IFS APPLICATIONS™
TECHNOLOGY AND ARCHITECTURE

TECHNOLOGY OVERVIEW
<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>IFS and Technology</td>
<td>2</td>
</tr>
<tr>
<td>Why is Technology Important?</td>
<td>2</td>
</tr>
<tr>
<td>Foundation 1 – the Open Evolutionary Platform</td>
<td>3</td>
</tr>
<tr>
<td>Industry Standard Tools and Technologies</td>
<td>4</td>
</tr>
<tr>
<td>Unix and Microsoft Windows Server Platforms</td>
<td>5</td>
</tr>
<tr>
<td>Component Architecture</td>
<td>5</td>
</tr>
<tr>
<td>Multi-Tier Architecture</td>
<td>6</td>
</tr>
<tr>
<td>Business Components</td>
<td>8</td>
</tr>
<tr>
<td>Advantages and Benefits of Components</td>
<td>9</td>
</tr>
<tr>
<td>User Interfaces</td>
<td>10</td>
</tr>
<tr>
<td>True Web Interface</td>
<td>11</td>
</tr>
<tr>
<td>Personal Portal</td>
<td>12</td>
</tr>
<tr>
<td>Powerful Windows User Interface</td>
<td>13</td>
</tr>
<tr>
<td>Mobile and Wireless Access</td>
<td>14</td>
</tr>
<tr>
<td>Reporting and Analysis</td>
<td>15</td>
</tr>
<tr>
<td>Open Integration</td>
<td>18</td>
</tr>
<tr>
<td>Categories of Integration</td>
<td>18</td>
</tr>
<tr>
<td>Open Integration Architecture</td>
<td>19</td>
</tr>
<tr>
<td>Activity and Business APIs</td>
<td>21</td>
</tr>
<tr>
<td>Access Providers</td>
<td>22</td>
</tr>
<tr>
<td>IFS Connect</td>
<td>24</td>
</tr>
<tr>
<td>IFS Connectivity</td>
<td>25</td>
</tr>
<tr>
<td>Electronic Data Interchange</td>
<td>25</td>
</tr>
<tr>
<td>Web Services</td>
<td>26</td>
</tr>
<tr>
<td>Proactive with Events</td>
<td>26</td>
</tr>
<tr>
<td>Replication</td>
<td>27</td>
</tr>
<tr>
<td>Business Logic Tier Technologies</td>
<td>27</td>
</tr>
<tr>
<td>Business Logic Implementation</td>
<td>28</td>
</tr>
<tr>
<td>Middleware</td>
<td>29</td>
</tr>
<tr>
<td>Business Document Reporting</td>
<td>30</td>
</tr>
<tr>
<td>Summary</td>
<td>31</td>
</tr>
<tr>
<td>Glossary</td>
<td>33</td>
</tr>
</tbody>
</table>
IFS APPLICATIONS™
TECHNOLOGY AND ARCHITECTURE

This document provides an overview of the architecture and technology used by IFS Applications 2002. It is written for an audience familiar with basic software, Internet, and ERP architecture concepts.

This technology overview focuses on the IFS approach to technology and the component and integration architecture of IFS Applications. Related white papers and technology overviews include:

- *IFS Technology and Architecture (White Paper)*
- *Security with IFS Applications (Technology Overview)*
- *Sizing and Scalability of IFS Applications (White Paper)*

IFS and Technology

Why is Technology Important?

Technology, that great engine of change, is rapidly bringing about new ways of conducting business. It drives competitive advantage, increased customer service and loyalty, faster time to market, and increased efficiency and profitability. The world is rapidly moving from connecting people with technologies like email and fax to connecting business processes across diverse application systems in entire business communities using Internet-based innovations. The very survival of companies requires keeping pace at an ever-accelerating rate.

IFS understands the critical nature of technology. Making the right choices in an ongoing technological evolution takes skill and careful planning. Many emerging technologies fail to deliver on their promise; for others, the implementation expense exceeds their value; as standards change, some technologies become obsolete.

At IFS, how technology is used is more significant than the technology itself. We don’t try to set the standards but conduct continuous research into emerging technologies and standards instead. As they mature and gain acceptance, they’re...
introduced into IFS Applications. Foundation1—IFS’ component architecture and technology platform—is designed to incorporate and deliver new industry standards and technologies in an evolutionary, cost-effective way.

IFS protects your business system investment by providing step-by-step, non-disruptive introduction of new technologies into operational systems.

**Foundation1 – the Open Evolutionary Platform**

IFS’ primary business is developing and delivering business application software. IFS leverages core technologies such as databases and programming languages from companies like Sun, Oracle, IBM, and Microsoft. These technologies are applied to solving business problems.

IFS began to develop Foundation1 in the mid-1990s as the strategic platform for the development, integration, and use of component-based business application solutions in the extended enterprise.

*The Role of Foundation1*

Foundation1 is different from technical platforms such as J2EE or Microsoft .Net Framework. Foundation1 is an umbrella platform, bringing together industry-standard technologies and tools with frameworks and tools of our own. Foundation1 makes use of multiple technical platforms which creates an optimal environment for the development and delivery of business applications. Foundation1 hides the complexity of technology, freeing application developers and customers to focus on business problems.

Foundation1 is more than just a development and runtime platform for IFS Applications. It includes the technologies and tools for integrating our applications with the world of other applications, other companies, and other people.
Ready for the Future

From its inception, Foundation1 was designed with the future in mind. IFS uses a layered architecture and extensive encapsulation to hide the complexity of underlying technologies and allow their introduction in a side-by-side, step-by-step evolutionary manner without disrupting core applications.

The architecture of Foundation1 has allowed the introduction of many new technologies. A web user interface was added using Microsoft Active Server Pages (ASP) technology in 1997. The IFS Personal Portal was added in 1999, and in 2001 the underlying technology was changed from ASP to Java Servlets, a part of the Java2 Enterprise Edition (J2EE) platform.

Some other technologies include the use of Unified Modeling Language (UML) in 1997, addition of the Java-based proactive event notification system in 1998, multi-site replication for distributed installations in 1999, and most recently, XML integration through web services.

These improvements were all accomplished in a non-disruptive way. IFS customers benefit from new technologies over time while keeping the core applications very stable and operational.

The technical platform on which IFS Applications is built—Foundation1—has been designed to evolve with time and technology. IFS Applications will offer the technology your organization needs, now and in the future.
Industry Standard Tools and Technologies

As information technology and business software mature, acceptance and use of common standards and practices are increasing. Universities are educating students on standard tools and technologies today. IFS recognizes and understands this change. Foundation1’s architecture allows the introduction of new standards and technologies as they become established. Technologies such as UML, XML, SOAP, Java, Java Servlets, and JSP did not exist five years ago, but are fundamental today to Foundation1 and IFS Applications.

IFS’ ongoing goal is to provide customer choice in terms of hardware, operating systems, application servers, web servers, and integration technology. To meet these goals, we are careful not to rely on vendor-specific functionality. Our encapsulation approach, using frameworks, allows different technology implementations without requiring changes to core applications.

Unix and Microsoft Windows Server Platforms

Building IFS Applications using open, standard tools and technologies gives customers the freedom to choose the deployment model that’s right for them and their budget. IFS supports most open platforms, including Microsoft Windows and major Unix variants. It’s possible to “mix and match” platforms since each tier in the architecture works through a standard network interface based on TCP/IP.

For extreme scalability and reliability, IFS supports deployment models ranging from small servers to large, high-performance database servers with multiple applications servers running in parallel.

Component Architecture

IFS Applications is built using the same principles employed by successful manufacturers—the production and assembly of components. Components are used at all levels of the architecture. Application solutions are built from business components, which in turn are built from smaller software components that implement the functional pieces of the application.

IFS began “thinking components” in 1994 at the time when object-oriented (OO) programming was making inroads into mainstream software development. Object orientation requires a high degree of adherence to interface standards for components (objects) to interact with each other. This allows change and evolution within a component without affecting other components. The details can be freely changed and improved without introducing instability in the rest of the system.
IFS realized that the benefits programmers saw in using OO could also be valuable for organizations developing and deploying business applications. This required the principles of object orientation to be applied to higher levels in the application design—and the concept of business components was born. A business component is the smallest installable and independently usable part of an application.

IFS also adopted other key principles of OO development, such as object modeling as a key element of software design. This led to a very effective approach to implementation, with inherent automation of much of the development process. IFS uses a combination of process modeling and UML object modeling to design IFS Applications today. The process model describes what the application does; the object model describes the components that make up the application and how they relate to each other.

Foundation1 was developed specifically to support a component approach. Migration of legacy systems to IFS Applications can be accomplished at the customer’s own pace by choosing only the business components they need at any point in time. Later, other components may be added without disruption to the components already in place. This results in a very stable operational system in spite of change. Your system can evolve in an orderly way without the “big bang” impact a non-component-based monolithic system would present.

**Multi-Tier Architecture**

A key to successful componentization is the use of a layered, multi-tier architecture. This is essential to developing standard component interfaces. Each layer has its own special job to do and can do it in a standard way.
Each tier has its own software components. These components are all derived from the UML design model, but are implemented using different technologies and programming languages suited for their purpose. The business logic tier contains software components that implement business entities and activities. Some presentation tier software components are implemented as Java Servlets and others as Windows forms, for example.

Data Storage
The data storage tier is based on the relational database model. This is the prevalent storage technology of today and for the foreseeable future. The database server is configured so that no data may be modified except by the business logic. This guarantees transactional integrity and protects the data from “back door” modification. Business logic (business rules) decides what may be stored or modified.

Business Logic
The business logic tier is the heart of applications. It implements business knowledge, functionality, and processes. It’s also where automated activities are performed; automated activities don’t need a presentation layer since there is no user interaction. This layer provides two application program interface (API) types: Activity APIs and Business (Biz) APIs. Activity APIs are conversational and transactional in nature and are used to perform individual activities in a business process. BizAPIs provide business and data services to other applications, using an XML document/messaging metaphor. BizAPIs are the key element in providing external IFS Application interfaces, such as web services, in conjunction with open integration.

User Interface Tier
The user interface tier provides interaction with human users and other client-side applications and devices. A server-side component, represented by business logic and data storage, may serve many different types of user interfaces, such as a Windows desktop, web browser, or mobile device. This is achieved by the component and standard interface approach.

Access providers are client-side libraries containing all the functionality and program interfaces needed for easy-to-access business logic by applications, devices, and user interfaces. They are written in the native language of each supplied access provider type—Java, COM, and .Net. The underlying middleware is
encapsulated, thereby removing technical complexity for developers while reducing implementation time.

Microsoft Office applications can call application business logic using simple Visual Basic code and the COM access provider, for example. Many devices and applications will provide their own interface software, often called a plug-in. Access providers deliver everything that is needed to integrate those devices and applications with core IFS Applications.

Life Cycle Management
Life cycle management is a term with many broad application and industry meanings. The context here applies to Foundation1 and the logistics of developing and deploying business applications.

The life span of an application can vary tremendously. Well-written applications that continue to meet a company’s needs may last for years, even decades. Applications that can embrace new technologies and evolve over time may in fact have indefinite life spans.

An application goes through many phases during its lifetime—from design and development to packaging and internationalization to deployment and operation. Foundation1 tools and processes support every step of the way. Administration of the operational system—adding users, setting up security rules, tailoring the configuration to specific needs, and day-to-day operation—is accomplished with IFS tools. When it’s time to perform maintenance or upgrade to later functionality, IFS tools are ready to do that job as well. IFS tools also help with some of the work of administering other software that is part of the overall solution—work such as Oracle and web server administration.

While very interesting, it’s beyond the scope of this paper to go into more detail about these many tools and processes.

IFS Connect
IFS Connect is IFS’ framework for integration and Electronic Data Interchange (EDI) with external business processes and applications. IFS Connect publishes BizAPIs through multiple protocols and formats—for example as web services. IFS Connect is bi-directional and is also used by IFS Applications to call external systems.
Business Components

The previous discussion showed the high degree of component granularity at the software component level. To make installation and administration easier, software components from all tiers are packaged into cooperating, collaborating business components. Business components can be installed and upgraded separately, enabling step-by-step implementation.
Once installed, business components may collaborate to perform business processes and activities. Foundation 1 allows a component to be designed for dynamic collaboration with another but still function correctly when used on its own. This allows independent development of components that form an integrated solution when used together. Many IFS Applications solutions use this technique to provide a more complete, integrated solution if additional components have been chosen. This provides the best of two worlds: highly granular components for development, deployment, and maintenance with richer functionality achieved by collaborating business components.

Advantages and Benefits of Components

Every part of the system is based on the component idea, yielding many benefits to the customer as well as to IFS. A brief summary:

- Individual components can be installed, upgraded, or replaced. This means, for example, that a single BizAPI, web page, or integration interface can be deployed independently.
- Customization of software for specific customer needs is easier since every component is built using a standard structure.
- All user interfaces of a given type have the same look and feel and navigation approach since they are all built in the same way.
- Security is inherent in the IFS architecture and development methodology, ensuring a consistent and reliable system.
- Large-scale software development projects may be undertaken with a high degree of parallelism across multiple development sites.
- Developers only need to understand the architecture—not the technology—to effectively work on different applications. They are free to focus on the business problem.
- Applications are easier to understand since they are built the same way.
- Reliability is greatly improved with standard interaction between components. Code is written and tested in small increments.
- Maintenance is simpler. Parts needing repair or replacement are isolated and may be easily changed without affecting others.
- Migration of legacy systems to IFS Applications can be accomplished at the customer’s own pace and in an orderly way. The customer is free to choose only the components that are needed and can add more components in the future without disruption.
User Interfaces

The purpose of a user interface is to help users perform their tasks in an efficient and effective way—in other words, to help them interact with and utilize the power of a great application. A single interface type is no longer sufficient to meet the needs of all users in an enterprise. Expert users will spend much, if not all, of their working day interacting with the system. Others may use the system only occasionally. Many will use the system from their own desktop. Others are more mobile and may want to use the system remotely, perhaps via the Internet. Application users may include employees of customers and partners as well. One of the most dynamic areas of technology today is the emergence of new mobile and wireless technologies such as PDAs, hand-held computers and the like.

IFS provides multiple user interfaces. The usability and performance of each interface type is optimized to meet the unique needs of users and their roles. The metaphors and design principles that are characteristic of each interface type are used. The web interface looks and behaves as web users would expect, using the browser metaphor and hyperlinking as the main instruments to navigate and perform tasks. Creation of a true web interface took more effort than just emulating the Windows interface in the web browser. The level of usability we wanted could not have been achieved in any other way.

User interfaces benefit from the component-based approach providing standard look and feel, and navigation for a given interface type. An improvement in design standards can easily and quickly be made across the entire suite of application user interfaces.

User interfaces allow a high degree of personalization by application users. Stored user profiles reflect these individual choices and settings.

User interface tier.
As discussed earlier, the multi-tier architecture places business logic in a separate, server-based layer, freeing the presentation layer (user interface) to focus on delivering the best possible human interface. This allows a “thin” client with no additional processing beyond presentation requirements. New presentation technologies, such as mobile clients, may be brought to market faster, therefore providing competitive advantages to IFS customers.

True Web Interface

IFS Applications provides web-based user interfaces presented in a browser. Combined with the IFS Personal Portal™, this interface is intended for anyone who needs a more intuitive, easier-to-use presentation with no client application software installation needed. It is also the right interface for users who must reach the applications from remote locations via the Internet.

The IFS web interface solution is implemented using broadly accepted web servers such as Apache (for Windows and Unix platforms) and Microsoft Internet Information System (for Windows platforms). Web pages are implemented as Java Servlets and run with a Java server engine like Tomcat as the web application server. This yields very good performance since pages are “compiled” once by the engine and then delivered to users without further compilation. This is a 100% pure Java implementation. Netscape Navigator and Microsoft Internet Explorer browsers are supported. Custom web pages can be built with JSP technology, using the provided Java access provider and web framework for server interaction.
The portal concept swept upon the information technology scene in the late 1990s and has been well accepted by the industry. IFS has developed the Personal Portal solution to provide this new way of interacting with applications. IFS Personal Portal provides a solution that is a truly integral part of IFS Applications—not just a shell on top of it.

A portal is a web page, viewed in a browser, that contains portlets, each of which is another web page or application window. Put another way, a portal is a compact way of looking at and working with many things at one time—all in one browser window. An application portlet may contain navigation links to drill down to detailed IFS Applications web pages.

Portals are personal to each user. The application portlets to be included can be selected, tailored, and rearranged to suit one’s work needs. Default portal configuration settings may be defined for user roles.

IFS Personal Portal is implemented in Java and can interface with non-IFS products. One’s email, calendar, or other applications can be used from portlets.

A major strength of IFS Personal Portal (and supplied application portlets) is that displayed information is current information, giving an up-to-date view of business status and processes. In short, portals make users more efficient and help them utilize the power of IFS Applications.
Powerful Windows User Interface

From its inception, Foundation1 has provided a robust, full-function Windows desktop user interface based on the object-oriented IFS/Client Developer™. IFS developed an extensive class library of presentation objects for building application user interfaces. By use of the common class library, all IFS Applications business components benefit from standardization of navigation and application presentation.

The interface has undergone continuous improvement and expansion during the intervening years with the introduction of ActiveX objects, major appearance reengineering, and stronger security features.

The Windows user interface is the interface of choice for frequent application users.

The Windows interface can optionally be supported by servers with Windows Terminal Server and/or Citrix MetaFrame. All user sessions run locally on the server with presentation displayed remotely on a desktop, using Citrix client software, or in a browser.

Layout of the Windows user interface provides a high degree of productivity for data entry. The end user can reorder or hide fields and columns, change tab order, and specify default values and input masks. Integration with Microsoft Office tools provides “one-click” export of data to applications like Excel and Word. Interactive graphics objects visualize complex data in one look. Presentation objects allow individual user preferences to be selected and saved in the user’s profile along with the last object setting, such as size and location.
Mobile and Wireless Access

In 1999, IFS was the first ERP vendor to present a standard business application for a Personal Digital Assistant (PDA). When the IFS eService Order solution for Palm was released, it was considered a minor revolution among computer journalists and analysts. For IFS, it was just another way to present the system to end users. PDAs are becoming more and more usable for people who don’t work in an office; hence IFS provided the system on portable devices. The mobile area has seen astonishing developments with new platforms and devices. Pocket PC, Palm OS, Symbian OS and Linux platforms are recognized as market leaders today.

IFS mobile and wireless services is a general platform for mobile clients, designed for now and the future. The technical platform delivers predefined interfaces to mobile devices enabling a cost-effective way to create tailor-made solutions for IFS Applications. IFS provides standard applications for the most commonly needed solutions such as service orders, work orders, and barcode use in the warehouse. Access to business information can be online—via radio, GSM or GPRS networks—or offline with information synchronization when the device is recharged.

The mobile and wireless architecture uses communication middleware to interface devices to IFS Applications using IFS Connect. The middleware encapsulates the various device technologies and operating systems yielding a simpler, more effective integration.
Reporting and Analysis

The purpose of reporting and analysis is to produce information that can be used by employees to gain knowledge, empowering them to make good decisions and take appropriate action. But there are hurdles to overcome. Often the time and cost spent creating and executing reports exceeds the added value of the results. Traditionally prepared reports also have the problem of containing a lot of detailed information, but perhaps not the exact information needed at the time. You can’t easily drill down to investigate further details, and the useful life of the report is often quite short.

IFS overcomes these hurdles by making reporting and analysis a natural part of the application—right where the daily tasks are performed. Reports are quick and easy to set up making reporting and analysis feel like just another business activity. Information is retrieved in real time from live business data.

Informal Analysis and Reporting

Users often need a quick analysis of operational data without creating formal reports. The IFS Windows client provides this capability in each application form and table window—it’s a standard part of the framework. The user can define—and save for later reuse—selections (queries), and groupings and sums on fields (columns). The layout of the window can be tailored as well with control over which columns are visible, the order in which they are shown, and the sort order (direction) for the data. The layout can also be saved with the selection for later reuse. Think of this as a mini-report. The selections and mini-reports can be shared among users by having them added to administrator-assigned default profiles.

The results can be sent to other desktop applications—Excel, Word, and others—with a single click using the built-in Output Channels capability. No programming skills are required. Printing all or part of the results is also a one-click action.
All of the functionality described above is available in the web user interface, which offers another great advantage: the personal portal and portlets. Portlets are designed to bring information to the user in a summarized and easy-to-understand manner, often eliminating the need for formal reports.

**Output Channels**

Sometimes a better understanding of data can be achieved by further analysis using external applications such as Microsoft Excel. IFS Applications offers a unique solution with a one-click transfer of any selection or report directly into Excel or other external applications. The idea is as simple as it is brilliant. With a simple click of the mouse, a file containing the data—in semicolon-separated, HTML or XML format—is created. The configured application is started, and the file is loaded.

Because of the simple and flexible design, Output Channels work with virtually every application on the market. Output Channels are available with web and Windows user interfaces.
Quick Reports

Quick Reports allow power users and report developers to develop more advanced reports for distribution. Quick Reports can be defined using SQL (a query tool for creating SQL statements is included) or the more powerful Crystal Reports software from Seagate. A runtime license for Crystal Reports is included with IFS Applications. Quick Reports can use the operational data store as well as the information access layer (see below).

Crystal Reports is perhaps the world’s most widely used reporting software. It offers advanced multi-level selection and grouping functions, a wealth of built-in diagram types, a scripting language, sub-reports, and drill-down all the way to source data.

Ordering and viewing of Quick Reports—using SQL or Crystal Reports—are fully integrated into IFS Applications and are available through web and Windows interfaces.

Information Access Layer

Like all high quality ERP applications, the IFS Applications operational data store uses a highly normalized relational database, optimized for high-speed transaction processing. This isn’t always the best organization for reporting and analysis. Highly normalized data often make it difficult to find the relevant data and in the format requested by the users.

Information access layer—a business-oriented view for reporting and analysis.
The information access layer (IAL) is a set of information views designed to present key information. A view is business-focused rather than application-focused. It can span multiple applications and makes business reporting and analysis much easier. The IAL is designed to be used without the need for a highly skilled programmer.

The information access layer has been designed to work as a foundation for OnLine Analytic Processing (OLAP) analysis. OLAP concepts such as star schemas with dimension and measurement views are implemented in the IAL.

IFS has included an option to make IAL views persistent by taking a snapshot copy of the information view and saving it to a separate table for quick access. This persistence option allows separation of reporting and analysis activity from transactional operational data. Externally, a live or persistent copy will appear the same to users.

**Advanced Analytics with OLAP and Balanced Scorecards**

IFS Business Performance™ offers strategic performance management and control through the use of reporting, analysis and scorecard capabilities. The IFS reporting tool allows users to create reports quickly and easily. With the analysis component, users can run OLAP (OnLine Analytic Processing) cubes allowing them to "slice and dice" through volumes of data to determine trends and support key decision-making. Third-party OLAP tools such as Knosys ProClarity can be used to view cubes. The scorecard functionality allows organizations to implement performance management through the definition of key performance indicators (KPI), used in conjunction with industry-leading standard balanced scorecard methodology. Balanced scorecards can be created, viewed, and managed directly from the web interface.

IFS Personal Portal enhances communication of strategic and performance objectives, enabling users to easily access relevant business performance information in reporting, analysis and scorecard areas.

IFS/Business Performance methodology helps customers move from data to information to knowledge to action.
Open Integration

Technology is enabling a wave of integration today. New industry standards now make it possible for applications to communicate and work together just as people have in the past. Business processes are expanding to include many organizations in a networked business community.

Integration is needed at many levels. People need business-to-business portals to reach all the parts of a business network. Applications from different vendors need to communicate and share information to provide the most effective total solution. Messaging standards, format transformation, and brokering software such as Microsoft’s BizTalk Server are enabling applications to be connected in different and easier ways.

The IFS open integration architecture is the collection of Foundation1 technologies, frameworks, and standards that enable quick and easy integration between IFS Applications and other business software applications and devices. IFS provides ready-to-use integrations to many of the most important ones. The hard work of packaging and delivering technology has been accomplished in Foundation1 and open integration, making it much easier to create new application integrations. The complexity has been concealed (encapsulated), letting developers and customers easily implement integrations with almost anything. IFS will continue in a step-by-step way to provide ready-to-use integration solutions for the most frequently required transactions, external system interfaces, and devices.

Categories of Integration

Integration is about sharing business information between IFS Applications and other applications, inside or outside the enterprise.

Integrations with business applications fall into two categories:

- Interaction integration, defined as additional ways for users to interact with applications.

- Process integration, defined as message-based integration between applications for automating business processes and transactions.

Interaction Integration

This type of integration could be as simple as access to application information from productivity applications like Microsoft Office, or from a device that is unable to run browser or Windows user interfaces. The support provided for barcode readers is an example of device integration. Excel spreadsheets have been used
to create a simple order form that can use the COM access provider to create an order in IFS Applications. A more advanced example would be integration of functionality from an IFS application into a custom web site. A characteristic of interaction integration is that access to any part of the business logic may be required. A conversational style of working, with good user interfaces, is required for human interaction.

**Process Integration**

This type of integration automates transactions and processes between applications. It started with file transfers and nightly batch jobs and evolved to EDI, spawning an entire class of new products. XML, SOAP and web services are rapidly being accepted as the standard way of integrating applications. Integration is done synchronously (within a single transaction) as well as asynchronously (queued messaging) using the Internet’s global communications network or the enterprise’s intranet. Synchronous integration between applications enables outsourcing of individual activities of business processes. A financial system could outsource currency conversion to a web service provided by a bank anywhere in the world.

Process integration can be done point-to-point (between two connected applications) or through message brokers. The message broker approach requires fewer configured and managed connections. XML has emerged as the standard message format for integration. This doesn’t mean there is—or may ever be—a universal standard for the content of process integration messages. Messages must be translated (transformed) between different content standards (schemas). Message flow through a message broker must be managed. This has led to the emergence of a new generation of process integration tools like Microsoft BizTalk Server and WebSphere MQ Integrator. Most enterprise application integration (EAI) products are developing in this direction.

**Open Integration Architecture**

Open integration refers to the collection of elements in all parts of the Foundation1 architecture that support integration with IFS Applications. These elements work together in a synergistic way to provide complete integration solutions beyond the capabilities of any single element. The functions of these elements are the following:

- Activity APIs—the conversational, transactional interface to IFS Applications—support integration with other applications through the use of access providers.
• BizAPIs provide high-level interfaces to the main business processes in IFS Applications and are the key element in integration with external applications. They are used via access providers or IFS Connect.

• Access providers are client-side programming libraries—for Java, COM, and .NET—that contain the functionality and APIs needed for easy access to business logic.

• Access Plug-ins are IFS integrations—using access providers to reach IFS business logic—with plug-ins provided by third-party applications that have their own defined way of extending to other applications and data sources.

• IFS Connect is the primary framework for using BizAPIs from external systems and applications using many protocols and formats.

• IFS Connectivity provides functionality for exchanging records with IFS Applications using generic in- and outboxes.

• Electronic Data Interchange with IFS Applications uses IFS Connectivity, BizAPIs, and IFS Connect for interchanging standards-based EDI messages with external applications.

• Web services expose IFS BizAPIs for use by external systems using XML, SOAP, and http standards.

• IFS Event Registry enables a proactive approach by applications. Configurable actions may be associated with defined events—actions such as sending external notifications to interested parties and/or starting other processes when events have occurred.

• XML Report Formatter is an optional component that can feed any printed or archived business document or core report in XML document format to external output management tools.

• Replication services provide integration—using IFS Connectivity and optionally IFS Connect—between local or remote instances of IFS Applications.

Each of these technologies will be discussed in more detail below.
Activity and Business APIs

Foundation1 defines two distinct types of APIs: Activity APIs and BizAPIs. These are used internally by IFS Applications but are also available for use in integrations. Interaction integration will typically use Activity APIs. Process integration is best achieved through BizAPIs.

The Activity API is a fine-grained conversational transaction API used by clients to perform individual activities in a business process. A client, from this perspective, is a human user or software agent that performs activities such as creating orders and approving invoices. Activity APIs are always invoked synchronously. Core IFS Applications is based on the multi-layered architecture described earlier with business logic running on servers. Activity APIs are the interface to IFS Applications. User interfaces supplied with IFS Applications communicate with these APIs.

BizAPIs are units of application logic that provide business data and services to other applications. BizAPIs are not intended for human users but are rather intended to be used by other applications with no need for conversational-style communication. A document request/response metaphor is used instead. The documents typically correspond to common business documents such as orders, invoices, and price lists.
BizAPIs are used synchronously by direct call or asynchronously with inbound or outbound documents (messages) queued for processing.

BizAPIs are implemented in Java. They natively receive (or send) XML documents. The document content is defined by an XML schema. W3C XML Schema (XSD) May 2001 Recommendation and XML Data Reduced (XDR) schemas are provided for each BizAPI.

Access Providers

Access providers are client-side programming libraries that contain the functionality and APIs needed for easy access to business logic. Access providers encapsulate the underlying communications protocol and handle client-server protocol semantics such as authentication and error handling. Input and output may be XML documents or other supported API internal record formats.

IFS has packaged the access APIs into separately installable access providers so that any application, whether it uses an IFS client framework or not, can have full access to business logic. All access providers contain the same functionality, but the implementation is created specifically for each targeted platform. This makes it very natural to use them.

IFS supplies access providers for the following development platforms:

- Java 1.3
- Microsoft COM
- Microsoft .Net

The Java access provider is written in 100% pure Java. The COM access provider contains only COM interfaces and the .Net access provider is implemented in managed code.

The COM access provider offers a powerful form of integration between business logic and Microsoft Office applications. With some simple programming, macros provide access to business logic from within a Word document or Excel spreadsheet, for example. The access provider encapsulates the complex job of programming middleware client-server interactions. The macro could be built to do simple SQL queries for including information in a report, or could perform more complex manipulation of information through use of Activity APIs.

Although access providers are typically used by end-user clients, they are also useful when integrating applications in the server tier. For example, a custom web site built with Microsoft Active Server Pages could use the COM access provider to invoke an Activity API. For these situations, the access provider’s “trusted”
mode can be useful. In trusted mode, the access provider will not authenticate the user, trusting the caller (in this example the web server) to have done so. Among other things, this makes it possible to build solutions that can be used without requiring users to identify themselves.

**Access Plug-ins**

Many devices and applications provide their own standards for developing software components that extend functionality by interfacing with an external application. These components are often called plug-ins, add-ins, adaptors, etc. They are called Access Plug-ins in IFS Applications architecture. Not all applications require a plug-in—Microsoft Office can use the COM access provider directly through Visual Basic for Applications (VBA).

Access providers contain everything needed to build an Access Plug-in for an application (assuming the plug-in can be built using Java, COM, or .Net).

Access plug-ins are developed for a specific application; access providers are generic to an entire development platform. Plug-ins may provide additional functionality useful to the specific application.
IFS and IFS partners supply a number of access plug-ins. The WaveLink access plug-in, supplied with IFS Connect, is one example. WaveLink is a software package for integrating barcode readers.

**IFS Connect**

IFS Connect is a platform-independent open framework for a unified integration of business processes with external business processes and applications. IFS Connect exposes BizAPIs using multiple protocols and formats—as web services, for example. Any IFS Applications report, EDI message or event can be channeled through IFS Connect using BizAPIs. IFS Connect does not distinguish between these different information streams—they are simply XML messages to be sent to a destination. Messages may be sent electronically to another application, printed and sent using fax or the postal service, placed in a file, or sent by email; any or all of these options may be chosen—for a given message—using easy-to-configure content-based routing.

The openness and flexibility of IFS Connect are achieved through the concept of connectors. A connector is a Java software component that translates internal XML format and middleware IIOP protocol to an external format and protocol. The Mail connector translates to MIME format and SMTP protocol, for example. Connectors are hosted by a middle-ware based application server that provides services such as SOAP enveloping and XML transformation to and from other application formats. Transformations may be accomplished using XSLT, or programmatically using Java.
IFS Connect includes a number of ready-to-use connectors and a development framework for building your own connector. IFS Connect is based on Internet communication standards and includes connectors for http, https (with Secure Sockets Layer), mail (SMTP and POP3), IBM WebSphere MQ, SMS messages, Windows net messages, client socket messages, and file-based interchange. Connectors play a key part in Foundation1 support for application-to-application integration. SOAP and Microsoft BizTalk Framework 2.0 standard envelopes are provided. This is an active area of standards creation; IFS will continue to evaluate accepted standards and implement them where appropriate.

**IFS Connectivity**

IFS Connectivity is a component that provides functionality for applications to exchange records. The mechanism used is a generic inbox and outbox structure. EDI has been based on this structure for some time, often using third-party software as the delivery mechanism to and from other systems. The Connectivity component is also used to carry replication traffic between IFS system instances in a multi-site installation. IFS Connect can be used as a carrier for IFS Connectivity traffic; these components are tightly integrated.

**Electronic Data Interchange**

IFS provides frequently needed EDI messages as part of application solutions. These messages use IFS Connectivity In and Out Boxes for receiving and sending. Third-party EDI products have been used to transport messages to and from external target applications. EDI has become more generalized with the use of BizAPIs and the IFS Connect framework; a common XML representation of the business document can be transformed to different EDI standard formats. EDI messages can be delivered directly to external application interfaces using connectors, or through third-party EDI applications.

IFS Applications and IFS Connect work with most third-party products such as Microsoft BizTalk Server and Sterling Commerce.

**Web Services**

Web services are rapidly evolving into broad industry use with the adoption of standards and support from major technology vendors. Web services offer the promise of wide integration between applications running on different platforms, written in different languages, and using different architectures. The industry stan-
dard of simplicity—using XML messages within Simple Object Access Protocol (SOAP) envelopes and sent through http protocols—has enabled a level of integration that has not existed before.

IFS has added the technologies needed to deliver web services in Foundation1 and IFS Connect frameworks. BizAPIs are the focal point, the glue, between incoming web services requests (via connectors) and back-end Activity APIs (business logic). IFS has created all of its existing EDI messages in a web service form. Any BizAPI can be quickly configured as a web service.

It should be noted that IFS is not bound to only http (the standard defined for web services) as the sending and receiving mechanism. Any of the connectors may be used to send to, or receive from, BizAPIs.

Proactive with Events
Foundation1 provides an Event Registry where application events are defined. A set of standard events is available with IFS Applications. Events can be used for an immediate alert to people when the application has detected something needing human intervention, a critically low stock level, for example. This is a proactive action on the part of the application. In the example, a human user does not need to continually monitor conditions. Instead, the application does the job and tells the user when the condition has occurred.
This behavior relies on the ECA (event, condition, action) model. When an event occurs (stock falls below a set level), a number of user-defined conditions are evaluated (the stock item is a promotion item, for example). If the conditions are met, the associated action is executed. The number of actions that can be defined for a single event is not limited. A single event could result in a mail notification, an outgoing message to another application, and a call to another Activity API, for example.

Event actions are implemented as outbound BizAPIs. The full range of IFS Connect features is available to send notifications. A delayed purchase order, for example, might trigger an action to send a mail to the purchasing manager. Another example would be sending a SOAP/XML message to a web service in a supplier’s application to automatically place an order. Supported event action types include:

- Send mail
- Send SMS (for GSM mobile phone networks)
- Send Windows popup message
- Send outgoing SOAP message
- Send outgoing message through any IFS Connect connector
- Call an Activity API in the business logic

To make the solution even more flexible, users may subscribe to an event and define the action they desire. When event conditions are met in business logic, the event “fires”, or executes, the associated action(s).

Replication

Foundation1 includes support for replication services—integration between instances of IFS Applications, local or remote. Replication can be done on a broad, or very selective basis, with tools provided to define replication objects. Any business object can be a replication object. Replication can be configured to replicate immediately upon change of data or on a periodic scheduled basis. Different instances may participate, with an instance being designated as “master” for certain business objects, and other instance(s) as “slaves” for those business objects.

Replication may be done for multiple instances by using database links or via connectors provided by IFS Connect.

Replication is a process accomplished at the business logic layer. This allows it to be highly configurable, more selective, better audited and monitored, and much faster than using the database server's built-in forms of table replication.
Business Logic Tier Technologies

Enterprise applications have been evolving in recent years in parallel with technology evolution. The focus is shifting from traditional data-intensive ERP applications, used for operating a single enterprise, to integrating processes with other enterprises and customers. Information is being used in new ways to provide new levels of service; work can be done across interconnected business communities; the flow of ideas and products is accelerating. This next evolutionary step is being described as ERP II, ERM II or ECM.

Business Logic Implementation

While traditional core processes and transaction management will always remain a critical part of your business, there are new requirements for communicating, sharing information and access, and integrating business processes across many companies. For application developers like IFS, the technical requirements for application components are becoming more diverse. Some components must be optimized for database access, others for in-memory processing, and yet others for communication.

IFS doesn’t believe that one single implementation technology meets the diverse requirements of today. As business applications move forward, components of those systems will need increasingly different characteristics. This results in a requirement for combinations of technologies. Encapsulation between architectural layers, provided by Foundation1, makes it possible to construct business components with different characteristics while preserving commonality.

The server tier (business logic) is encapsulated behind access providers. Developers of user interfaces, or integrations with other applications, do not need to be aware of the technology used to build the Activity API or BizAPI they wish to use. Access providers are available for COM, Java, and .Net, giving full API access to virtually any other application. To provide even greater integration flexibility, IFS Connect exposes all BizAPIs through a multitude of protocols and formats, e.g., web services using http and XML.

PL/SQL Packages for Activity APIs

IFS has implemented business logic for database-intensive components according to object-oriented principles. Based on the UML design, PL/SQL packages and database views are generated. The views are used to read information; the packages perform updates and provide support for business activities. Each UML class definition becomes a PL/SQL package. The package itself is not an object, but acts as
an object handler that operates on stored objects—rows in the relational database. This approach is necessary to handle the enormous data volumes in a business application.

The packages execute inside the Oracle database server, which provides the runtime environment. This architecture is ideal for components requiring high-speed, frequent access to large amounts of data. It delivers high performance, scalability, and a very cost effective deployment footprint. It meets the requirements of a small company that needs only a few component solutions to get started, yet it can scale to the largest enterprise needing the entire suite of solutions with multi-site, multinational deployments on high-performance servers. Delivering this range of performance assures the smaller customers that their system will be able to scale up smoothly as their needs grow. Most Activity APIs are implemented using PL/SQL packages.

**Java Application Servers for Business APIs**

The IFS Java application server framework handles communications-intensive and integration-intensive parts of the application. Application logic runs in separate processes outside the database manager. IFS Connect and all BizAPIs, are implemented using this framework.

The application server framework is based on Common Object Request Broker Architecture (CORBA) middleware. The application logic is implemented in Java. Natively compiled code is used in critical parts of the framework for optimal speed. This technology can be deployed in a highly distributed way. Since servers are stateless, scalability is virtually linear from a single server to dozens of servers working in parallel.

The Java server framework provides full interoperability with Activity API software components, implemented as PLSQL packages.

**Middleware**

The key to high-performance reliable systems is dependable middleware that facilitates communication and interaction between components in a distributed system. There are two main types of middleware:

- Component middleware
- Messaging middleware

Component middleware is generally used for communication between tightly coupled components in the same application residing in the same physical environ-
ment. High-speed, high-reliability network and synchronous communication between objects is the default mode of operation.

Messaging middleware is generally used within applications to achieve loosely coupled asynchronous processing and queuing. Another, perhaps more significant, use of messaging is for communications between applications over long distances and unreliable networks.

IFS Applications uses Common Object Request Broker Architecture (CORBA) as its central component middleware. Communication between servers, and from clients to servers, uses the Internet Inter-ORB Protocol (IIOP), designed to optimize communication over the Internet. The use of CORBA gives benefits such as location transparency, which means that neither clients nor servers need to know the physical location of other servers.

Load distribution and fail-over capabilities are also provided by the middleware. This means that multiple “copies” of an application server can be started on the same, or different, physical hosts. The CORBA middleware can spread (balance) the load across these servers, and can also recognize that one has become unavailable and direct the work to the available servers. These factors are key elements in the high scalability and reliability that can be achieved.

Note that IFS does not use component middleware (CORBA) for communications between system installations. The middleware is not exposed to client and integration developers but is encapsulated by framework functionality available to those developers. The middleware is simply a private matter between servers or between access providers and servers.

While CORBA is the central component middleware used in IFS Applications, the Windows client uses Oracle Net middleware to access Activity APIs. Microsoft COM is used in some Windows-specific parts of applications in integrations with Microsoft products.

IFS Connect enables IFS Applications to communicate using industry-standard messaging middleware such as IBM WebSphere MQ. Outbound and inbound BizAPIs can be configured to send and receive messages using these products. The emulation of synchronous calls through a “send and wait for the response” mode of operation is also supported.

IFS provides deployment flexibility in your choice of Windows or Unix servers for the database back-end, the middleware, the application servers, IFS Connect components, and the pure Java Web server.
Business Document Reporting

Although XML and Internet transactions are becoming more important, many core business flow documents and reports must still be faxed or printed. IFS Applications provides a solid architecture for ordering, producing, archiving, formatting, and printing business documents and core reports.

Reports can be ordered by the user from within the application, created automatically by the business logic, or scheduled. Execution of a report can be immediate or take place in the background at a later time. Reports can be scheduled to run at a specific time on a daily or weekly basis. When the report is run in the background, the Event Registry automatically informs the person that ordered the report when it is ready. When ordered manually, reports can be previewed online before printing.

One critical aspect of core business reporting is report archival for later viewing, exactly as it looked when first produced. The IFS Application report archive can save both the report data (so that the report can be reproduced with the exact same information using a different layout) and a PDF copy (which includes the exact layout) of the report. The PDF copy also enables reports to be viewed with exact layout in an Intranet or Extranet solution.

Once produced, the report must be formatted and printed. This is the job of IFS Print Server. Multiple print servers can act as gateways to printers and can be placed anywhere on the network. Sensitive reports can be routed to a print server in a physically restricted area to protect classified reports from prying eyes.
The architecture of IFS Applications is also suitable for use with specialized output management solutions such as StreamServe. Through the Report XML Formatter, any report can be forwarded as an XML document to the IFS Connect integration framework. The full wealth of IFS Connect-supported formats and protocols can then be used to forward the Report XML document to another application or output management system.

Summary
The intent of this paper has been to present the technologies and architecture on which IFS Applications is built. Key points that were made are:

- **Foundation1** is IFS’ strategic platform. It is designed for—and is delivering—new technologies in an evolutionary step-by-step way. IFS is “technology neutral”, which reduces the risk to IFS and to our customers by not allowing strategy to be based on a particular technology or vendor. We introduce new technologies based on market demand and customer requirements.

- The component approach is used throughout. This allows customers to choose applications and add more applications incrementally in a non-disruptive way.
Multiple user interfaces—Windows, web, and mobile—to meet diverse user needs.

Wide choice of hardware/operating system platforms and vendors.

Complete life cycle tools and management processes.

True Java-based web user interface and IFS Personal Portal.

Total system and application security features.

Multiple technologies and frameworks embracing J2EE and .Net with strong CORBA middleware.

Solid operational data store and business logic for core applications, designed for high transaction volumes.

IFS Connect architecture, frameworks, and tools for integration of IFS Applications with other systems and applications to build a connected community of enterprises and customers.

Extensive integration built in between IFS Applications and Windows desktop applications.

Strong use of industry standards such as XML and SOAP.

Support of third-party formatting and rendering software for print and fax output.

Proactive applications with the Event Registry for notifications to appropriate parties.

Scalable solutions for the smallest to largest enterprises.

Electronic Data Interchange and web services solutions.

You have IFS’ commitment to always moving forward with technology in a fast-changing world. IFS Applications and Foundation1 are not just an ERP system. They constitute a complete set of business applications and methodologies based on the latest technologies to support the modern enterprise in all its needs.
Glossary

Access Plug-in: IFS integrations—using access providers to reach IFS business logic—with plug-ins provided by third-party applications.

Access Provider: IFS client-side libraries containing all the functionality and program interfaces needed to access business logic by applications, devices, and user interfaces. They are written in the native language of each supplied access provider type—Java, COM, and .Net.

Activity API: A fine-grained conversational transaction API used by clients to perform individual activities in a business process.

Apache: A freely available web server, distributed under an “open source” license and supported on Windows and most Unix platforms. Supported for both IFS web client and IFS Connect use.

ASP: Microsoft’s Active Server Page web technology for creating dynamic html content using server-side scripting (Jscript or VB script). ASP is supported by Microsoft’s IIS web server.

Asynchronous: A loosely coupled way of communicating between two software components using messages. The sender resumes processing as soon as the message is accepted and does not wait for a reply. Messages are normally placed in an inbound or outbound queue before delivery to the target component.

BizAPI: Business APIs provide high-level interfaces to the main business processes in IFS Applications and are the key element in integration with external applications. They are used via access providers or IFS Connect, are written in Java, and speak XML.

Microsoft BizTalk Server: BizTalk Server is Microsoft’s message broker for integrations. Applications desiring to integrate through BizTalk Server provide their own message schemas to BizTalk Server which then takes care of transforming formats between applications.

Business component: A business component is the smallest installable and independently usable part of an application. It represents a self-contained business application and is made up of software components at all levels of the architecture. Business components may dynamically discover and collaborate with other business components for broader, integrated functionality.
Business logic tier: The middle tier of the architecture where business knowledge, functionality, and processes are implemented and automated activities are performed. Provides two application program interface (API) types: Activity APIs and Business (Biz) APIs.

COM: Component Object Model—Microsoft’s strategic building block approach for developing applications. A middleware that provides communication between software components in a Microsoft infrastructure. Supported by an IFS access provider for communications between Windows-based clients and business logic.

Connector: A component of IFS Connect that supports outbound and inbound communications with external sources and destinations. A connector is specific to a given method of transporting a message—using http, mail, an external messaging system, a file, and others.

CORBA: Common Object Request Broker Architecture. An industry standard middleware for communication between software components in a distributed multi-platform environment.

Data storage tier: Persistent (permanent) storage for business object data based on the relational database model. No data may be modified except by the business logic tier.

EDI: Electronic data interchange is a standard format for exchanging business data based on ANSI X12 and closely coordinated with EDIFACT, an international standard. IFS provides EDI messages for common business transactions using IFS Connectivity, delivered using IFS Connect or third-party tools supporting EDI transfers. IFS’ EDI transactions, in conjunction with BizAPIs, may be exposed as web services.

Envelop: An industry standard XML “container” for other message types, typically another XML message. SOAP is a standard container for messages used with web services.

Event Registry: An IFS facility that enables proactive applications to send notifications in various forms when predefined events occur within applications. Notifications are configurable as actions. Users may subscribe to events.

Foundation1: IFS’ component architecture and technology umbrella platform which brings together industry-standard technologies and tools with IFS frameworks and tools. Multiple technical platforms for the development and delivery of business applications. It includes the technologies and tools for integrating IFS
Applications with the world of other applications, other companies, and other people.

**Host:** A physical hardware device (a computer) that runs an operating system and software processes. Traditionally called a server but that term now refers to software processes running on a host.

**http:** HyperText Transfer Protocol is the set of rules for exchanging files (text, graphic images, sound, video, and other multimedia files) on the World Wide Web. It is an application protocol which is normally sent using the tcp/ip communications protocol. The primary means of communicating between a client browser and a web server.

**https:** Https is a secure form of http using Secure Sockets Layer (SSL).

**IFS Connect:** IFS Connect is IFS’ framework for integration and electronic data interchange (EDI) with external business processes and applications. IFS Connect publishes BizAPIs through multiple protocols and formats provided by the supplied connectors. It provides for enveloping and transformation of messages.

**IFS Connectivity:** IFS Connectivity provides functionality for exchanging records with IFS Applications using generic in- and out-boxes. It is used by IFS Applications to send messages to BizAPIs or connectors.

**IIOP:** Internet Inter-ORB Protocol allows distributed programs, written in different programming languages and running on different platforms, to communicate over the Internet. IIOP is a critical part of the CORBA architecture. IIOP is highly optimized for network communications.

**IIS:** Internet Information Server is Microsoft’s web server. It provides for static html support as well as dynamic html support through Active Server Page (ASP) technology.

**Internationalization:** Internationalization, sometimes referred to as localization, is the process of providing support for multiple languages, currencies, and country-specific application functionality.

**J2EE:** Java 2 Platform, Enterprise Edition is a Java platform for enterprise computing designed by Sun Microsystems and other leading industry partners. It supports development and use of thin clients in a tiered architecture.

**Java Servlet:** Servlets are Java programs that add functionality to a web server like applets add functionality to a browser. A page request from the user’s browser will run the servlet directly, or by referring to it in another web page.
**JSP:** Java Server Page is a technology for controlling the content or appearance of web pages through the use of Java servlets, small programs that are specified in the Web page and run in a Web application server (such as Tomcat) to modify the Web page before it is sent to the requesting user’s browser.

**Message Broker:** The message broker model of application integration provides a neutral “third party (broker)” that has knowledge of formats required by the integrating applications. It provides queuing of messages for forwarding to the destination and transformation (mapping) of formats such that each application sees only its own formats. Microsoft’s BizTalk Server and IBM WebSphere MQ Integrator are examples of products that support the broker model.

**Microsoft .Net:** .Net is the Microsoft XML Web services platform for allowing applications to communicate and share data over the Internet, regardless of operating system, device, or programming language. It consists of a number of tools and standards for the development, deployment, and integration of .Net-based applications. IFS provides a .Net access provider for accessing business logic from client-side .Net-enabled applications.

**Open integration:** Open integration is the collection of Foundation1 technologies, frameworks, and standards that enable quick and easy integration between IFS Applications and other business software applications and devices.

**PLSQL stored procedure:** Oracle’s programming language (PL/SQL) for high performance transaction-oriented applications. A PL/SQL package may be stored and executed under control of the database server. IFS uses these for most activity APIs using an object-oriented design and implementation.

**Portal:** A portal is a web page, viewed in a browser, that contains portlets, each of which is another web page or integrated application window. It provides a compact way of looking at and working with many things at one time—all in one browser window—and may contain navigation links to drill down to detailed IFS Applications web pages.

**Portlet:** Portlets are configurable application “windows” or views positioned within a portal and contain information from IFS Applications. IFS portlets are written in Java and are designed to take advantage of the portal approach by presenting up-to-the-minute operational information. The application portlets to be included can be selected, tailored, and rearranged to suit one’s work needs.

**Replication:** Replication services provides integration between instances of IFS Applications, local or remote, using IFS Connectivity and IFS Connect. Individual
business objects can be configured to replicate between master and slave instances.

**Server**: A software process running on a physical host platform.

**SOAP**: Simple Object Access Protocol is a standard for exchange of messages between applications running in different environments. It is an “envelope” using an XML format and contains a payload in the form of an application XML message. It’s most common use is in web services and with Microsoft BizTalk server.

**Software component**: Software components are the functional parts of an application and appear on all tiers. They’re derived from the UML design model, but are implemented using different technologies and programming languages suited for their purpose. The business logic tier contains software components that implement business entities and activities. Some presentation tier software components are implemented as Java Servlets and others as Windows forms, for example. Business components are made up of software components.

**SMS**: Short Message Service is for sending messages to mobile phones that use Global System for Mobile (GSM) communication. GSM and SMS services are primarily available in Europe and Asia. IFS event actions include sending SMS messages.

**SSL**: Secure Sockets Layer is a commonly-used protocol for managing the security of message transmissions on the Internet. SSL uses public and private key encryption and includes the use of digital certificates. See “https”. May also be referred to as Transport Layer Security (TLS).

**Synchronous**: A tightly coupled way of communicating between two software components using messages. The sender suspends processing until a response is returned from the target component. Messages are passed to the target component (without going through a message queue). Also called the request-response mode of communicating.

**Tomcat**: Tomcat is a Java servlet engine (web application server) that supports both web pages built as Java classes (servlets) and pages built as JSPs. Tomcat is an open use server and supported by IFS for the Web Client solution and for IFS Connect use with the http connector. Tomcat may be run as a standalone web server and web application server, or as a plug-in to an IIS or Apache web server. Java servlets are compiled upon first use and retained for immediate reuse by following requests, yielding a high performance solution.
Transformation: The process of changing a message from one format to another. An XML order message from IFS Applications can be transformed to another application system’s order format, for example. IFS Connects supports optional message transformation using an XSLT or programmatically with a Java class.

UML: Unified Modeling Language is a notation standard for modeling real-world objects as an initial step in developing applications using object-oriented design methodologies. IFS designs all software components using UML.

User interface tier: The user interface tier provides interaction with human users and other client-side applications and devices. A server-side component, represented by business logic and data storage, may serve many different types of user interfaces, such as a Windows desktop, web browser, or mobile device.

Web service: Web services are a standards-based way of providing application-to-application integration between applications running on different platforms, written in different languages, and using different architectures. XML application messages are placed in Simple Object Access Protocol (SOAP) envelopes and sent through http protocols. XML schemas (XSD) are provided to define message contents.

WebSphere MQ: IBM’s messaging infrastructure for guaranteed one-time delivery between an extensive set of supported platforms. IFS Connect provides a connector for using MQ as an integration or messaging channel.

WebSphere MQ Integrator: IBM’s information broker for one-to-many connectivity with transformation, intelligent routing and information flow modeling across multiple business systems of different types.

XML: Extensible Markup Language, a formal recommendation from the World Wide Web Consortium (W3C), is a flexible, standard way to create common information formats and share both the format and the data on the web, intranets, and elsewhere. It is the natural evolutionary follow-on to html, which specified only presentation and interaction characteristics.

XSD: XML Schema Definitions are a standards-based way of describing XML documents. Content elements, data types, and other detailed information completely describe an XML document. A “published” schema consists of an xsd type document, which is itself an XML document.

XSL: eXtensible Style sheet Language describes how an XML document should be displayed, transformed to another XML document or other document type, and to
define the parts or patterns of the XML document. It is to XML what Cascading Style Sheets (CSS) are to html, but with much richer capability and power.

**XSLT**: XSL Transformations specify how to transform (change) an XML document into another XML document, a completely different type of document, or into html. It describes the mappings and actions used to transform a source document into a result document.

IFS and all IFS product names are trademarks of IFS. The names of actual companies and products mentioned herein may be the trademarks of their respective owners.