**Fact Mats**

**Interface vs Integration**

**Interface vs. Integration**

*Definition: An interface is where two or more separate software products communicate under limited capacity. Data is maintained in multiple locations; thus, requiring more administration. A fully integrated system means that the products are one. Source: Patterson*

**The Interface vs Integration**

An interface is where two or more separate software products communicate under limited capacity. Data is maintained in multiple locations; thus, requiring more administration. A fully integrated system means that the products are one.

**The Interface – A “Bridge” or Separate Application**

A software interface is a bridge that allows two programs to share information with each other, even though they may have been developed by different sources or use different programming languages. An interface will often use a standard file format such as XML to move information from one system to another. Some interfaces are separate programs that can be configured and deployed with a range of systems, like Microsoft’s BizTalk, whereas some are built into a product itself.

**The Interface – “Real Time” Synchronisation is Not Possible**

A downside of using an interface is that it doesn’t allow you to sync data between the systems in real-time. For some companies, this may not be an issue as they only need time record data to sync with payroll twice a month, right before payroll is due for example. For others, this may be an issue. In some solutions, you can choose how often information is synchronized between systems (once a week, twice a month, once a month, etc.)

**Integration – Synchronisation is Not Necessary, The Solutions Share the Same Database**

With integrated software, there is no synchronization process since the solutions all share the same database. This feature is one of the biggest benefits of having an integrated software solution.

**Interfaces Require “Mapping”**
Interfaced systems don’t share the same database, so an interface often requires maintaining “mappings” between systems. This is the process of matching codes from one system with codes from the other system, like job codes in HR with job codes in your scheduling solution. “Mappings” act as the directory for where information from one system goes into the other. For example in the interface table, you would write which time code maps onto a specific pay code in your payroll system so that people are paid correctly.

**Interface – Change Applications, Change Mapping**

With an interface, if any changes are made in either system, the mappings table may have to be updated as well. Otherwise, the software might be pulling information from the wrong places. Using the same example, if you added a new pay code in your payroll system, you need to make sure that you also change the mappings in your interface if you want this code to be used by other systems.

Since integrated solutions share the same database, there is no process of mapping codes between systems, which can reduce errors. All of the changes apply automatically in each part of the system.

For example, with ‘System X Payroll’, if you make a change in the ‘System x General Ledger’, that change is automatically accessible by ‘System X Payroll’ – there is no need to make a change in both systems.

**The Interface in More Depth**

In computing, an interface is a shared boundary across which two or more separate components of a computer system exchange information. The exchange can be between software, computer hardware, peripheral devices, humans and combinations of these. Some computer hardware devices, such as a touchscreen, can both send and receive data through the interface, while others such as a mouse or microphone may only provide an interface to send data to a given system.

A software interface may refer to a wide range of different types of interface at different "levels": an operating system may interface with pieces of hardware. Applications or programs running on the operating system may need to interact via streams, and in object oriented programs, objects within an application may need to interact via methods.

A key principle of design is to prohibit access to all resources by default, allowing access only through well-defined entry points, i.e. interfaces. Software interfaces provide access to computer resources (such as memory, CPU, storage, data, etc.) of the underlying computer system; direct access (i.e. not through well designed interfaces) to such resources by software can have major ramifications—sometimes disastrous ones—for functionality and stability.

The interface of a software module A is deliberately defined separately from the implementation of that module. The latter contains the actual code of the procedures and methods described in the interface, as well as other "private" variables, procedures, etc. Another software module B, for example the client to A, that interacts with A is forced to do so only through the published interface.

An interface is thus a type definition; anywhere an object can be exchanged (for example, in a function or method call) the type of the object to be exchanged can be defined in terms of its
interface rather than specifying a particular class. This approach means that any class that implements that interface can be used.

In computer programming, an Application Programming Interface (API) is a set of subroutine definitions, protocols, and tools for building application software. In general terms, it is a set of clearly defined methods of communication between various software components. A good API makes it easier to develop a computer program by providing all the building blocks, which are then put together by the programmer. An API may be for a web-based system, operating system, database system, computer hardware or software library. An API specification can take many forms, but often includes specifications for routines, data structures, object classes, variables or remote calls. POSIX, Microsoft Windows API, the C++ Standard Template Library and Java APIs are examples of different forms of APIs. Documentation for the API is usually provided to facilitate usage.

In computer software, an application binary interface (ABI) is the interface between two program modules, one of which is often a library and/or operating system and the other one is usually an application created by a regular programmer. In contrast to an API, which defines structures and methods one can use at software level, an ABI defines the structures and methods used to access external, already compiled libraries/code at the level of machine code. It does this by determining in which binary format information should be passed from one program component to the next, or to the operating system in the case of a system call. Thus it sets details such as the calling convention.

**Integration & The Benefits**

**Integration – Working as “One” Solution**

Software integration means that the products work as one solution. Instead of passing information between the two systems over a bridge, the systems share the same code and database. They’re like a puzzle where each piece interlocks tightly with the others and together, all the pieces become one. If you have a wide range of systems that need to be updated, or complex real-time reporting requirements, then an integrated system like an ERP may be the best choice.

**Integration – Acting as a Coordinated Whole**

System integration is defined in engineering as the process of bringing together the component sub-systems into one system (an aggregation of subsystems cooperating so that the system is able to deliver the overarching functionality) and ensuring that the subsystems function together as a system, and in information technology as the process of linking together different computing systems and software applications physically or functionally, to act as a coordinated whole.

System integration involves integrating existing often disparate systems and is also about adding value to the system, capabilities that are possible because of interactions between subsystems. In the modern world connected by Internet, the role of system integration engineers is important: more and more systems are designed to connect, both within the system under construction and to systems that are already deployed.

Many types of business software such as supply chain management applications, ERP systems, CRM applications for managing customers, business intelligence applications, payroll and human resources systems typically cannot communicate with one another in order to share data or business rules. For this reason, such applications are
sometimes referred to as islands of automation or information silos. This lack of communication leads to inefficiencies, wherein identical data are stored in multiple locations, or straightforward processes are unable to be automated.

Enterprise application integration is the process of linking such applications within a single organization together in order to simplify and automate business processes to the greatest extent possible, while at the same time avoiding having to make sweeping changes to the existing applications or data structures. Applications can be linked either at the back-end via APIs or (seldom) the front-end (GUI).

In the words of the Gartner Group, EAI is the "unrestricted sharing of data and business processes among any connected application or data sources in the enterprise."

The various systems that need to be linked together may reside on different operating systems, use different database solutions or computer languages, or different date and time formats, or may be legacy systems that are no longer supported by the vendor who originally created them. In some cases, such systems are dubbed "stovepipe systems" because they consist of components that have been jammed together in a way that makes it very hard to modify them in any way.

If integration is applied without following a structured EAI approach, point-to-point connections grow across an organization. Dependencies are added on an impromptu basis, resulting in a complex structure that is difficult to maintain. This is commonly referred to as spaghetti, an allusion to the programming equivalent of spaghetti code.

An Integration Platform is defined as a computer software which integrates different applications and services. It differentiates itself from the Enterprise application integration which has a focus on supply chain management. It uses the idea of System integration to create an environment for engineers.

Integration platforms can be built from components, purchased as a pre-built product ready for installation or procured from an integration Platform as a Service (iPaaS) offering.

Source: Wikipedia; Lokisys