

scalable across multiple processing units, and state data can be persisted across multiple requests. This is achieved by having a thread-safe data store. Thus, in one example of a gateway **900**, all of the data in the entire system, including the web service registry **530**, the web method registry, the authorization tables, the billing tables which indicate how to bill clients, the actual billing data which indicate the details of a bill, potentially the authentication tables (users/passwords), potentially the authentication ID to user mappings, and the audit logs are kept in a thread-safe data store, such as the repository **1610**. There is no state information in any of the modules of the gateway module **900** as all state is effectively kept in this data store. This feature is an implementation detail of how the various modules accomplish their intent.

[**0126**] The gateway module **300, 500, 900** enables the production of a homogeneous, comprehensive and extendable solution that will work for parties in a web service distribution framework. The gateway **300, 500, 900**, can recursively service a chain of web services providers **20**.

[**0127**] Another problem that arises from providing web services **25** is that in order to create revenues, the web services **25** must also incorporate billing of the client applications **15**, auditing of all transactions that occur, administration of client applications **15** and billing schemes and any other common functionality that arises from the fact that a web service **25** is being sold to a client application **15**. For example, each web service provider **20** can host any number of web services **25**. Each web service provider **20** can contain any number of functions (or methods), which can potentially be billed in a different manner. Service providers **20** might wish to tailor billing and authorization schemes for each function to its client applications **15**. Therefore, the permutation of user to billing/authorization pairs to be maintained can be large.

[**0128**] There are many complexities regarding billing and authorization. Each billing and authorization method can take on any number of parameters. For example, a pay per use billing scheme can be tagged with any price. Groups of functions, perhaps across web services **25**, might only make sense to be billed and authorized the same way. For example, blur image effect filters published by different companies and distributed across different web services **25**. Billing schemes can vary across time and temporal cycles. For example, a web service provider **20** might want to charge more during busy hours. Authorization may be independent of billing schemes. For example, an administrator might want to temporarily deny a user from accessing certain functions. However, the billing scheme should remain effective once the restriction is lifted. Finally, some billing methods are intrinsically tied to authorization. For example, the client might pay \$**100** to use a given web service function one hundred times. Afterwards, the web service **25** will have to become unavailable.

[**0129**] Billing schemes may be created such that a billing module in the web services management system **1600** (or the billing module **970** in the gateway module **900**) requires information regarding which web services methods a client application **15** has used. Such information is tracked by the logging and metering module **1640** (or the metering module **950**) and stored in the repository **1610** (or a central repository of the gateway module **900**). Alternatively, the logging

and metering module **1640** (or the metering module **950**) may pass the web services methods usage information directly to the billing module (or billing module **970**) to be used or stored for later use.

[**0130**] The following description will refer to web services management systems **1600**. However, the authorization schemes portion of the following description also applies to the gateway module **900**.

[**0131**] A web service comprises one or more methods which users or client applications may invoke. Different web services may have a different amount of methods. In embodiments of the gateway module **300, 500, 900**, web services are registered with a central repository. Identifications of the methods of the web services are stored in the repository and grouped into bundles. A bundle of methods contains one or more methods from one or more web services. For example, consider four web services where a first web service contains 10 methods, a second web service contains 20 methods, a third web service contains 100 methods, and a fourth web service contains 50 methods. A bundle may be created to contain one method which belongs to the first web service, two methods which belong to a second web service, and six methods which belong to a fourth web service. Another bundle may be created to contain six methods which belong to the first web services, 11 methods which belong to the second web service, 71 methods which belong to the third web service, and all 50 methods which belong to the fourth web service. Other combinations of methods belonging to a plurality of web services containing a plurality of methods may be created.

[**0132**] With reference to **FIG. 18**, the following describes how the web services management system **1600** is used to package web services. **FIG. 18** shows client applications **15** assigned to packages **1801** comprising of bundles **1802**. The bundles **1802** comprise web services methods or routines **1803** which a client application **15** may use when calling web services methods **1803**. In this example, the basic bundle **1802** contains matrix manipulation, math, miscellaneous, and third party web services methods **1803**.

[**0133**] Web services methods are registered in the repository **1610** (or a central repository in the gateway module **900**). The web services methods are then grouped into bundles **1802**. Client applications **15** may also register with the web services management system **1600**. Client applications **15** usage rights relating to the bundles are stored in the repository **1610**. Thus, by accessing the repository **1610** to determine if a client application **15** has usage rights to a bundle, the authentication and authorization module **1620** determines if a client application **15** is authorized to access a web services method in that bundle.

[**0134**] The bundles **1802** are grouped into packages **1801**. The web services methods **1803** of bundles **1802** are billed and authorized in the same manner within the same package **1801**. For example, Package 1 contains a basic bundle, a file conversion bundle, and a text editing bundle. Package 2 contains a basic bundle, a bitmap methods bundle, a vector methods bundle, and a special effects bundle. Other packages **1801** containing other combinations of bundles **1802** may be created.

[**0135**] Advantageously, providing web services functionality through packages **1801** eliminates much of the logistics