特定任務或實施特定抽象資料類型。此系統可執行於分散式計算環境中，在該等分散式計算環境中任務係由透過通訊網路鏈結的遠端處理裝置實行。在分散式計算環境中，程式模組可位於包括記憶體儲存裝置的本端及遠端電腦儲存媒體兩者中。

【0066】 隨附申請專利範圍的前言中，「製造物品」、「程序」、「機器」之詞彙係意欲限制請求項爲落入可專利標的之範疇。

【0067】 此處上述的任何或所有的替代實施例可使用於任何所欲的結合以形成額外的混合實施例。應瞭解於隨附申請專利範圍中定義的標的並非

必須限制爲上述的特定實施。上述的特定實施僅揭露作爲實例。

【符號說明】

【0068】

100 應用程式

102 API

104 作業系統

106 程式

108 編譯器/解譯器

110 元資料

112 開發工具

400 計算機器

402 處理單元

404 記憶體

406 虛線

408 可移除儲存

410 不可移除儲存

412 通訊連接

414 輸入裝置

416 輸出裝置

420 圖形處理單元

【生物材料寄存】

【 0 0 6 9 】 國內寄存資訊(請依寄存機構、日期、號碼順序註記)

無

【 0 0 7 0 】 國外寄存資訊(請依寄存國家、機構、日期、號碼順序註記)

無

【序列表】(請換頁單獨記載)

無

【發明申請專利範圍】

【第1項】 一種電腦實施程序，包含：

存取元資料步驟，存取儲存在一電腦中的元資料，該元資料描述該電腦的一作業系統的應用程式介面，該元資料由一機器可讀式程式語言獨立格式描述該等應用程式介面的元件，且該元資料被由一標準格式儲存於作爲該電腦的該作業系統之部分的一已知位置中；

接收指示步驟，將一指示接收入記憶體中，該指示將在一目標程式語言中的一程式中的一經命名的元件連結至該電腦的該作業系統的該等應用程式介面之一者的一元件；以及

建立投射元件步驟，由該電腦的一處理系統使用該元資料處理該程式，以建立定義在該目標程式語言中的該程式中的一投射元件的電腦程式碼，該投射元件爲該作業系統的該應用程式介面的該元件對於該程式中的該經命名的元件的一投射，在運行時間該投射元件配置該電腦的該處理系統以根據類型編列一作業系統表現與一應用程式表現之間的資料，使得在運行時間該程式中的該投射元件配置該電腦的該處理系統以允許該程式存取該電腦的該作業系統的該應用程式介面。

【第2項】 如請求項1所述之電腦實施程序，其中建立該投射元件之該建立投射元件步驟包含

以下步驟：在編譯該程式時，建立定義一或更多個投射元件的編碼。

【第3項】 如請求項1所述之電腦實施程序，其中建立該投射元件之該建立投射元件步驟包含以下步驟：在解譯該程式時，建立定義一或更多個投射元件的編碼。

【第4項】 如請求項1所述之電腦實施程序，其中該投射元件包含一介面，該介面包含方法、特性及事件。

【第5項】 如請求項1所述之電腦實施程序，其中該投射元件配置該處理系統以從該作業系統傳播例外至該應用程式。

【第6項】 一種製造物品，包含：

一電腦儲存媒體；

電腦程式指令，該等電腦程式指令儲存於該電腦儲存媒體上，當由一電腦的一處理系統處理該等電腦程式指令時，該等電腦程式指令配置該處理系統爲包含：

經配置以儲存元資料的記憶體，該描述該電腦的一作業系統的應用程式介面，該元資料由一機器可讀式程式語言獨立格式描述該應用程式介面的元件，且該元資料被由一標準格式儲存於作爲該電腦的該作業系統之部分的一已知位置中；

一語言投射器，該語言投射器經配置以接收一指示，該指示將在一目標程式語言中的一程式中的一經命名的元件連結至該電腦的該作業系統的該等應用程式介面之一者的一元件；以及

該語言投射器進一步經配置以使用該元資料建立電腦程式碼，該電腦程式碼定義在該目標程式語言中的該程式中的一投射元件，該投射元件爲該作業系統的該應用程式介面的該元件對於該程式中的該經命名的元件的一投射，在運行時間該投射元件配置該電腦的該處理系統以根據類型編列一作業系統表現與一應用程式表現之間的資料，使得在運行時間該程式中的該投射元件配置該電腦的該處理系統以允許該程式存取該電腦的該作業系統的該應用程式介面。

【第7項】 如請求項6所述之製造物品，其中該語言投射器包含一編譯器，該編譯器經配置以在編譯時間建立該電腦程式碼，該電腦程式碼定義一或更多個投射元件。

【第8項】 如請求項6所述之製造物品，其中該語言投射器包含一解譯器，該解譯器經配置以在解譯該程式時建立該電腦程式碼，該電腦程式碼定義一或更多個投射元件。

【第9項】 如請求項6所述之製造物品，其中該投射元件包含一介面，該介面包含方法、特性及

事件。

【第10項】 如請求項6所述之製造物品，其中該投射元件從該作業系統傳播例外至該應用程式。

【第11項】 一種計算機器，包含：

一處理系統，該處理系統包含一或更多個處理器與一或更多個電腦儲存媒體以及儲存在該一或更多個電腦儲存媒體上的電腦程式指令，該等電腦程式指令在由該處理系統處理時配置該處理系統爲包含：

經配置以儲存元資料的記憶體，該元資料描述該計算機器的一作業系統的應用程式介面，其中該作業系統的該等應用程式介面的元件被由一機器可讀式程式語言獨立格式描述且被由一標準格式儲存於作爲該電腦的該作業系統之部分的一已知位置中；以及

一語言投射器，該語言投射器經配置以接收一指示，該指示將在一目標程式語言中的一程式中的一經命名的元件連結至該計算機器的該作業系統的該等應用程式介面之一者的一元件，且該語言投射器進一步經配置以使用該元資料建立電腦程式碼，該電腦程式碼定義在該目標程式語言中的該程式中的一投射元件，該投射元件爲該作業系統的該應用程式介面的該元件對於該程式中的該經命名的元件的一投射，在運行時間該投射元件配置該電腦的該處理系統以根據類型

編列一作業系統表現與一應用程式表現之間的資料，使得在運行時間該程式中的該投射元件配置該電腦的該處理系統以允許該程式存取該電腦的該作業系統的該應用程式介面。

【第12項】 如請求項11所述之計算機器，其中該語言投射器包含一編譯器，該編譯器經配置以在編譯時間建立該電腦程式碼，該電腦程式碼定義一或更多個投射元件。

【第13項】 如請求項11所述之計算機器，其中該語言投射器包含一解譯器，該解譯器經配置以在解譯該程式時建立該電腦程式碼，該電腦程式碼定義一或更多個投射元件。

【第14項】 如請求項11所述之計算機器，其中該投射元件包含一介面，該介面包含方法、特性及事件。

【第15項】 如請求項11所述之計算機器，其中該投射元件配置該處理系統以從該作業系統傳播例外至該應用程式。

【第16項】 如請求項11所述之計算機器，其中該應用程式介面的該元件包含一資料結構，該資料結構包含經命名的資料欄位的收集，且對於該元件的該投射元件包含在該目標程式語言中的

一物件，該物件包含一經命名的特性收集，其中該等經命名的特性對應於該等經命名的資料欄位。

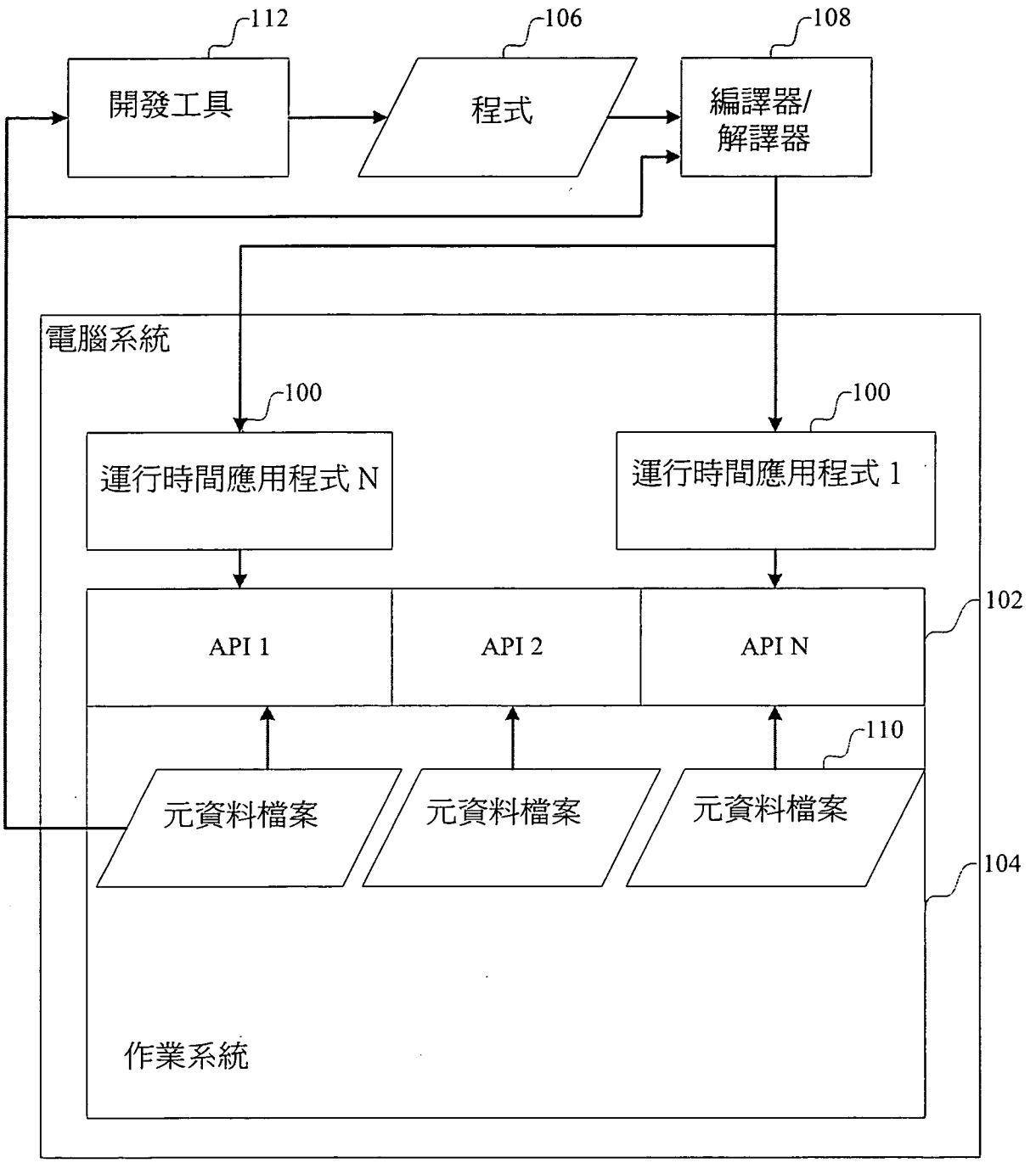
【第17項】 如請求項11所述之計算機器，其中該應用程式介面的該元件包含一資料結構，該資料結構具有在該目標程式語言中的一對應類型，且其中對於該元件的該投射元件經配置以將操作重新導向在該目標程式語言中的該對應類型。

【第18項】 如請求項11所述之計算機器，其中該應用程式介面的該元件包含一方法，且對於該方法的該投射元件包含在該目標程式語言中的一例外。

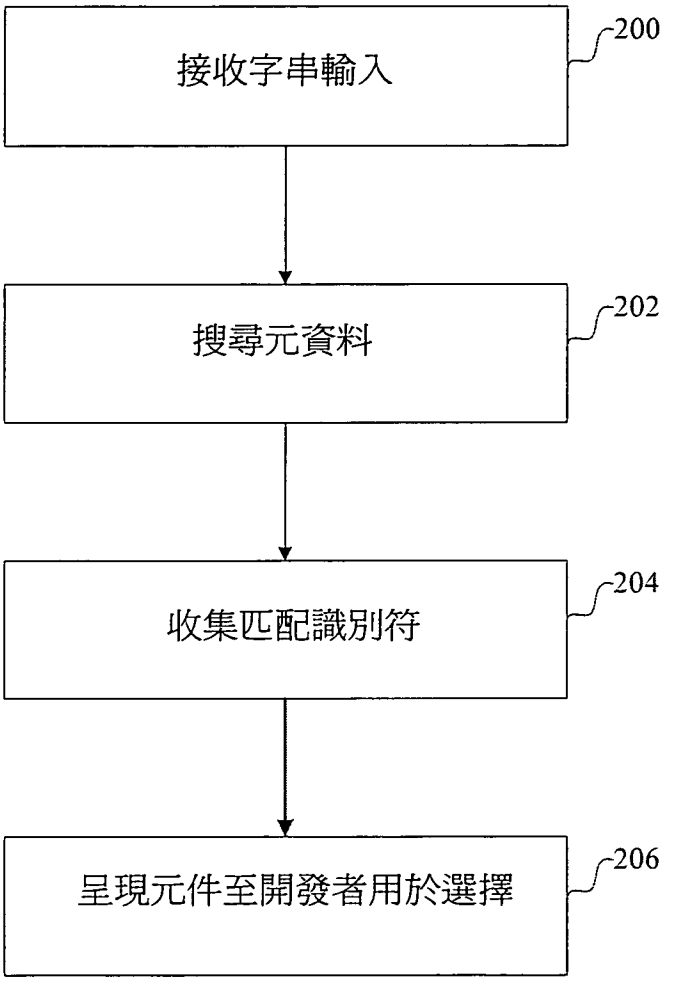
【第19項】 如請求項11所述之計算機器，其中該應用程式介面的該元件包含一列舉類型，該列舉類型包含複數個經命名的值，且對於該列舉類型的該投射元件包含在該目標程式語言中的一物件，該物件對於該列舉類型的該複數個經命名的值的每一經命名的值包含一唯讀欄位。

【第20項】 如請求項11所述之計算機器，其中該應用程式介面的該元件包含一回撥功能，且對於該回撥功能的該投射元件爲在該目標程式語言中的一可呼叫物件。

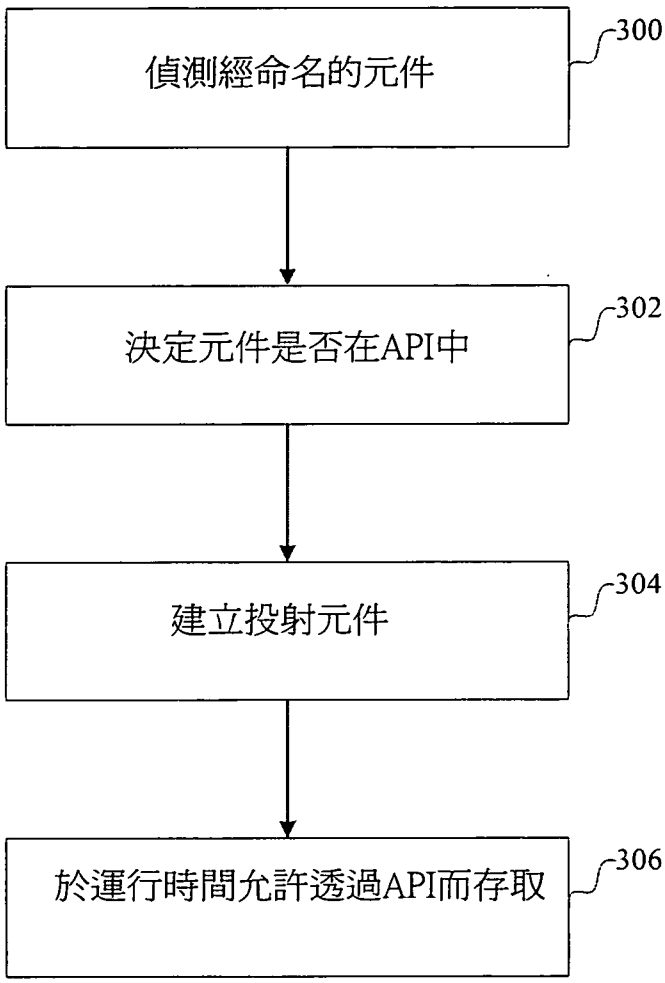
圖式


112
開發工具
106
程式
108
編譯器/
解譯器
電腦系統
100
運行時間應用程式 N
100
運行時間應用程式 1
API 1
API 2
API N
102
110
元資料檔案
元資料檔案
元資料檔案
104
作業系統

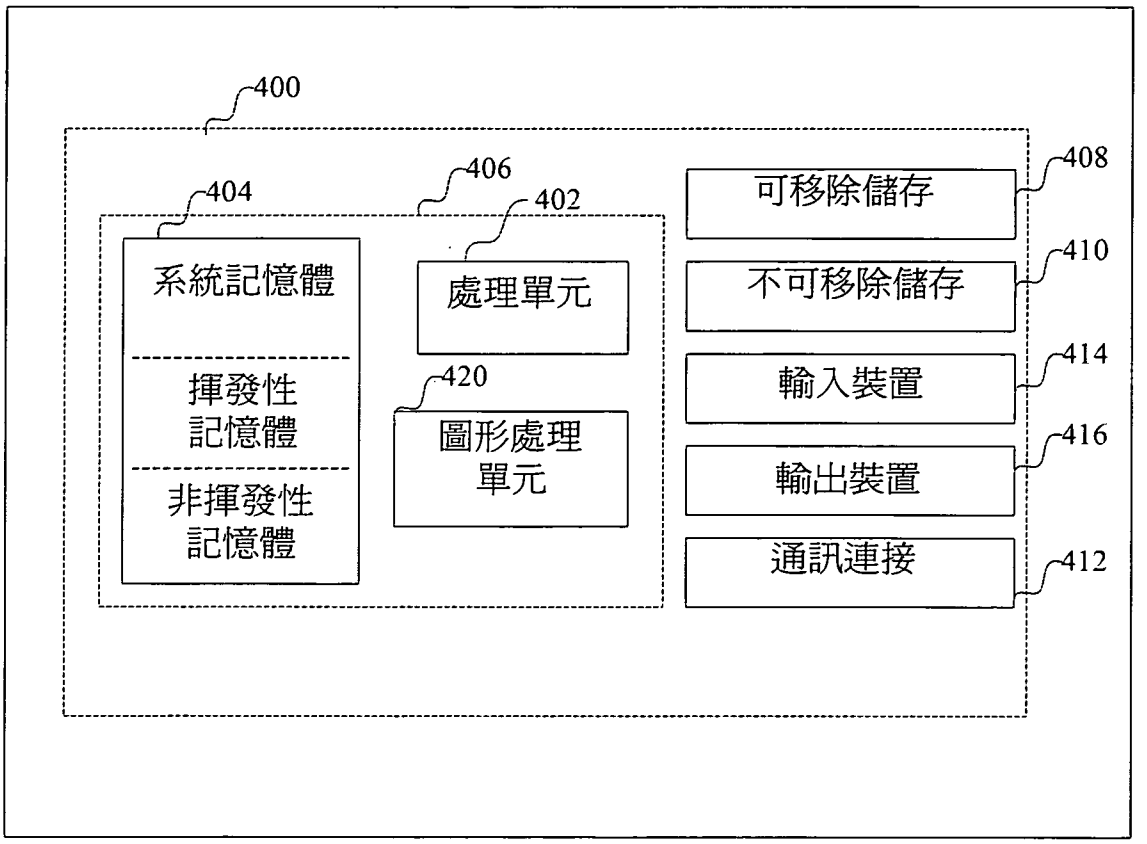
第 1 圖



第 2 圖


偵測經命名的元件
300
決定元件是否在API中
302
建立投射元件
304
於運行時間允許透過API而存取
306

第 3 圖


400
404
406
402
系統記憶體
揮發性記憶體
非揮發性記憶體
處理單元
420
圖形處理單元
可移除儲存
408
不可移除儲存
410
輸入裝置
414
輸出裝置
416
通訊連接
412

第 4 圖