sage 300 sage 300c

Understanding the benefits of the Sage 300 software architecture

Contents

Introduction

Why is it important to understand the architecture of a business management software system?

When choosing a business management solution, you are making a significant investment. The immediate and ongoing costs include licensing the software, training staff and adjusting business processes. Selecting a product with superior functionality will deliver a quick payback from more efficient operations. Selecting a product with a superior architecture will further ensure that your investment will continue to pay dividends for many years to come.

A superior software architecture enables products built within its framework to adapt to fast-changing technology and stand the test of time. Sage has made a substantial investment in developing the Sage 300 architectural framework—and this investment continues with the web-based framework for Sage 300c, featuring web screens.

Sage 300 and Sage 300c are available in three editions, which, in increasing levels of functionality, are Sage 300 Standard, Advanced, and Premium. All editions share the same architecture and code base. The Sage 300c web screens leverage a new presentation framework, but reuse the existing business view and database layers.

The Sage 300 and Sage 300c architecture has been created by Sage, and is owned by Sage. This architectural technology is the key distinction between Sage and our competitors.

A software architecture exists in an ecosystem of industry-standard technology, and its benefits cannot be stated without reference to this technology. But using industry-standard technology is not the same as having an architecture. Many software companies without a real architecture make inflated claims based on technology they *use* but have not *developed*. The Sage 300 and Sage 300c architecture has been created by, and is owned by, Sage. This architectural technology is one of the key distinctions between Sage and its competitors.

To appreciate the value of a robust software architecture, consider the risk and expense to which a business will be exposed by building a large, complicated business application on a poor architecture. Such a system is difficult and expensive to maintain. Adding new features requires a major redesign of the system, and reliability is compromised as new defects are introduced. As a result, you wait longer for reliable updates, while your competitors using well-architected products are able to quickly embrace new technologies and features and enjoy the resulting productivity gains sooner.

Before introducing the Sage 300 architecture, this document will provide real-life examples of benefits that flow from a superior architectural design.

What is an architecture?

A software architecture is a foundational design specification—not a product. Software architecture is not a product any more than traditional architecture is a building.

Just as a beautiful building embodies beautiful architecture, a superior software product embodies superior software architecture. The architecture is the design framework according to which the product is constructed. Architecture determines the stability of the product and the limits to which its functionality can be extended.

Software architecture determines how well (or how poorly) a software product fits into the ecosystem of industry-standard software technology. It determines how well the product can adapt to new technologies and business and computing paradigms. And it determines how well it can scale to meet increasing user numbers, transaction volumes, and other demands.

What are the organizational fundamentals of a great business management software architecture?

First, the **separation of core business logic from interface and database services** is essential. With these three layers separated, core business logic can be connected to evolving interface components and devices without costly reimplementation efforts and with the ability to maintain a single business logic code base. The three layers are separated by two "interfaces" (an Application Program Interface, or API), providing "plug-in points" that connect the layers.

Architecture determines how well (or how poorly) a software product fits into the ecosystem of industry-standard software technology.

Organization of the core business logic is also critical. If business processes embodied in the logic are to connect to industry-standard technologies, every element of the business logic must be accessible from the other layers of the application.

In object-oriented design, well-implemented business logic is organized into a collection of components or "objects", the behavior of which can be modified and customized without making changes to the object itself. (In object-oriented terminology, the objects can be "sub-classed" and one object can "inherit" properties from another.)

The benefit of a strong architecture and good implementation is that the resulting product:

- **Embraces industry-standard technology** quickly and naturally.
- **Customizes quickly and safely** to fit the unique needs of your business.
- **Deploys flexibly** to paradigms such as cloud or SaaS.

- Scales to meet the evolving demands of your business.
- Stands the test of time.

Stands the test of time

A product with a sound architecture does not need to be rewritten every time technology or platforms change. A company with a true architectural message does not change the message every year or two as technologies and platforms change. A Windows® user interface and a web browser-based user interface should both be useable within the same framework.

This does not mean that an architecture does not adapt and shift. The very essence of a good architecture is that it adapts easily. But **the fundamental tenets of great architecture do not change.** If the central messages about a product's architecture keep changing, it might be a clue that the architecture is flawed and unsustainable.

Embraces industry-standard technology

Software technologies that interconnect software applications are among the most important software technologies for a business application.

- The **Microsoft® Component Object Model (COM)** allows multiple desktop applications to work together.
- The **Model-View-Controller (MVC)** pattern divides a software application into three interconnected parts, separating internal representations of information from the ways that information is presented to users or accepted by them.
- For communication over the Internet, **Extensible Markup Language (XML)** and **JavaScript Object Notation (JSON)** allow servers and Internet devices to exchange information quickly and efficiently.
- **Open Data (OData)** is an open protocol that allows the creation and consumption of queryable and interoperable RESTful APIs in a simple and standard way.

To take advantage of these technologies, a product must have defined contact points (or interfaces) into which the technology can be connected. An architecture must define these contact points and have a consistent way for the application to communicate with them.

The issue is not that an application can use a particular technology (such as COM or XML), or be structured with the MVC pattern, but that the application provides contact points designed to work with emerging standards and technologies.

Customizes quickly and safely

Every business has unique needs, so a good software architecture must support some degree of customization. Common requirements for customization include importing or exporting information, adding optional fields, customizing screens and reports, automating commonly used processes, and even modifying the core business logic itself.

The strength of a good architecture lies not in its compatibility with a single, specific tool or technique, but in its flexibility and power to connect to multiple tools and techniques.

Further, a strong architecture enables the whole application to use this tool—every single component, not just a select few. Can Visual Basic for Applications (VBA) intercept a few data entry fields, or can it drive the complete business logic of an application? Can every database table be exported, or only a select few?

Finally, it is important that when a product is customized, the **core code and data are protected and customizations remain upgrade-safe.** The best technique for achieving this is to employ objects whose functionality can be inherited in the desktop screens.

Deploys flexibly to new paradigms

Today's products must adapt not only to new technologies, but also to information technology (IT) and business paradigms, such as online hosting. You may not want to deploy a hosted application today, but it is important that the product you purchase can be deployed flexibly to accommodate this paradigm and others that may emerge in the future.

A key deployment factor is **the ability to deploy a software product in multi-tier environments**, with the same core business logic able to run on database servers, application servers, or client machines. Flexibility of deployment is a sure sign that the underlying architecture is sound.

Scales up as your business grows

To scale, a program must be capable of **handling an increasing number of transactions and users**, and of **storing and retrieving rapidly growing data sets** without performance issues.

In the case of a business management software product, the program must be capable of handling a large and growing number of records, including customers, vendors, accounts, items, orders, and other data. To do this, a product must interact efficiently with its database services—so the underlying architecture must be able to take advantage of database-specific operations to tune the product's performance.

Allows greater flexibility

Because the needs of small and medium businesses are unique, a single software solution or vendor will not likely satisfy all needs of all companies. Ideally, you have chosen a business management solution such as Sage 300 or Sage 300c that manages the majority of your business processes in one place but realistically, there will be other point software solutions that your company also uses for more specialized processes. Therefore, it's vital that you have the ability to integrate all of your disparate software solutions together for better business workflow.

A strong architecture allows **freedom of choice and integration** across the whole software stack. When you choose business software with a great architecture, you can then choose from a large number of third-party add-on products that seamlessly integrate with—and greatly enhance—core business functionality.

As new versions of integrated products are released with features that could benefit your business, you'll have the ability to **upgrade and manage connected applications** in the way that provides the greatest benefit to your business. You need the flexibility to be able to choose multiple software solutions and not risk interoperability issues.

Misleading claims about software architecture

Many software products do not include even the basic features of a software architecture. Some lack even basic object orientation at the application level.

Does that stop them from producing “architecture” white papers? Of course not. So what do they talk about? Technical details are put forward as “proofs” that they have an architectural plan. Microsoft technology is often cited, as if this automatically bestows an architecture on products that use it. Let’s look at some common claims that you should watch out for.

Claim: “Using a Microsoft language like VB, C, C#, or C++ means a product has a strong architectural foundation.”

This is a common claim for products that are written as monolithic Visual Basic or C/C++. It implies that even though a product isn’t modular or extensible, it somehow has an architecture just because it was written in a Microsoft-supported language.

The claim’s credibility is based on Microsoft’s cachet, but the truth is that the implementation language has nothing to do with whether a product has an architecture or not.

Good or bad programs with good or bad architectures can be produced in any programming language. However, **architectural value comes from a well-designed framework that makes the product modular, extensible, and customizable.** In fact, it’s often better to write different application components in whichever programming language or system is best for the job.

Claim: “Developing exclusively for Microsoft platforms means a product has an architecture.”

Developing for Microsoft’s platforms has some advantages. Using “best of breed” tools has advantages; tools generally work well together. The truth is that either a “Microsoft only” or a “best of breed” strategy may be a good approach to take. But let’s be clear—this has nothing to do with having or not having an architecture.

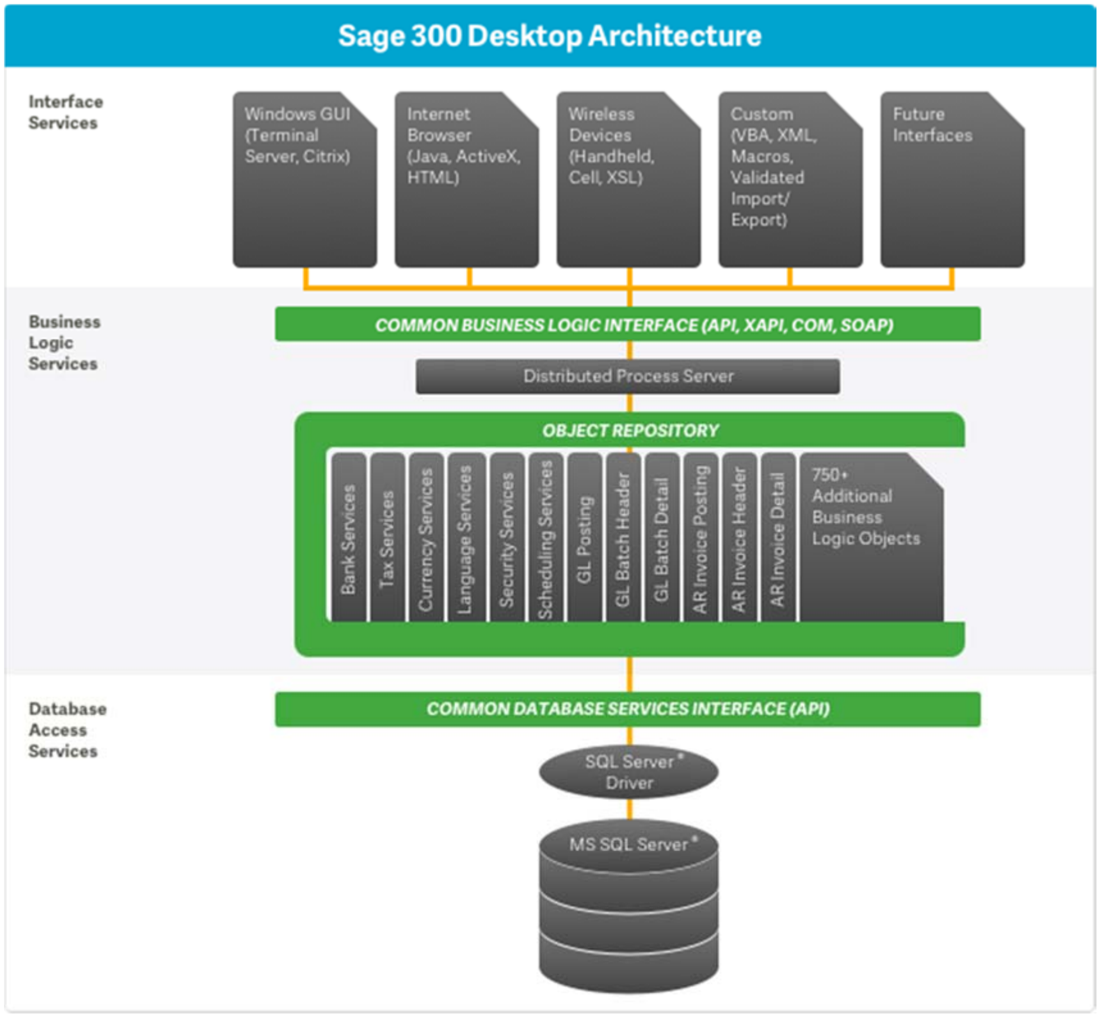
Claim: “The product is written in .NET, so it has a strong architecture.”

Microsoft® .NET™ is the basis for the current family of programming languages supported by Visual Studio®. These are an updated set of programming languages with a common runtime and a common programming framework, and they include a lot of good technology.

However, **being written for .NET doesn’t mean a product has a good architecture**; it is just an implementation decision. You can just as easily implement a poorly architected system in .NET as you can in any other programming environment.

The Sage 300 architecture

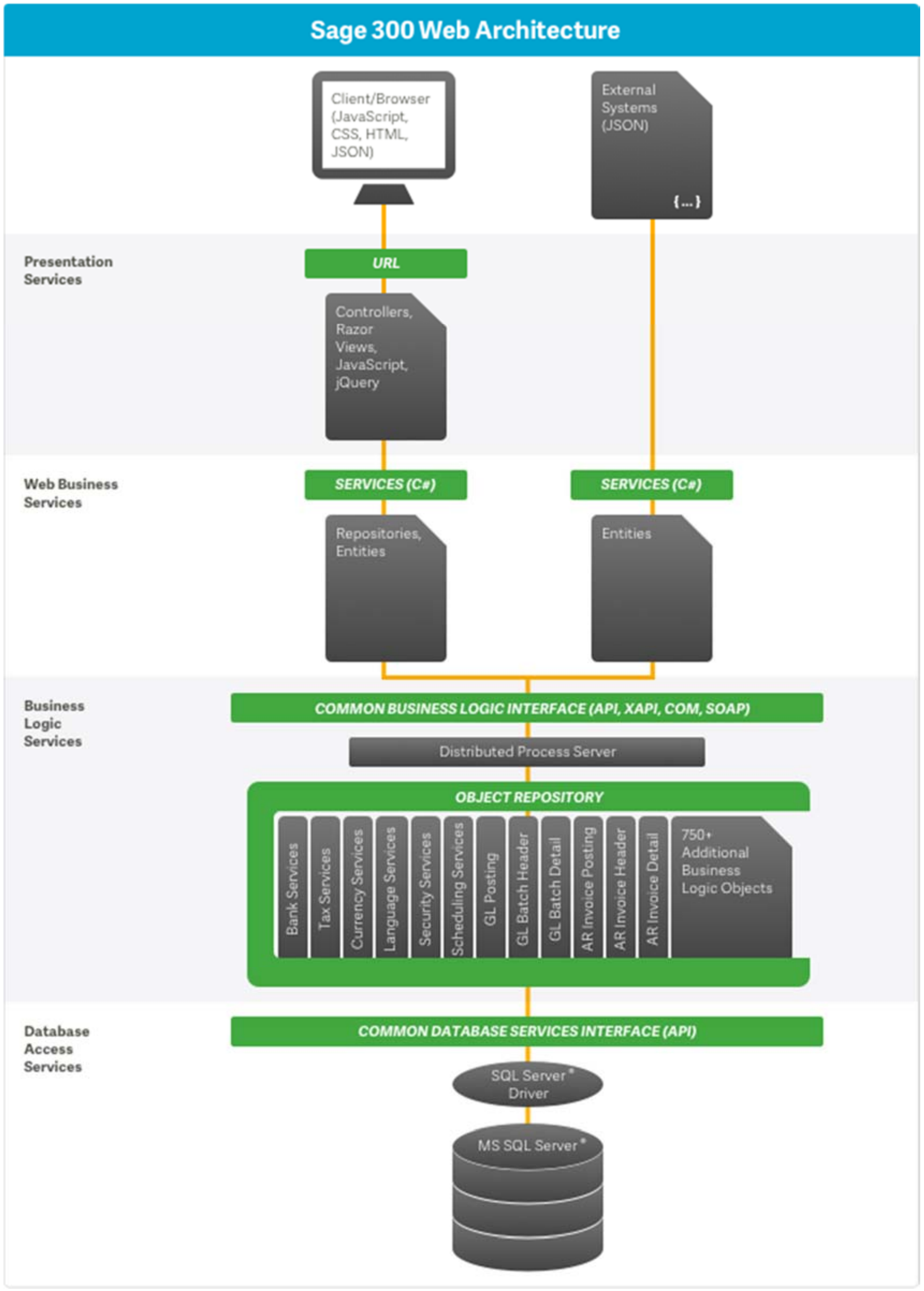
The Sage 300 architecture is based on separation of core business logic from user interface and database services.


Sage 300 Desktop Architecture
Interface Services
Windows GUI (Terminal Server, Citrix)
Internet Browser (Java, ActiveX, HTML)
Wireless Devices (Handheld, Cell, XSL)
Custom (VBA, XML, Macros, Validated Import/ Export)
Future Interfaces
Business Logic Services
COMMON BUSINESS LOGIC INTERFACE (API, XAPI, COM, SOAP)
Distributed Process Server
OBJECT REPOSITORY
Bank Services
Tax Services
Currency Services
Language Services
Security Services
Scheduling Services
GL Posting
GL Batch Header
GL Batch Detail
AR Invoice Posting
AR Invoice Header
AR Invoice Detail
750+ Additional Business Logic Objects
Database Access Services
COMMON DATABASE SERVICES INTERFACE (API)
SQL Server* Driver
MS SQL Server*

The Sage 300 Desktop architecture is object-oriented and separates user interface services, core business logic, and database services. The architecture allows for easy adaptation of new user interfaces, while maintaining a single common code base for business logic.

Among other strengths, this separation of services supports the development of different user interface technologies, including the Sage 300c web screens launched in 2015.

While Sage 300 Desktop screens use Visual Basic to display functionality in a Windows application, Sage 300c web screens use a browser-based user interface to interact with the same business logic and data. **Users can use either interface to process transactions simultaneously without conflicts.**

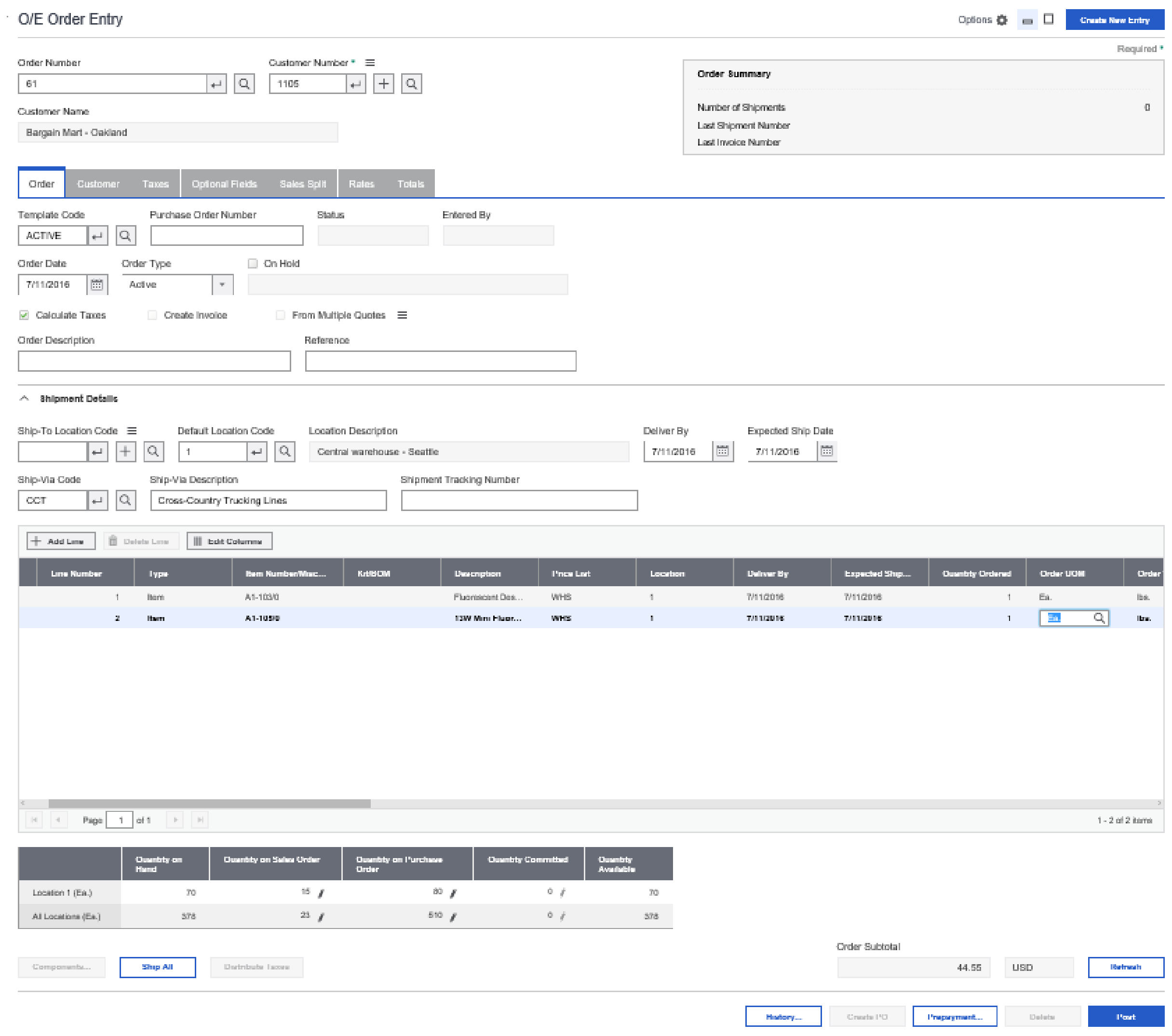


The Sage 300 Web architecture allows for browser and Web API access in Sage 300c, and is implemented in a Model-View-Controller pattern. The architecture leverages the business logic and database layers of the Sage 300 Desktop architecture.

Separation of core business logic

The Sage 300 architecture is designed to stand the test of time by isolating and minimizing dependencies on workstation and network operating systems, and on user interface environments. **The architecture features a strict separation of interface services, presentation services, business logic services, and database access services**, with interfaces that provide consistent layer-to-layer communication.

Sage 300 user interface code is separate from business logic. If a new user interface has to be added, this can be done without affecting core business logic. Whether you're using a Windows application, a browser-based application, or a mobile app, you're accessing the exact same core logic.



Order Number: 61
Customer Number *: 1105
Customer Name: Bargain Mart - Oakland

Order Summary
Number of Shipments: 0
Last Shipment Number:
Last Invoice Number:

Template Code: ACTIVE
Purchase Order Number:
Status:
Entered By:
Order Date: 7/11/2016
Order Type: Active
On Hold: ☐
Calculate Taxes: ✓
Create Invoice: ☐
From Multiple Quotes: ☐
Order Description:
Reference:

Ship-To Location Code:
Default Location Code: 1
Location Description: Central warehouse - Seattle
Deliver By: 7/11/2016
Expected Ship Date: 7/11/2016
Ship-Via Code: CCT
Ship-Via Description: Cross-Country Trucking Lines
Shipment Tracking Number:

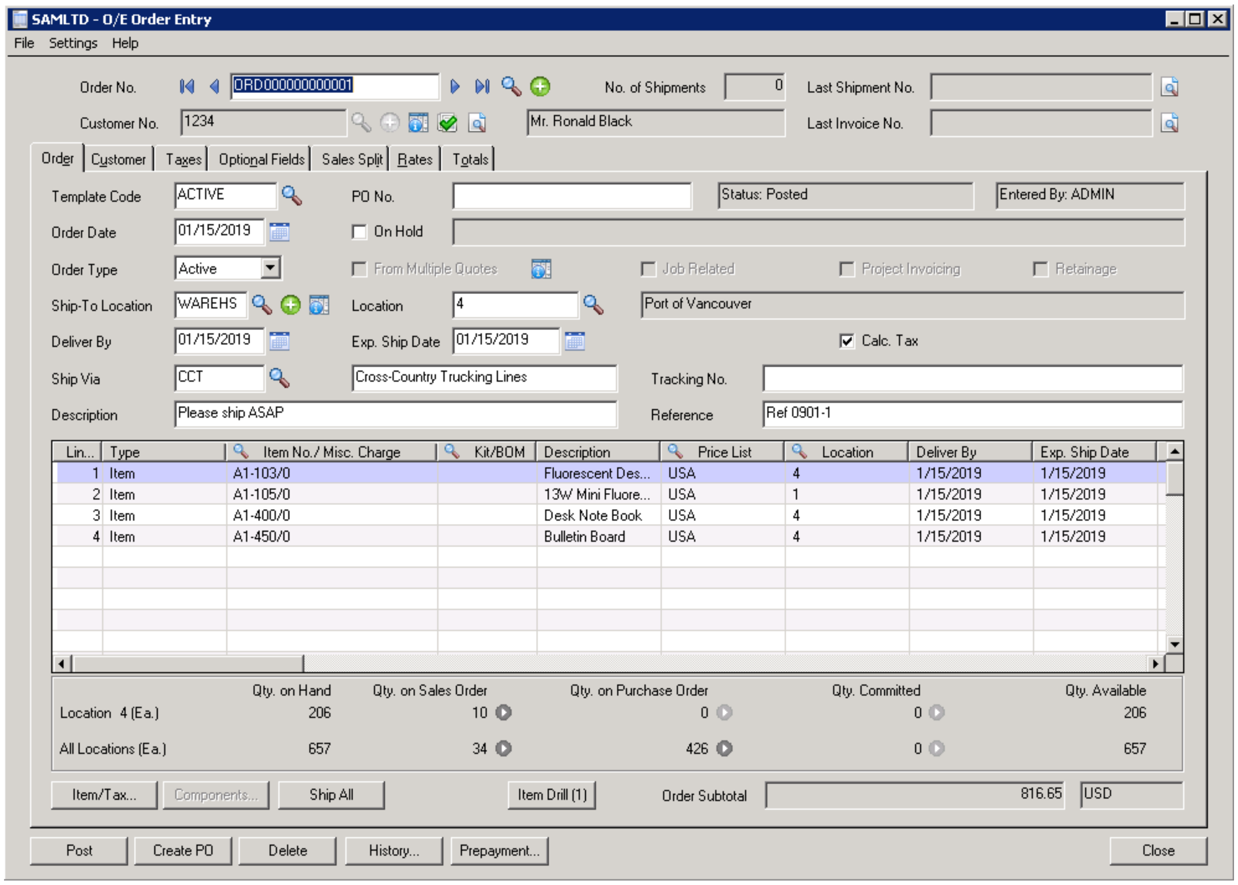
Line Number	Type	Item Number/Misc...	Kit/BOM	Description	Price List	Location	Deliver By	Expected Ship...	Quantity Ordered	Order UOM	Order
1	Item	A1-103/0		Fluorescent Des...	WHS	1	7/11/2016	7/11/2016	1	Ea.	lbs.
2	Item	A1-105/0		13W Mini Fluor...	WHS	1	7/11/2016	7/11/2016	1	Ea.	lbs.

Page 1 of 1 — 1 - 2 of 2 items

	Quantity on Hand	Quantity on Sales Order	Quantity on Purchase Order	Quantity Committed	Quantity Available
Location 1 (Ea.)	70	15	80	0	70
All Locations (Ea.)	378	23	510	0	378

Order Subtotal: 44.55 USD

A Sage 300c web screen for entering orders in a web browser.


SAMLTD - O/E Order Entry

File Settings Help

Order No.	ORD000000000001	No. of Shipments	0
Customer No.	1234	Mr. Ronald Black	
		Last Shipment No.	
		Last Invoice No.	

Order | Customer | Taxes | Optional Fields | Sales Split | Rates | Totals

Template Code	ACTIVE	PO No.		Status: Posted	Entered By: ADMIN
Order Date	01/15/2019	On Hold			
Order Type	Active	From Multiple Quotes	Job Related	Project Invoicing	Retainage
Ship-To Location	WAREHS	Location	4	Port of Vancouver	
Deliver By	01/15/2019	Exp. Ship Date	01/15/2019	✓ Calc. Tax	
Ship Via	CCT	Cross-Country Trucking Lines		Tracking No.	
Description	Please ship ASAP			Reference	Ref 0901-1

Lin...	Type	Item No./ Misc. Charge	Kit/BOM	Description	Price List	Location	Deliver By	Exp. Ship Date
1	Item	A1-103/0		Fluorescent Des...	USA	4	1/15/2019	1/15/2019
2	Item	A1-105/0		13W Mini Fluore...	USA	1	1/15/2019	1/15/2019
3	Item	A1-400/0		Desk Note Book	USA	4	1/15/2019	1/15/2019
4	Item	A1-450/0		Bulletin Board	USA	4	1/15/2019	1/15/2019

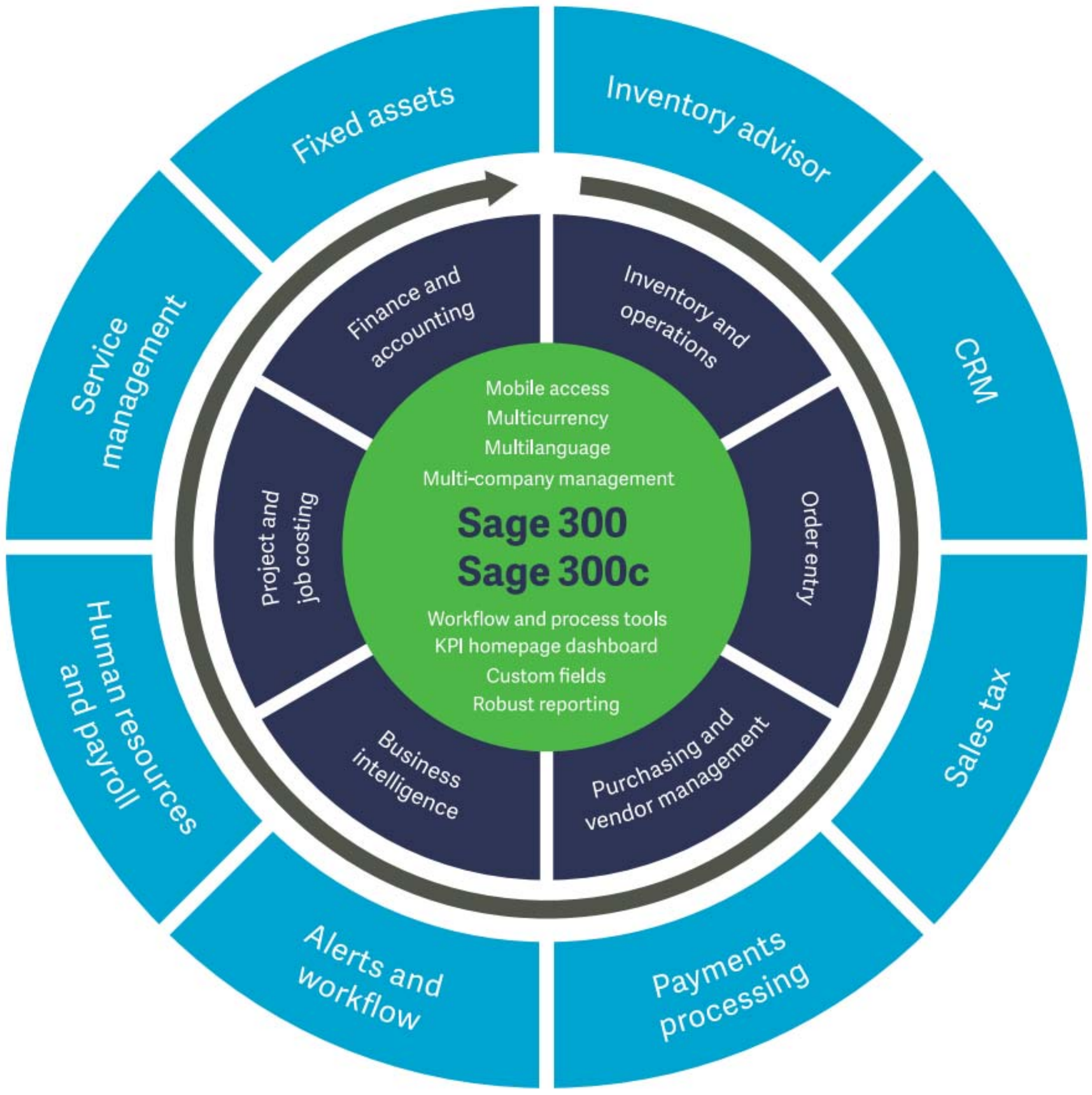
	Qty. on Hand	Qty. on Sales Order	Qty. on Purchase Order	Qty. Committed	Qty. Available
Location 4 (Ea.)	206	10	0	0	206
All Locations (Ea.)	657	34	426	0	657

Order Subtotal 816.65 USD

A classic Sage 300 screen for entering orders using a Windows application.

Other kinds of interfaces to the business logic, such as macros and import/export features, communicate with the business logic through a Common Business Logic Interface (CBLI), which has various components (API, XAPI, COM, .NET). This wealth of interfaces allows programs written in most programming languages to interface with Sage 300, including Visual Basic, Delphi™, Java®, JavaScript™, Perl, C#, C/C++, and J#.

This wealth of interface services allows integration with an expanded family of end-to-end enterprise products, allowing Sage 300 to expand beyond the usual roles of accounting modules. All these other applications are tightly integrated with the core accounting modules by interacting with the CBLI.

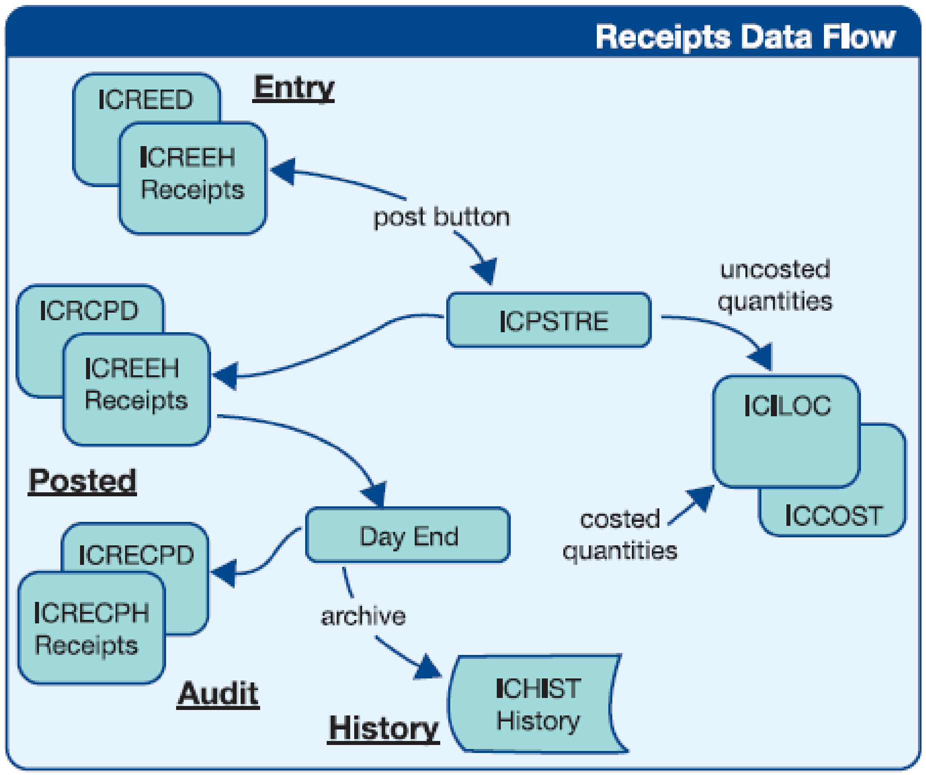


Sage 300 and Sage 300c are at the core of a family of connected services and end-to-end applications.

Sage 300 supports Microsoft SQL Server and the Sage 300 business logic does not depend on the peculiarities of a single database version. Because all database access is abstracted into a Common Database Services Interface (CDSI), adaptation to a new database version can be done quickly in one place (the database driver), with less detailed testing of the whole application required.

Object-oriented design

Sage 300 business logic is implemented as a collection of more than 750 objects that communicate with one another in a common object repository. The following example shows the objects that are involved in receiving inventory items and the messages they pass to one another.



Sage 300 business logic objects are implemented on a common communications framework by "major" function. This framework ensures consistent interaction among all objects, and the "large" object size allows complete business functions to be referenced, which simplifies customization.

Protocols ensure maximum object flexibility

A well-implemented set of objects follows established protocols—that is, they behave in a consistent and uniform way. The objects are constructed from a common "template" of business logic that all objects share, which ensures that protocols are consistent.

If new protocols are necessary, **it is usually possible to simply change the common template and re-instantiate all objects with the enhanced template.** New protocols are used to connect to new technologies and standards.

This organization creates incredible leverage:

- To make all Sage 300 objects behave as .NET objects requires only that the core protocols are .NET enabled.
- To make all relevant Sage 300 objects managed by the Distributed Process Server (which can schedule and deploy objects on multiple machines) requires only that the core protocols be linked.
- To make all Sage 300 objects XML enabled requires only that the core protocols are XML enabled.

We can continue:

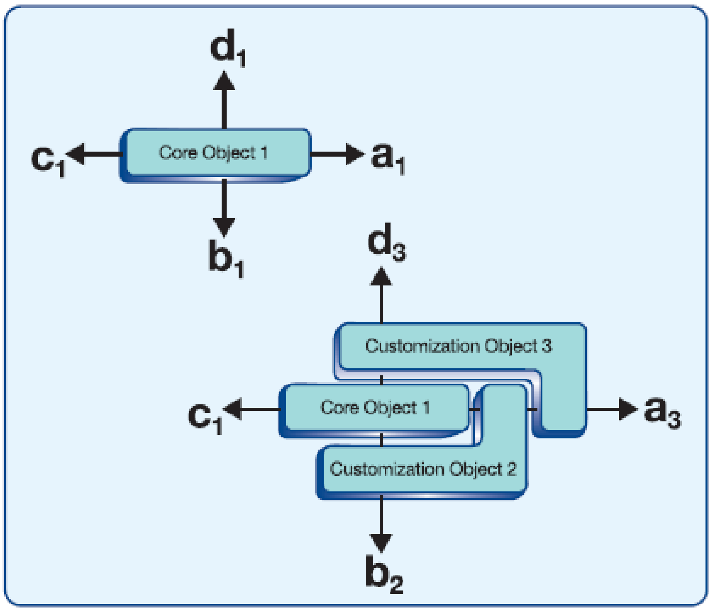
- Every single business entity is accessible through the Sage 300 Web API (RESTful Services).
- Every single business logic object is Internet accessible through the Sage 300 SOAP Web Service.
- Every single business logic object can be driven by VBA.
- Every single table can be exported or imported.

As new technologies and standards are constantly introduced, the consistency and uniformity of an object-oriented design is a key element of any architecture.

Simple, stable, future-proof customization

In an object-oriented design, objects inherit functionality from one another by "sub-classing" functionality and modifying part of the behavior. Customization is easy, with a wide range of development and customization tools available to manipulate objects.

Modifying core behavior in Sage 300 is simplified through the availability of third-party software products designed for this purpose. **The inheritance model used in the Sage 300 Desktop allows multiple products to modify the core behavior of Sage 300 objects**, with each third-party product behaving as though it is the only third-party product present. The customizations are also safe, because none of the code of the original object has changed.



Multilevel inheritance supports safe, flexible customization.

Sage 300 business logic objects support multilevel inheritance, providing customization options without jeopardizing the integrity of the core business logic.

In this example, Core Object 1 is part of a program and communicates with other objects through four messages—a, b, c, and d.

When another object, Customization Object 2, subclasses Core Object 1, it modifies the behavior of two of Core Object 1's messages, a and b. A third object, Customization Object 3, can sub-class Core Object 1 and Customization Object 2, modifying messages a and d.

Large objects are ready to deploy anywhere

What does it mean for an object-oriented design to be implemented at the right scale? It means that objects are the right size.

As an analogy, consider the major components of a car: engine, chassis, transmission, axles, and so on. Smaller objects like bolts, washers, and wiring are more numerous than these components, but less functionally significant in the overall design. In the same way, Sage 300 comprises millions of lines of code, but only slightly more than 600 objects.

Other business management software products are organized into literally thousands of objects, requiring much more effort to manipulate because each high-level task involves patching together ten or more small objects with snippets of code.

When large business logic objects are used, **objects represent real, high-level application tasks, such as posting to a general ledger.** This high-level functionality is available to any user or third-party developer who wants to customize—without the need to modify the actual business logic code. Large objects that represent real application tasks can also be automated by products such as the Process Server.

Design flexibility and the hosting paradigm

The Sage 300 architecture is **designed for flexible, scalable, and cost-effective deployment in a hosted environment.** A key cost factor in the hosting scenario is how many independent companies can be run on a single server and managed easily from a central point.

The Sage 300 architecture has a unique combination of design features that make it ideal for hosting:

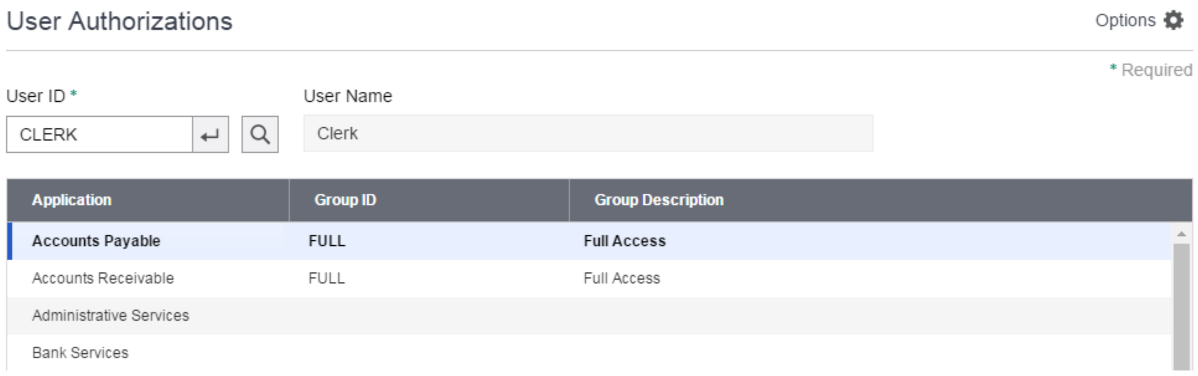
Sage 300 design feature	Sage 300 hosting advantage
Site directories separate unrelated companies running on the same server. Each company identifies its own security and access rights for groups of users.	Use a single server, whether there are 50 users in one company or 50 companies with ten users each. Sharing a server does not mean restrict companies to a single security methodology.
Multiple versions of the product can be installed on the same machine. Installation on a machine does not activate the version for all companies at once. Users or consultants can activate a version one company at a time.	As the software moves from one version to the next, conversions on a single server machine can be phased as convenient. Companies may therefore control the timing of a conversion process.
Language is identified for individual users of a company. Sage 300 currently supplies product translations for simplified Chinese, traditional Chinese, French, and Spanish. For Sage 300c web screens, user and browser language settings determine screen and message languages.	A single server machine can be used for companies in different countries or to run a company that has multiple user language requirements on the same machine.

Single-server support for multiple companies and security systems

In a hosted environment, Sage 300 **supports setting up multiple companies on a single server**, each with its own security settings.

If business management software does not support this, each client must be set up on a separate server. This is not a true hosting model, but rather a managed services model. The problem with this approach is that it limits fail-over and load balancing flexibility, which are key high-availability factors.

In a high-availability hosting environment, large, redundant network storage devices are used to balance loads so that multiple servers are available to process data for multiple companies. Multiple security systems on a single data store is a requirement for this setup. Sage 300 security is well suited to the hosting paradigm, as it **allows security to be defined by company and user on the same server.**



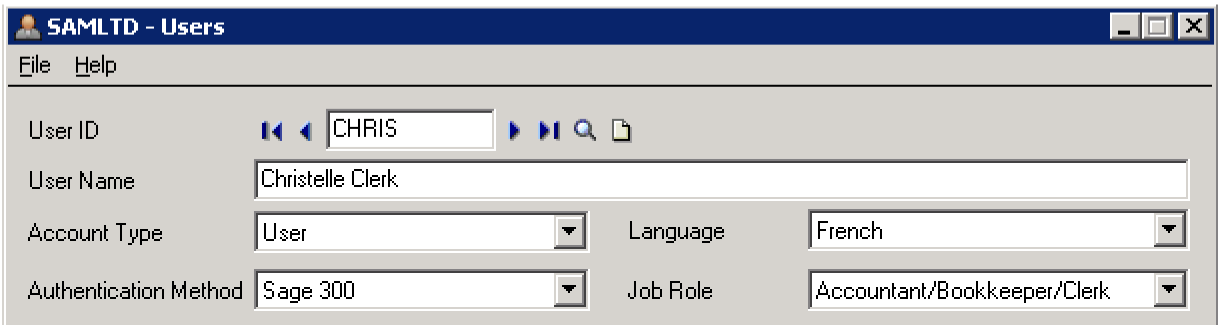
User ID *: CLERK
User Name: Clerk

Application	Group ID	Group Description
Accounts Payable	FULL	Full Access
Accounts Receivable	FULL	Full Access
Administrative Services		
Bank Services		

Sage 300 User Authorizations specify the applications that individual users can access.

User-selectable language settings

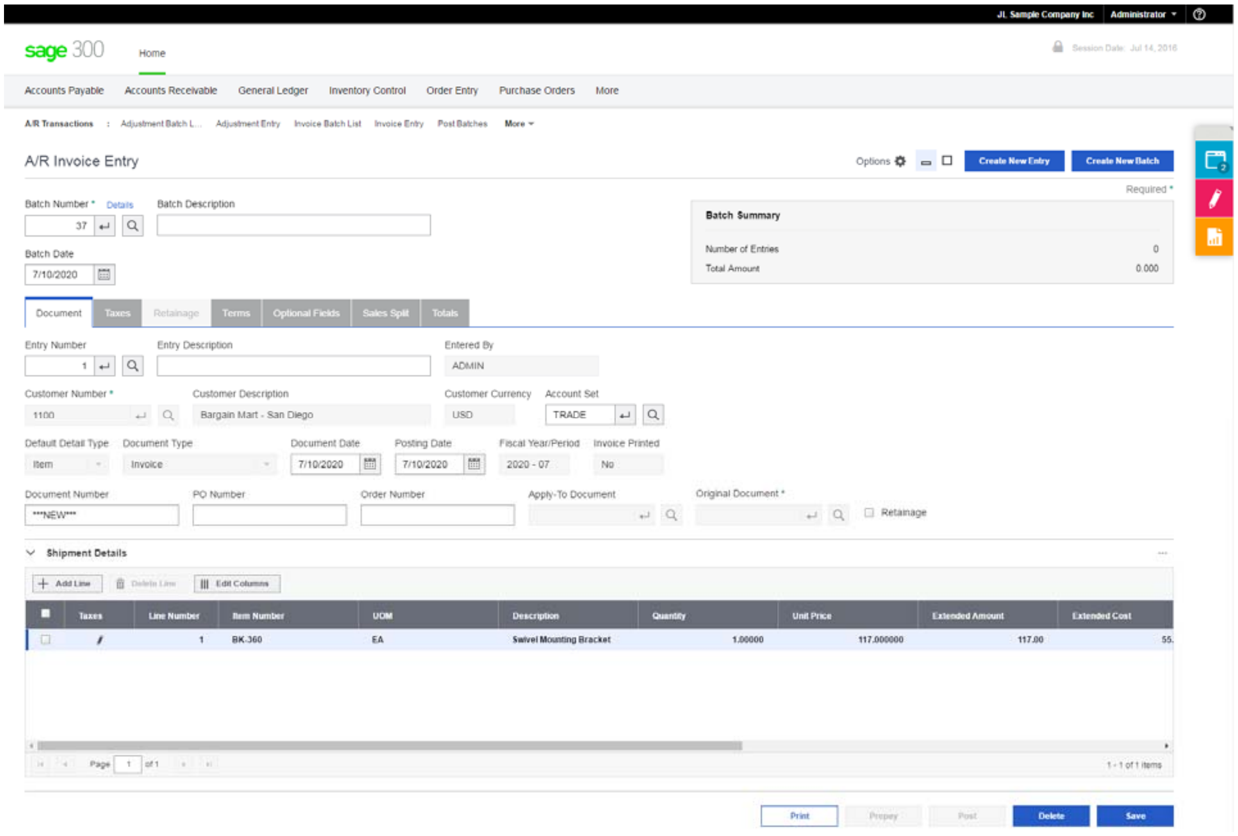
Several remote offices in different countries may all need to access the same data, using different languages, and all at the same time.



User ID: CHRIS
User Name: Christelle Clerk
Account Type: User
Language: French
Authentication Method: Sage 300
Job Role: Accountant/Bookkeeper/Clerk

An administrator can set up Sage 300 users within the same company to use different languages.

After initial setup, users access Sage 300 in their preferred language.



JL Sample Company Inc | Administrator

Session Date: Jul 14, 2016

Home

Accounts Payable | Accounts Receivable | General Ledger | Inventory Control | Order Entry | Purchase Orders | More

A/R Transactions : Adjustment Batch L... | Adjustment Entry | Invoice Batch List | Invoice Entry | Post Batches | More

A/R Invoice Entry

Options | Create New Entry | Create New Batch

Required *

Batch Number *: 37 (Details)
Batch Description:
Batch Date: 7/10/2020

Batch Summary
Number of Entries: 0
Total Amount: 0.000

Document | Taxes | Retainage | Terms | Optional Fields | Sales Split | Totals

Entry Number: 1	Entry Description:	Entered By: ADMIN				
Customer Number *: 1100	Customer Description: Bargain Mart - San Diego	Customer Currency: USD	Account Set: TRADE			
Default Detail Type: Item	Document Type: Invoice	Document Date: 7/10/2020	Posting Date: 7/10/2020	Fiscal Year/Period: 2020 - 07	Invoice Printed: No	
Document Number: ***NEW***	PO Number:	Order Number:	Apply-To Document:	Original Document *:	Retainage	

Shipment Details

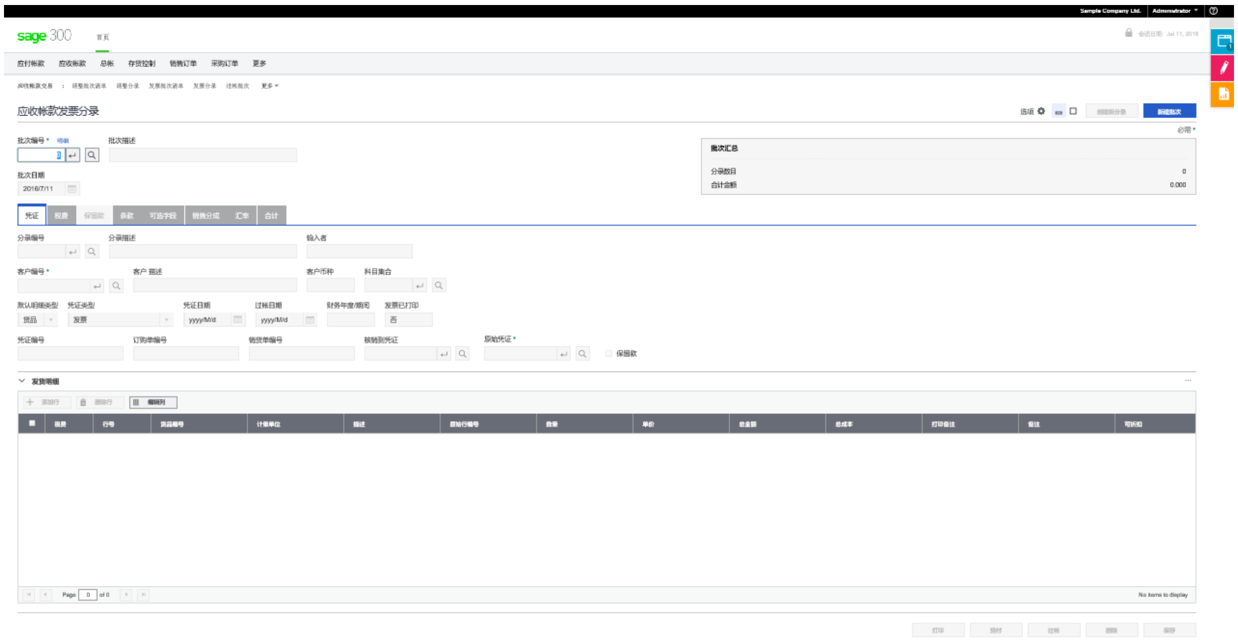
Add Line | Delete Line | Edit Columns

	Taxes	Line Number	Item Number	UOM	Description	Quantity	Unit Price	Extended Amount	Extended Cost
		1	BK-360	EA	Swivel Mounting Bracket	1.00000	117.000000	117.00	55.

Page 1 of 1 | 1 - 1 of 1 items

Print | Prepay | Post | Delete | Save

Sage 300c web screen for Invoice Entry in English.



Sample Company Ltd. | Administrator

会话日期: Jul 11, 2016

首页

应付帐款 | 应收帐款 | 总帐 | 存货控制 | 销售订单 | 采购订单 | 更多

应收帐款交易 : 调整批次清单 | 调整分录 | 发票批次清单 | 发票分录 | 过帐批次 | 更多

应收帐款发票分录

选项 | 创建新分录 | 新建批次

必需 *

批次编号 * (明细)
批次描述:
批次日期: 2016/7/11

批次汇总
分录数目: 0
合计金额: 0.000

凭证 | 税费 | 保留款 | 条款 | 可选字段 | 销售分成 | 汇率 | 合计

分录编号:	分录描述:	输入者:				
客户编号 *:	客户 描述:	客户币种:	科目集合:			
默认明细类型: 货品	凭证类型: 发票	凭证日期: yyyy/M/d	过帐日期: yyyy/M/d	财务年度/期间:	发票已打印: 否	
凭证编号:	订购单编号:	销货单编号:	核销到凭证:	原始凭证 *:	保留款	

发货明细

添加行 | 删除行 | 编辑列

	税费	行号	货品编号	计量单位	描述	原始行编号	数量	单价	总金额	总成本	打印备注	备注	可折扣

Page 0 of 0 | No items to display

打印 | 预付 | 过帐 | 删除 | 保存

Sage 300c web screen for Invoice Entry in Simplified Chinese.

Conclusion

The architecture of a software product determines how well, or how poorly, it fits into the software landscape of industry-standard technology, as well as existing and evolving technology and business paradigms.

Superior business management software, such as Sage 300 and Sage 300c, is organized into layers, with a clear separation of core business logic from interface and database services. This separation of layers ensures that **functionality can be customized simply and reliably without affecting core services.**

The benefits of a strong architecture and good implementation are that the resulting product will:

- Stand the test of time.
- Embrace industry-standard technology quickly and naturally.
- Customize easily to fit the specific needs of your business.
- Deploy flexibly to new paradigms.
- Scale to the changing size of your business.

A superior architecture is not created overnight. Creating a product with a superior architecture requires a massive investment over many years. **Sage has been making that investment for decades, and will continue to do so.**

Over the years, as software fads and trends have come and gone, the architectural message and object-oriented approach of Sage 300 have remained unchanged. The Sage 300 architecture stands the test of time and is ready to deliver value to your business for years to come.