SOAP STANDARD

The Single Object Application Protocol (SOAP) is a XML-based protocol for information exchange in a decentralized, distributed environment. It consists of the following components:

- Envelop–defining a framework for describing a message's content and how to process it
- Set of encoding rules for expressing instances of application-defined data types
- Convention for representing responses and remote procedure calls

Microsoft, IBM and other IT firms proposed SOAP to World Wide Web Consortium as one of middleware transport standard based on XML. Microsoft applies SOAP in its BizTalk framework (a software platform which supports XML applications), and IBM, Oracle and Sun Microsystems have also incorporated SOAP into their Universal Description, Discovery Integration (UDDI) registry.

UDDI STANDARD

Universal Description, Discovery Integration (UDDI)–any company that wants to locate a component to provide some specific service will be able to send a description of the desired component to the UDDI registry and find out if such a service exists and where. The Microsoft .NET Framework relies on the UDDI registry, providing the example for other companies to follow the software giant.

For example, a customer of an e-retail business can trace its delivery through a link to a transportation firm. This link is provided by the latter to the e-retailer. In the future, the e-retailer will look to the UDDI registry for that component and link.

WSDL STANDARD

WSDL (Web Service Description Language)–specifies a common XML framework for describing a network service. A network service is a software

Copyright © 2003, Idea Group Inc. Copying or distributing in print or electronic forms without written permission of Idea Group Inc. is prohibited.

component that performs a specific function and can be accessed by another application (a client, a server or another network service) over the Internet using ubiquitous protocols. The network service is delivered on demand through a well defined interface.

A network service usually is hosted in a central location and as a shared service is delivered to different devices and users. It is accessible through the Internet as an XML interface via a communication protocol SOAP, being listed by the UDDI registry, and delivered as an XML message.

Several vendors offer their version of network services; for example, IBM's Web Application Framework for e-business, Microsoft's .NET, HP's NetAction, Oracle's Dynamic Service Framework, and Sun Microsystems' Open Net Environment (ONE). Each vendor provides its software tools to pursue a stronger grip on proprietary technology; Sun Microsystems on Java and Microsoft on Windows. A developer's tools include common features:

- Programming environment, including a set of API's (Application Programming Interface) that integrate service components as a part of network services
- Server infrastructure for operating network services
- Interfaces to different devices on the Internet

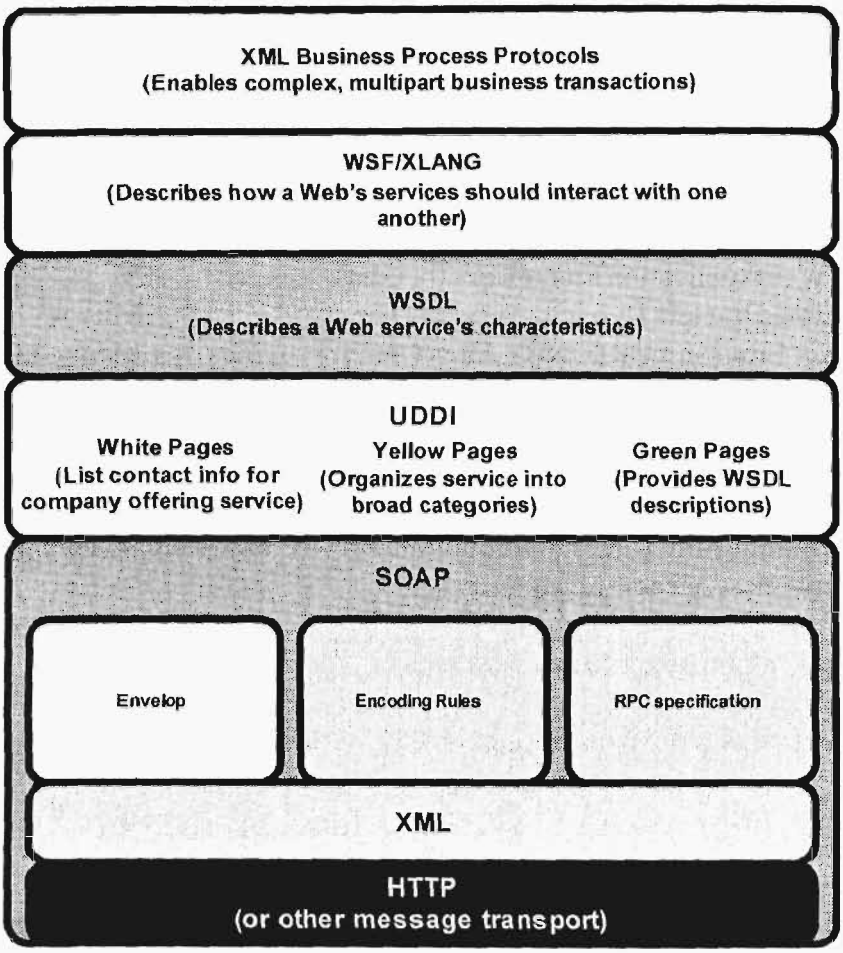
BASIC WEB SERVICES PROTOCOLS

The XML standard was designed to facilitate exchange between multiple systems. It is a framework for data description. However, it needs another three standards such SOAP, WSDL and UDDI (described above) to make Web services effective. Figure 5-9 depicts a relationship among these standards. Only Microsoft and IBM have led the development of all three. In fact, SOAP, WSDL, and UDDI are the foundation of Microsoft's .NET initiative, which is based on Web services.

All these three standards are multi-vendor specifications, allowing each of them to develop their Web service framework. A Web service applying these standards may be written in any language and run on any platform, as long it has an XML wrapper that conforms to these basic Web services standards (PriceWaterhouseCoopers, 2002).

Copyright © 2003, Idea Group Inc. Copying or distributing in print or electronic forms without written permission of Idea Group Inc. is prohibited.

Figure 5-9: A Stack of Web Services



APPLICATIONS INTEGRATION

The emergence of a digital marketplace has impacted the traditional business landscape much like the fall of the Berlin Wall on foreign policy. The traditional "brick-and-mortar" structures of the past are evolving into a whole new way of conducting business. The new business structure evolves into the "brick-and-click" way of doing business. It means that the Application Layer of the Enterprise Information Infrastructure must be integrated into one electronic environment.

Copyright © 2003, Idea Group Inc. Copying or distributing in print or electronic forms without written permission of Idea Group Inc. is prohibited.

Figure 5-10 illustrates such an integration that is characterized by the following system interactions:

- Interactions of fulfillment systems such as SCM (Supply Chain Management), ERP (Enterprise Resource Planning), CRM (Customer Relationship Management), MIS (Management Information Systems, including MKS [Management Knowledge Systems]) which take care of mission-critical information processing of the enterprise is secured by the application logic INTEGRATION,

- Interactions of facilitating systems such an Enterprise Information Portal (EIP), e-DMS (e-Document Management System), WFS (Workflow System), e-Biz, EUC (End User Computing) which support the above systems in fulfilling their missions is secured by the ELECTRONIZATION through the application of the Web technology,

- Interactions of mobile applications is provided by wireless networks and gateways (MOBILIZATION) to enterprise networks,

- All the above applications can interact and be integrated through enterprise networks (LAN, WAN, GAN, and Intranet and Extranet) and the Internet,

- All the above interactions and integration are possible through the applications and communications STANDARDIZATION.

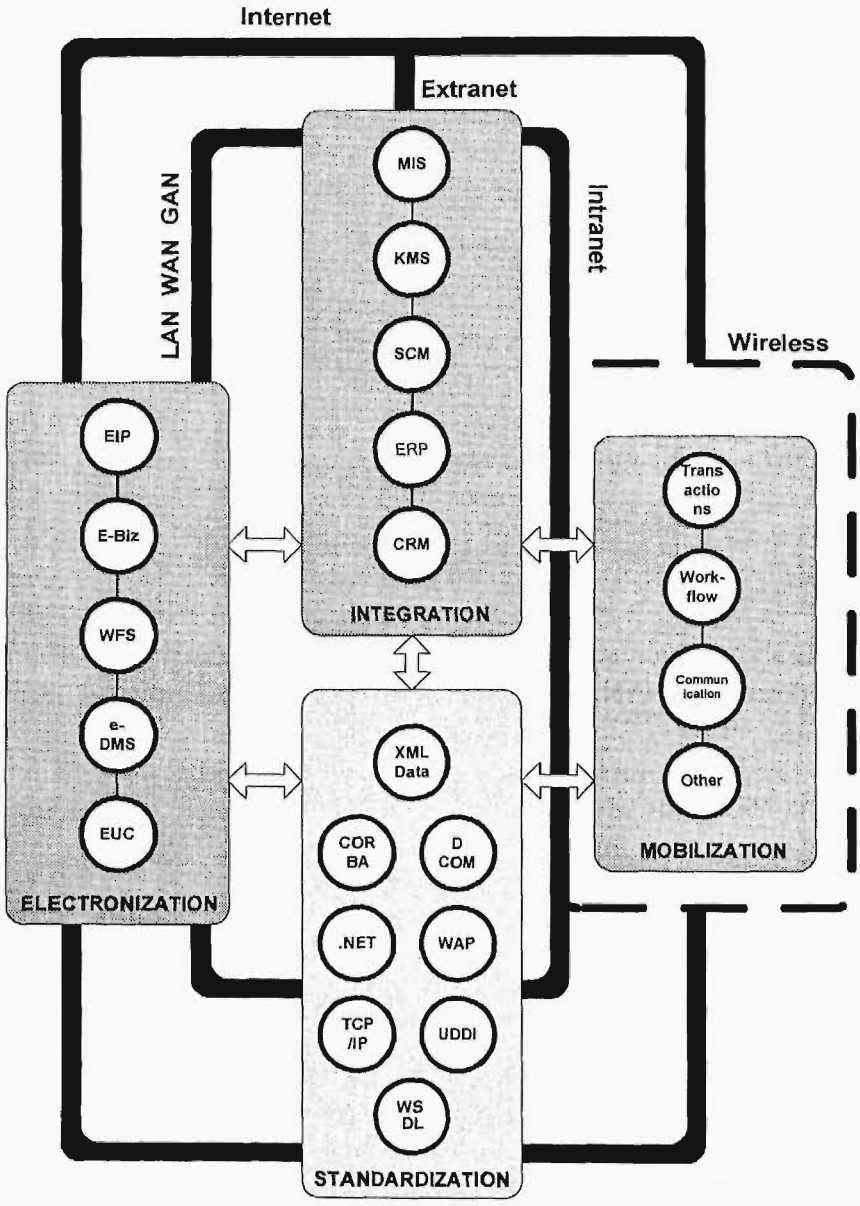
The integration of these applications takes place nowadays through their decomposition into components that are handled according to such standards as CORBA, EJB, DCOM, or .NET. These standards require that a business process must be broken down into a set of components that communicate with one another through various *integration points*. Determining the best implementation for a specific integration point requires that the enterprise's Application Layer defines several functional and system-level attributes of communication.

Functional attributes of applications integration, according to *PeopleSoft*'s framework, include:

- Direction of the processing (that is, inbound and/or outbound). Does the integration point accept data (inbound direction) or does it transmit information (outbound direction)?

Copyright © 2003, Idea Group Inc. Copying or distributing in print or electronic forms without written permission of Idea Group Inc. is prohibited.

Figure 5-10: Applications Integration



Copyright © 2003, Idea Group Inc. Copying or distributing in print or electronic forms without written permission of Idea Group Inc. is prohibited.

- Expected application behavior and performance. What is the timing of the integration? Does it happen right away? Is a response required to complete a business transaction? If there is no real-time link between the components, what is the latency between when the sending components send information and when the receiving components process it?

- Volume required to support inbound and/or outbound processing. In today's integration landscape, data volume is an issue. Usually, a large volume of data requires a different type of communication than smaller-volume integration.

System attributes, according to *PeopleSoft* framework, include:

- Available formats. What formats are the sending components capable of producing? Can the receiving components accept that format? Is some sort of transformation required to migrate the data from one format to another?

- Low-level interfaces available to implement the *integration point*. Application Programming Interface (API) is another term that gets wide use and whose meaning changes depending on context. Some people refer to the *entire integration point* as an API. For the context of this consideration, API will refer to the more technical layer that allows the communication between applications. *Integration point* will refer to the higher-level interface exposing the application component. An application programmer should deal with the *integration point,* not the lower level API.

Application integration is an important, but complex undertaking. There are several major types of application integration points that must be addressed well to deliver a comprehensive, powerful integration environment. Current application integration solutions do not support the breadth of application integration required by organizations today.

Most solutions consist primarily of something between integration points and more low-level API's that are packaged with an application or application suite. They are designed to allow third-party applications to initiate communication through a standard synchronous mechanism. They are not a complete solution. They do not represent easily identifiable business entities. Other

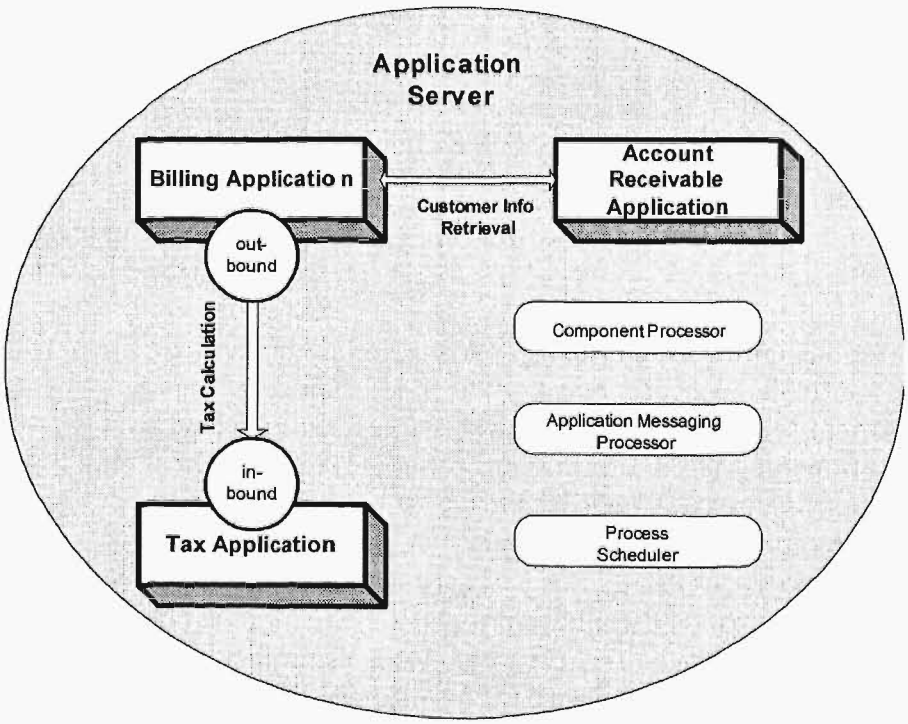
Copyright © 2003, Idea Group Inc. Copying or distributing in print or electronic forms without written permission of Idea Group Inc. is prohibited.

integration methods include: tightly coupled integration and loosely coupled integration.

Tightly Coupled Logic Integration[8]

When components are tightly coupled, the two parts are linked in real-time. The called component must return some type of information before the initiating component can complete its transaction. Tightly coupled integration is sometimes referred to as synchronous processing, or request/reply. For example, when a Billing application is calculating a customer bill online, it needs tax information. Assume that the Billing application developer decides that rather than maintaining tax rules, rates, and tax algorithms within their Billing application, they wish to leverage a tax application. In this case, a component in the Bill application requests processing from a component in the Tax application. To get an accurate tax calculation, the Billing application may also

Figure 5-11: The Tightly Coupled Integration (The Database Server is not shown)



Copyright © 2003, Idea Group Inc. Copying or distributing in print or electronic forms without written permission of Idea Group Inc. is prohibited.

need to retrieve customer information from the accounts Receivable application, as illustrated in Figure 5-11.

From the standpoint of the Billing application, it is initiating the communication, so there must be an outbound integration point to retrieve the tax information. Correspondingly, there must be an inbound integration point in the Tax application to accept the information necessary to produce an accurate tax calculation. Since the final bill amounts include tax, the Billing user must wait while the tax application processes the request and replies with the tax amount. This integration requires tight coupling to deliver the required performance. Tight coupling would be accomplished by providing an integration point in the tax calculation component, which performs the tax calculation and returns the desired data via a reply when synchronously called.

The volume of data transmitted is of low to medium volume, consisting of detailed bill information, such as customer location and product information. The volume of information received is low, consisting of the calculated tax amount for the bill. The big advantage to tight coupling in this instance is the real-time link it provides, since the Billing application needs the tax amount to calculate the bill. However, in order for Billing to call a different calculation component, it either would need to be modified for each new tax calculation routine it was interfaced with, or it would require a flexible mechanism for calling out. The second option is more difficult, but can maximize options across multiple organizations with varying requirements for tax services.

Loosely Coupled Logic Integration[9]

Not all integration is best achieved through tightly coupling applications integration. The benefits from that type of integration can easily be seen when the required integration engages communications among several components. Let's take an example of hiring a new employee that requires the interaction of the following applications (Figure 5-12):

- Human Resources – enters the new employee's personal information and job information,

- Enterprise Directory – assigns the new employee a network ID and appropriate security access to other company information resources based on the employee's job code and department,

Copyright © 2003, Idea Group Inc. Copying or distributing in print or electronic forms without written permission of Idea Group Inc. is prohibited.

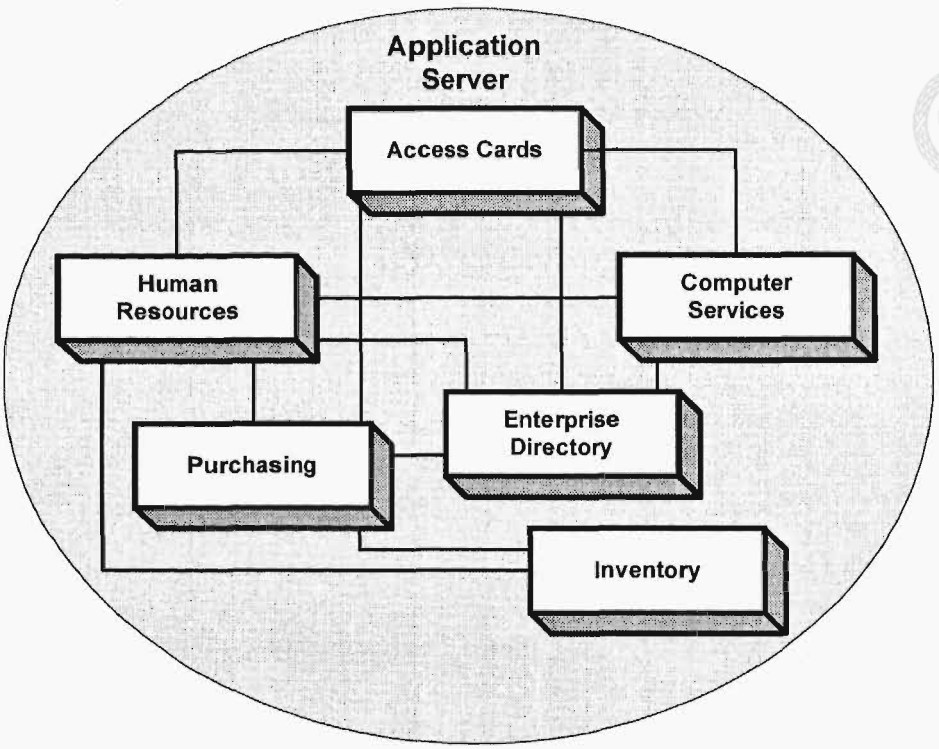
- Access ID Card Application – adds the employee to a database to track building clearance and generates a security card to allow entry to appropriate buildings,

- Computer Service Application – creates a work order to build a computer for the new employee, based on the appropriate configuration for the employee's department,

- Purchasing – enters requisitions for purchasing standard documentation and office supplies on behalf of the new employee.

The business requirement is that the business event of hiring an employee in the Human Resources application automatically triggers the appropriate processing in the other applications. The overall process is actually more complex at most enterprises, which typically have more applications involved

Figure 5-12: The Loosely Coupled Integration (The Database Server is not shown)



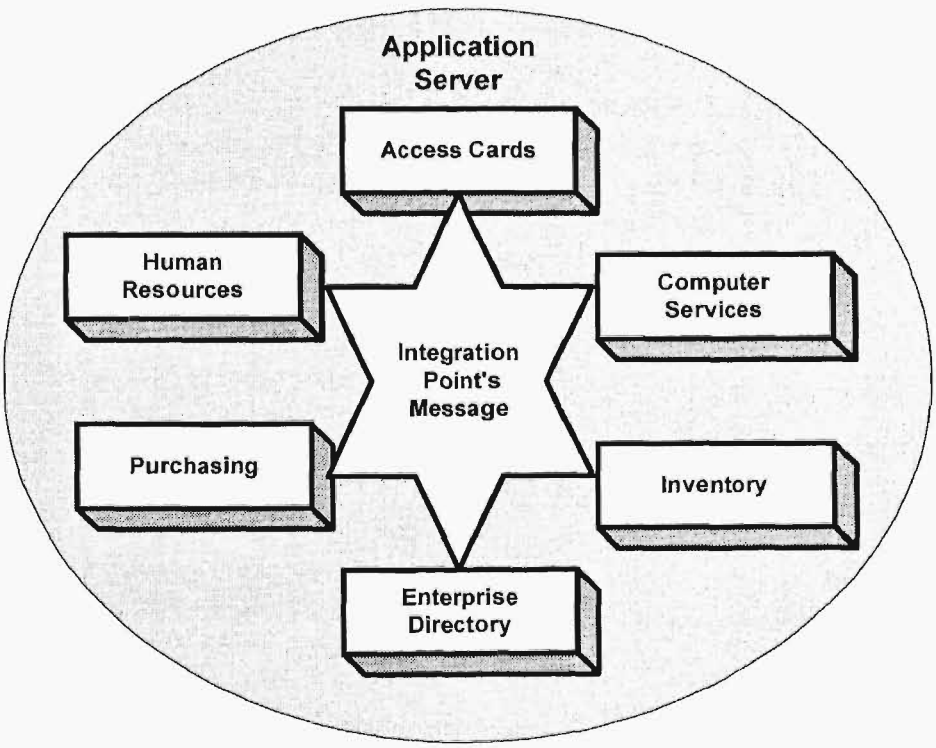
Copyright © 2003, Idea Group Inc. Copying or distributing in print or electronic forms without written permission of Idea Group Inc. is prohibited.

and require follow-up processing steps for each component, as well as other related communications between the other components.

Since Human Resources does not require information from the other components in order to complete its unit of work, loosely coupled integration is the preferred integration approach. The HR application can publish an electronic Employee Hire message, and all the components needing that information can subscribe to the message. Besides this first step, the task integration requires interactions among remaining components. Rather than integrating the business process components through separate synchronous connections, it would be better to define a single outbound integration point for the HR application through which data would be published in loosely coupled configuration. The format of such an integration point in this case would be a message, as shown in Figure 5-13.

In practice, a collaboration among different components takes place among different vendors' software. This collaboration changes according to

Figure 5-13: The Integration Point's Message



Copyright © 2003, Idea Group Inc. Copying or distributing in print or electronic forms without written permission of Idea Group Inc. is prohibited.

the enterprise's evolution of practice. However, this simple HR example indicates how many different components are usually involved in a complex procedure.

In a loosely coupled messaging system, the other involved components would subscribe to the HR application's message to a relevant form for their individual applications. In other words, each receiving component takes what it needs from the HR application's message, and converts it to a meaningful form for its application. To do so, an application messaging system will deliver messages to the appropriate components (look at the Application Messaging Processor in the *PeopleSoft* Internet Architecture on Figure 5-1).

When possible, applications should communicate in loosely coupled fashion. If a transaction needs data from another system to continue, or if an end user is waiting for a response, the integration point will tend to require tight coupling. For the broader range of application integration, including most system-to-system or business-to-business communication, loosely coupled integration based on messaging is sufficient.

MOBILE INTEGRATION

Wireless networks transport information between thin clients and content providers. Mobile clients have limited screen, keyboards, and storage; therefore servers must play a strong role in solving these problems. Other problems are caused by the small bandwidth, usually between 9,600 and 19,200 bps, which is responsible for slow data transfer and long delays, and higher latencies of responses to users.

To minimize these problems, some standards are evolving, such as:

- Wireless Application Protocol (WAP) for mobile Internet applications
- NTT DoCoMo's i-mode protocol (Japan) widespread in the marketplace

WAP was developed by Ericsson, Nokia, Motorola and UP (now Openwave) in September 1997. WAP applies Internet protocols when possible but departs from them to address limits of wireless communication. Because the wireless provider may be dropped by the end-user, WAP orients a Wireless Session Protocol to a content provider. It means that an enterprise

Copyright © 2003, Idea Group Inc. Copying or distributing in print or electronic forms without written permission of Idea Group Inc. is prohibited.