

contract method calls **751**, or web services **25** sending responses to either method calls **703** or API contract method calls **753**.

[**0087**] If the client application interface unit **310** receives a login request (**1002**) to a web service **25**, the client application interface unit **310** sends the login request to the login services module **980**. Alternatively, the login request may be passed to the communication processor **311** to be passed onto the login services module **980**. The login services module **980** processes the login request (**1003**). The authentication module **520** validates the client application **15**, as described above. If the login request is successful, i.e., the client application **15** is valid, then the authentication ID provider **960** issues an authentication ID for the client application **15**. Information regarding the web service **25** for which the authentication ID is valid may be stored in the authentication ID validator **965**. Alternatively, the information relating to the issued authentication ID may be stored in an additional repository accessible by the authentication ID validator **965**. The client application interface unit **310** returns the authentication ID to the client application **15** (**1004**). The gateway module **500** now returns to a state of listening for communications (**1001**). This thread will be further described below.

[**0088**] If the client application interface unit **310** receives a client call **701** or a developer call **751** (**1005**), the client application interface unit **310** passes the call **701**, **751** to the communication processor **311**. The call **701**, **751** contains an authentication ID passed as a parameter. The web method call processor **960** would process a client call **701** (**1006**), as described above. The API contract processor **965** would process a developer call **751** (**1006**), as described above. The removed authentication ID is sent to the authentication ID validator **965** for validation. If the authentication ID is valid, then the communication processor **311** passes the corresponding WS call **702** or API call **752** to the web services interface unit **312** to be delegated to the appropriate web service **25** (**1007**). Information regarding the client application **15** and the call **701**, **751** may be logged in a repository. The gateway module **500** now returns to a state of listening for communications (**1001**).

[**0089**] If the web services interface unit **312** receives a WS response **703** or a WSDL **753** (**1008**), then the web services interface unit **312** passes the WSDL **703** to the communication processor **311**. The web method call processor **960** would process a WS response **703** (**1009**), as described above. The API contract processor **965** would process a WSDL **703** (**1009**), as described above. If the call **702**, **752** and the response **703**, **753** are asynchronous, then information stored in a repository may be accessed to determine the client application **15** which sent the original call **701**, **751** corresponding to the response **703**, **753**. If the call **702**, **752** and the response **703**, **753** are synchronous, then the identity of the client application **15** which sent the original call **701**, **751** is clear. The client response **704** or modified WSDL **754** is passed to the client application interface unit **310** to be sent to the client application **15** (**1010**). The gateway module **500** now returns to a state of listening for communications (**1001**).

[**0090**] Other steps may be added to the method (**1000**), including metering the gateway module **900** usage and billing the client application **15**, as well as billing a web service provider **20**.

[**0091**] In alternative examples of a communication processor **311**, the web method call processor **960** and the API contract processor **965** may comprise further sub-components, to handle their tasks. For example, the web method call processor may either contain a SOAP method call processor and a SOAP response processor. Similarly, the API contract processor may contain a WSDL communication processor and a WSDL response processor. Method call processors for other protocols and API contract processors for other API contract documents may be added to the web method call processor and the API contract processor, respectively. Furthermore, the communication processor **311** may be created to only contain desired sub-components.

[**0092**] An alternative to the login thread (**1001-1002-1003-1004-1001**) described above will now be described. Some of the detail in this alternative description may be used to augment the description of the above thread as well.

[**0093**] Using a secured channel is safer but slower than an unsecured channel since a secured channel requires the extra encryption/decryption steps. An alternative solution is to have the client application **15** log into the web service **25** once by sending client application credentials, typically a user name and password, over the secure channel and in return the client application **15** will receive a unique authentication identifier (ID) over the secured channel. Sometimes an authentication ID is called a session ID. However, there is a distinction between a session ID that refers to a locked communication between a client and a server and a session ID that refers to the fact that authentication has occurred. Thus, the term authentication ID is used in this specification.

[**0094**] Successive calls to the web service **25** are then made over an unsecured channel with the authentication ID to identify the client application **15**. Since the client application credentials are not sent during the successive calls, the calls no longer need to be done over a secure channel. The calls can be sent over an unencrypted channel, such as http. This will improve performance as well as limit the number of times that the client application credentials are sent. When the server receives a web service call, it will authorize the client application **15** by verifying that the authentication ID is valid at that point in time.

[**0095**] This use of an authentication ID is only partially acceptable since the client credentials are safe as they are passed over the secure channel once and the client application **15** can still be authenticated for access to web services using the authentication ID. The problem is that since the web service calls are not done over a secured channel, the authentication ID could be compromised. Anyone who is observing the unsecured channel could note the authentication ID as it is used in the web service calls. The observer could then reuse this authentication ID and gain unauthorized access to the web service **25**.

[**0096**] One adaptation to the use of an authentication ID is to have the authentication ID time out after a certain period of time. Once an authentication ID has expired, anyone who has obtained it with or without authorization will no longer be able to use it and the authorized user will have to logon again and receive a new authentication ID. While the time-out of an authentication ID solution is better than no solution, there is still the problem that a misuse of a web service may occur for a limited time.

[**0097**] Another aspect of the gateway module **900** described here uses a pool of authentication IDs. A pool of