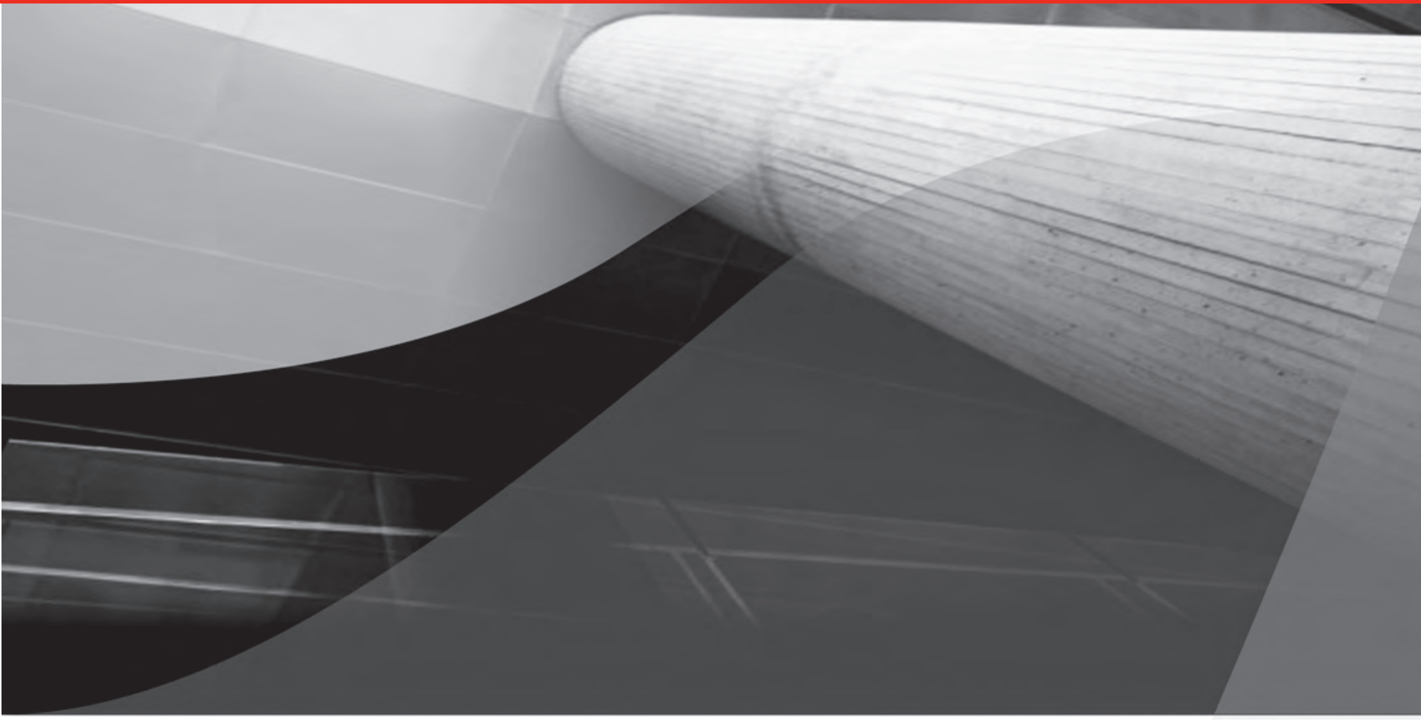
ORACLE



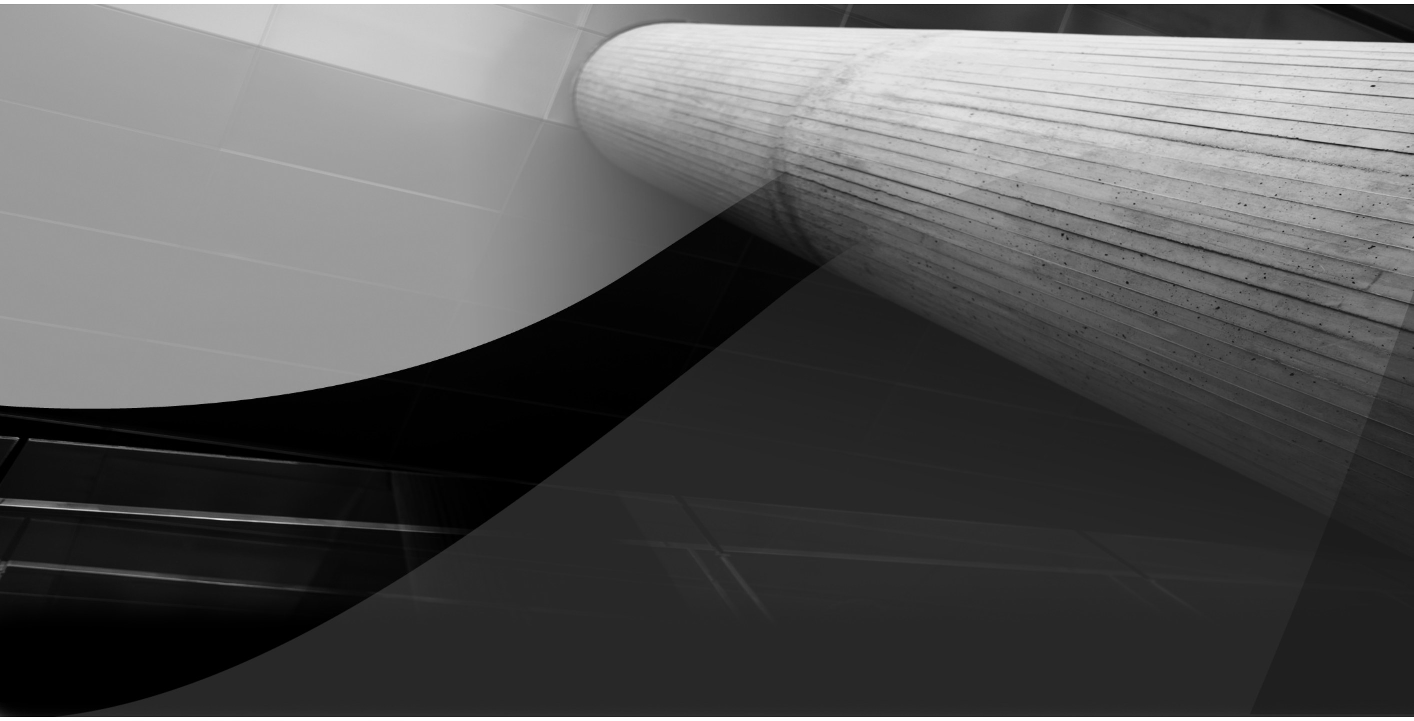
Managing Oracle Fusion Applications

ORACLE
FUSION APPLICATIONS

Best Practices for a High-Performance Enterprise Application Management Strategy

Richard Bingham
Senior Principal Technical Support Engineer, Oracle

ORIGINAL • AUTHENTIC
Oracle Press
ONLY FROM McGRAW-HILL



CHAPTER 2

A Fusion Applications Technical Overview

This chapter explains the general principles, technologies, and architectures upon which the Fusion Application features run. With literally thousands of Oracle products at its disposal, Fusion Applications leverages the latest technical advancements, product releases, expert functions, and comprehensive capabilities. Although most technology solutions are based on the latest releases of native product sets, Fusion Applications has also served as a catalyst for further extension and advancement. To cover the wide-ranging technical landscape, even at a high-level, this discussion is divided into five main areas:

- **Technical Architecture Fundamentals** Explains the basic technologies and principles used across the applications.
- **The Technology Stack** Looks at which Oracle components are used to deliver the structures explained in the preceding section.
- **A Processing Walkthrough** A simplified example of how users' common actions can trigger each part of the technology stack and how the components work together at runtime.
- **Coexistence Overview** How Oracle Fusion Applications can work with other enterprise applications right out of the box, using Oracle's prepackaged integrations.
- **The Extension Architecture** How configuration, personalization, and extension are all possible within Fusion Applications.

Fusion Applications uses a huge range of technologies, and, as such, it's impractical to think that you can achieve a deep level of understanding of every last piece straight away. This section provides just enough detail to help you form a suitable basis upon which a sensible and practical approach to applications management can begin. The concepts discussed here are further discussed in additional chapters and supplemented by the references in the Appendix.

Technical Architecture Fundamentals

This section provides an outline of the key concepts used throughout Fusion Applications, starting with the business users' view and drilling into each platform component from there. Although you may already understand some of these basic concepts, it's still worth your time to review them all to understand the Fusion Applications context.

The User Interface

As explained in Chapter 1, Fusion Applications emphasizes the evolution of the way in which traditional enterprise applications interact with their users. With its user interface flow engineered around Business Process Model (BPM) Level 4 tasks, it more closely mirrors the daily work involved in common job roles. This creates a user experience (UX) that is intuitive and visually attractive, and most importantly enhances productivity. It has been estimated that Fusion Applications has cut the number of clicks, fields, and forms that are normally required to complete a task by about 70 percent, quickly delivering significant efficiencies. The Fusion Applications UX architecture contains a few basic elements that are helpful to understand.

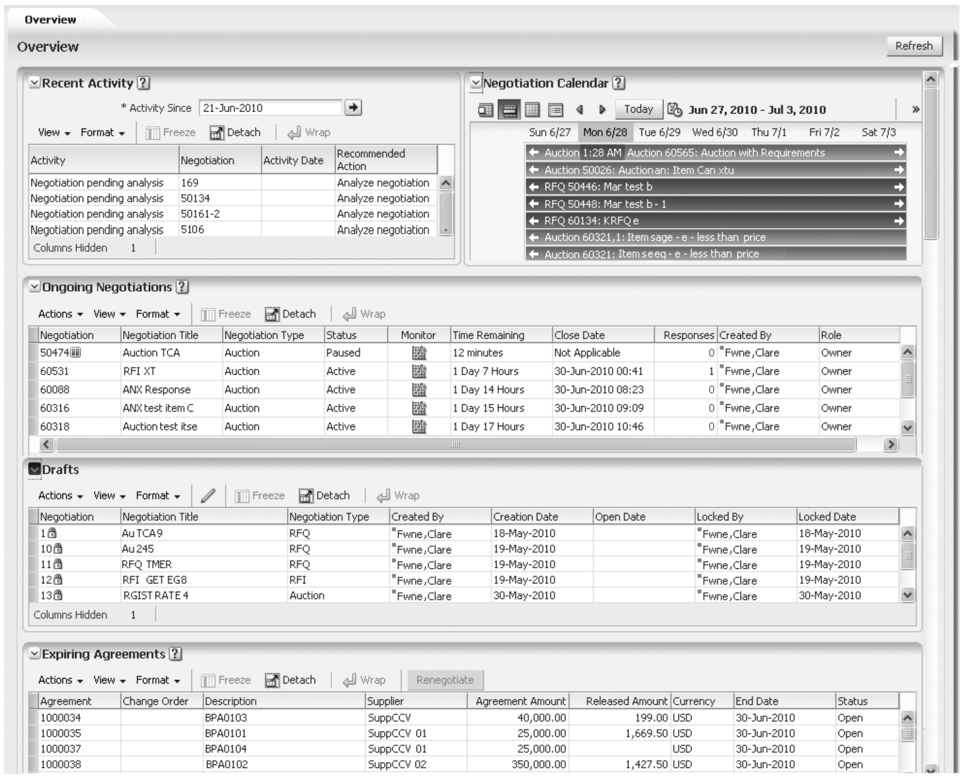
Dashboards

As with all modern enterprise applications, Fusion Applications is not focused solely on the speed of data input and transaction administration. Much of this has already been automated, and the focus is now firmly on providing business users with answers to two key questions to help users perform their jobs more efficiently:

- What do I need to know?
- What do I need to do?

Fusion Applications uses dashboard pages to provide each specific user, based on his or her functional role, with a variety of views (portlets) into the application. First, a general welcome dashboard includes information from all across the application. In addition, each specific functional area or business process (such as Finance Manager or Procurement Buyer) will have its own dashboard (Figure 2-1) for reviewing a more focused set of information, where it's possible to take direct action.

Top-level welcome dashboards commonly contain two specific components, along with tables of summary data and charts. The first component is a *Worklist* and provides a summary of all pending notifications, approvals, and to-do action items in one central place, all based on the



Recent Activity

* Activity Since 21-Jun-2010

Activity	Negotiation	Activity Date	Recommended Action
Negotiation pending analysis	169		Analyze negotiation
Negotiation pending analysis	50134		Analyze negotiation
Negotiation pending analysis	50161-2		Analyze negotiation
Negotiation pending analysis	5106		Analyze negotiation

Columns Hidden 1

Negotiation Calendar

Jun 27, 2010 - Jul 3, 2010

Sun 6/27 Mon 6/28 Tue 6/29 Wed 6/30 Thu 7/1 Fri 7/2 Sat 7/3

Auction 1:28 AM Auction 60565: Auction with Requirements
Auction 50026: Auctionan: Item Can xtu
RFQ 50446: Mar test b
RFQ 50448: Mar test b - 1
RFQ 60134: KRFQ e
Auction 60321,1: Item sage - e - less than price
Auction 60321: Item seeq - e - less than price

Ongoing Negotiations

Negotiation	Negotiation Title	Negotiation Type	Status	Monitor	Time Remaining	Close Date	Responses	Created By	Role
50474	Auction TCA	Auction	Paused		12 minutes	Not Applicable	0	Fwne,Clare	Owner
60531	RFI XT	Auction	Active		1 Day 7 Hours	30-Jun-2010 00:41	1	Fwne,Clare	Owner
60088	ANX Response	Auction	Active		1 Day 14 Hours	30-Jun-2010 08:23	0	Fwne,Clare	Owner
60316	ANX test item C	Auction	Active		1 Day 15 Hours	30-Jun-2010 09:09	0	Fwne,Clare	Owner
60318	Auction test itse	Auction	Active		1 Day 17 Hours	30-Jun-2010 10:46	0	Fwne,Clare	Owner

Drafts

Negotiation	Negotiation Title	Negotiation Type	Created By	Creation Date	Open Date	Locked By	Locked Date
1	Au TCA9	RFQ	Fwne,Clare	18-May-2010		Fwne,Clare	18-May-2010
10	Au 245	RFQ	Fwne,Clare	19-May-2010		Fwne,Clare	19-May-2010
11	RFQ TMER	RFQ	Fwne,Clare	19-May-2010		Fwne,Clare	19-May-2010
12	RFI GET EG8	RFI	Fwne,Clare	19-May-2010		Fwne,Clare	19-May-2010
13	RGIST RATE 4	Auction	Fwne,Clare	30-May-2010		Fwne,Clare	30-May-2010

Columns Hidden 1

Expiring Agreements

Agreement	Change Order	Description	Supplier	Agreement Amount	Released Amount	Currency	End Date	Status
1000034		BPA0103	SuppCCV	40,000.00	199.00	USD	30-Jun-2010	Open
1000035		BPA0101	SuppCCV 01	25,000.00	1,669.50	USD	30-Jun-2010	Open
1000037		BPA0104	SuppCCV 01	25,000.00		USD	30-Jun-2010	Open
1000038		BPA0102	SuppCCV 02	350,000.00	1,427.50	USD	30-Jun-2010	Open

FIGURE 2-1. *An overview dashboard page for supplier negotiations*

users' job role. Items in the Worklist can be selected and right-clicked to see a list of common actions to take. The second common component in welcome dashboards is the watchlist. This region contains one or more functional objects (transactions, people, accounts, etc) that have been flagged as being of particular interest and are therefore eligible for close monitoring. Any specific attribute can be included in the watchlist, such as statuses, values, and quantities.

Layout

The pages that make up the Fusion Applications user interface are constructed around a standardized set of *design patterns* that ensure consistent layout, content types, and styles throughout. At the highest level,

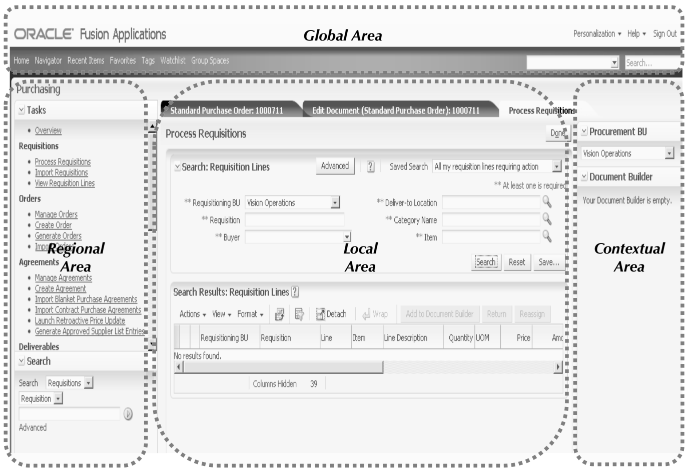

ORACLE Fusion Applications
Global Area
Personalization Help Sign Out
Home Navigator Recent Items Favorites Tags Watchlist Group Spaces
Purchasing
Tasks
Regional Area
Process Requisitions
Search: Requisition Lines
Local Area
Search Results: Requisition Lines
Procurement BU
Document Builder
Contextual Area

FIGURE 2-2. *The Fusion Applications page layout*

each page is composed of four main parts, as illustrated in Figure 2-2. The top *global area* holds all the menus and links that apply to every page, maintaining a parent context within which the product features run. This offers navigation, collaboration, and help functions. The *regional area* offers users some helpful features that support the overall business process in which they are currently operating. The central *local area* is the heart of the page's focus, where users can interact with the data related to the task being performed. The final *contextual area* is, as its name suggests, a place for additional information that supports the current task, specifically that of the local area.

Web 2.0 and Social Networking

Fusion Applications is designed to support the next generation of information workers (often known as *generation-Y*), whose expectations of application usability far surpasses that offered by traditional enterprise applications. The term *Web 2.0* is commonly used to encompass these expectations. Some examples of such features include pages that are composed of dynamically refreshing real-time data (such as Flash and Asynchronous JavaScript and XML [AJAX] data); that contain rich graphical components such as maps, charts, and fit-for-purpose diagrams; and that have embedded collaborative techniques such as wikis, network diagrams, and instant messaging. Fusion Applications is up to this challenge and provides a host of components that exist either as discrete services from the underlying technology platform (such as WebCenter) or as powerful widgets embedded directly into the user interface pages.

Fusion Application is not littered with these components just for the sake of looking good or following the latest trend. Each Web 2.0–style component has been selected for inclusion because it offers significant improvements and efficiencies over using an alternative, and it helps maintain a professional and easy to use approach to the overall design.

Embedded Analytics

The graphical components of the user interface substantially support the presentation of complex data in an easy-to-consume way, and Fusion Applications leverages a standard set of tools to embed them in the most appropriate places. This is accomplished primarily by another standard design pattern and a set of specific components that work seamlessly with the Oracle Business Intelligence Enterprise Edition (OBIEE) server to keep the information up to date and accurate. More detail on how this works is provided later in the section "Business Intelligence."

The Model-View-Controller Architecture

So how do these front-end principles link to the back-end processing? The whole of the Fusion Applications interface was written using Oracle JDeveloper 11g and its embedded Application Development Framework (ADF). Native to this is the Model-View-Controller (MVC) architecture, a standards-based development paradigm that is available for many different programming languages. It uses many of the same principles found in E-Business Suite's Oracle Application Framework (OAF).

In its most basic interpretation, the *Model* is a software layer that handles all the interaction with the database (or any data source) and controls the execution of the business logic code. The *View* is, as the name suggests, responsible for the display of the application user interface, and the *Controller* is essentially the wiring between the two that handles page flows and events as they occur. Abstracting the code objects into these layers has two main benefits: First, it facilitates a modular approach by using sets of self-contained objects, and, second, it greatly reduces internal dependencies, reducing maintenance and promoting code reuse.

Oracle ADF itself actually abstracts the Model layer further, specifically breaking out pieces around data persistence, object-relational mapping, transaction management, and business logic execution into a *Business Services layer*, implemented in Fusion Applications within ADF Business Components (ADFbc). This leaves the remaining Model layer in control of connecting these business services with view/controller pieces via its data binding and data control components.

Business Logic Execution Architecture

Fusion Applications actually represents a complete set of Java enterprise applications, deployed in the Oracle WebLogic Server (WLS) Java 2 Platform Enterprise Edition (J2EE) runtime environment. In addition, a small percentage of business logic runs outside of the Java runtime environment, namely some Procedural Language/Structured Query Language (PL/SQL) code for a few data-intensive activities, and some C code for a few well-defined, mature processes that need that lower level of control.

Further details on code standards for Fusion Applications can be found in the developer guides listed in the Appendix.

Orchestration

It will come as no surprise that Fusion Applications has been totally built around the Service-Oriented Architecture (SOA), having more than 1000 web services covering all its major functional objects and their associated tasks. This runs on the native Oracle SOA Suite technology, taking specific advantage of the Business Process Execution Language (BPEL) and workflow features therein. All of the artifacts and metadata that make up these features are open standards–based.

The Security Architecture

Security has always been imperative for Enterprise Applications, especially with internal security breaches forming a significant problem. All modern enterprise applications contain a wide range of features and integration points that could potentially expose any unsecured entry routes. Examples might be misconfigured independent servers and unsecured application programming interfaces (APIs) that can be invoked to produce undesirable results. Another concern is how interfaces into Enterprise Applications are evolving, with more devices being used for business tasks, such as mobile and handheld applications, all working in real time with both internal departments and external partner organizations.

Fusion Applications addresses these concerns with a range of security solutions; however, underpinning each of these is a security architecture known as *role-based access control* (RBAC). Fusion Applications is based on the *segregation of duties* principle, meaning that access is broken down into specific tasks that are closest to what people actually do in their daily jobs. These tasks are grouped into roles that are then applied using the RBAC approach.

The purpose of RBAC is often summarized in this way: *Who* can do *what* on *which* set of data. This is implemented in Fusion Applications in the following way: The *who* is represented by a *Job Role*, itself composed of one or more *Duty Roles*. This Job Role gets either explicitly or automatically *provisioned* (assigned) to new users. The *what* identifies a particular level of *Function Security*, meaning it *defines* the privileges contained inside Duty Roles that control the range of associated work tasks. Finally, the *which* represents *Data Security*, defining specific datasets upon which a user can apply actions (via function security). An example would be that for one user with a Job Role of FL-Buyer, purchase orders associated with the California Business Unit can only be viewed, whereas orders from the Miami Business Unit can be both viewed and edited.

Another security measure that is increasingly important is the control of personally identifiable information (PII). Fusion Applications defines three levels in which all data is categorized, with specific controls on PII data. The first category is *Public* and has no extra control. Examples might be simply company names and addresses that are used everywhere, including on external documents. The second category is *Internal-Public* and provides an extra layer of control for more sensitive data using Data Security. An example here might be a person's work contact details. The final category is *Confidential*, and this data is secured at all levels, including storage in the database, communication through the network, and of course in the user interface. Examples of confidential data are bank account numbers and user's passwords.

With security a key part of the applications administrator's role, extensive details on the technologies and their related management tools are presented in later chapters.

Data Storage and Retrieval

The database traditionally forms a stable foundation for data-centric enterprise applications, onto which everything else is built. You've seen how Fusion Applications leverages various advanced features for capture, manipulation, and presentation of data, but the basic architecture for storage and retrieval remains fairly constant. That said, two particularly innovative features come with Fusion Applications.

First, Fusion Applications has a unique deployment framework known as the *Pillar Replication Framework*, which, although not part of the initial release, will allow for separately distributed databases to contain different functional parts of the application. Internally, Fusion Applications splits the application features into groups of product families, each one known as a pillar. Consider, for example, three pillars. The first pillar may contain Financials and Supply Chain Management (including Project Portfolio Management [PPM] and Procurement), the second pillar is Human Capital Management, and the third is Customer Relationship Management. Each of these three pillars will run in its own, separate database instance. To keep all the dependent data (such as product items, employees, and enterprise structures) synchronized among the pillars, Fusion Applications will use a technology platform component, Oracle Data Integrator (ODI), an evolution of the acquired Sunopsis product. This will perform the Extract, Transform, Load (ETL) and Master Data Management (MDM) type functions. More details on ODI are provided in the next section.

In addition to the planned Pillar Replication Framework, Fusion Applications will natively leverage Real Application Clusters (RAC) for performance and scalability. RAC is Oracle's virtualization technology that provides Enterprise Grid Computing support for the Oracle Database. In a nutshell, this allows organizations to run each single database instance across several different physical servers. This, together with the Pillar Replication Framework (and similar flexible middleware deployment options), provides a wide range of capabilities that should satisfy almost any complex topology and capacity requirement.

The Technology Stack

This section represents a particularly significant section of this book, because it covers the complete spectrum of components in the Fusion Application Technology stack. Although specific reference material (listed in the Appendix)

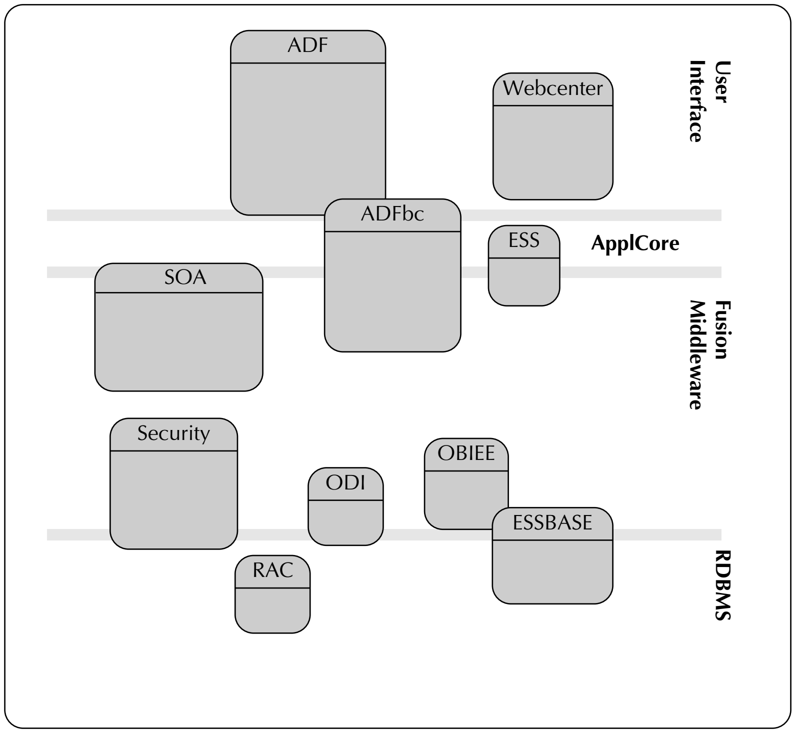


FIGURE 2-3. *Fusion Applications technology stack summary*

will cover each component in much more depth, Fusion Applications administrators first need to identify and appreciate the anatomy of the system. Then they can begin to understand how these components work together. Again, the focus of the discussion is on understanding the application technology to support the requirements of the business; we won't discuss technologies for their own sake.

Figure 2-3 provides an overview of some of the key components of the Fusion Applications technology stack, and we'll look at each piece in turn. Chapters 5 to 10 provide more detail on how an applications administrator would manage each discrete component, or at least those in a specific class.

The Front End

As with most effective and seemingly simple solutions, behind the rich and intuitive Fusion Applications user interface sits a considerable amount of complexity and technology.

Application Development Framework

As mentioned earlier, Oracle's Application Development Framework (ADF) provides most of what the business users see. The View layer, built as part of the MVC architecture, provides the user interface pages, forms, and components, with the Controller layer handling the user interaction such as button clicks (known as *events*). ADF is based on, and fully compliant with, the open standard specification for Java Server Faces (JSR 127) for the development of a rich user interface. Oracle's implementation, ADF-Faces, remains compliant with the standards but offers an extended range of more than 100 AJAX-like client components that provide powerful yet standardized ways in which business users can interact with the application. Examples include maps, graphs, and pivot-tables.

The Fusion Applications ADF layer also merges the View and Controller artifacts to ensure that pages look and work in an integrated and consistent manner. In addition to components and pages, ADF contains a *taskflow* mechanism that controls precisely how processes navigate from one object to another, such as page navigation and code calls. Taskflows are discrete components, being reused whenever a specific sequence of steps is required.

Although this section doesn't go into pages of developer-level detail, clearly it's important that the applications administrator understand the technology behind the user interface, as maintaining its availability and performance represents an obvious priority. To assist with this, later chapters offer more detail, together with some utilities and diagnostics for monitoring and troubleshooting. In addition, for those organizations creating and running customizations and extensions using their own ADF objects and code, full supporting details can be found in the developer guides listed in the Appendix.

Simple Validation

Sitting behind the user interface is a layer of embedded logic that performs basic validation on both the data entered and the events fired. This logic operates by two main techniques: *Expression Language* (EL) and *Groovy Scripting*. Expression Language provides an easy-to-use syntax for handling the data held within the application session, reducing the need for full Java code execution to handling common or simple actions. Groovy Scripting provides a Java-like language that is used a little closer to the business logic, commonly within the MVC Model layer. Similarly to EL, Groovy offers a lightweight way in which to execute simple logic such as attribute value calculation and explicit validation routines.

ApplCore

As mentioned, Enterprise Applications try to use common components everywhere possible. Although we touched on these earlier, some other common technical components also provide services to the application. A small subset of these are uniquely provided to satisfy enterprise application requirements, and are extensions to the standard technology products. This forms a thin component layer that exists near the top of the technology stack. In Fusion Applications, this layer is known as ApplCore (Applications Core technologies).

In Fusion Applications, ApplCore is a much thinner layer than the equivalent inside the other Oracle Applications products, mainly because it directly leverages the advanced capabilities from the latest versions of the technology products (Database, Fusion Middleware, and so on). Minimizing the footprint of this layer by using native technologies greatly reduces internal dependencies, processing complexity, and the overall maintenance needs. The following provides an overview of the main ApplCore services.

Attachments

For many transaction documents and business objects, the addition of unstructured organization-specific material can be very important, but an application form or field to hold it may not be always available. A simple example could be the technical specification of complex items on a purchase order—probably not something that's needed on every order and hard to imagine how suitable fields might be arranged on a page. The solution in most enterprise applications is for business users to upload supplemental documents and attach these to specific parts of a transaction document, such as an order line. Enterprise applications often limit the uploaded documents in terms of size, but not on file types or formats.

Oracle Fusion Applications provides this same attachment capability in all suitable places, leveraging Oracle's Universal Content Management (UCM) product range, and specifically the Document Management Repository features (formerly Stellent). This permits attachments of various types, including URL, text values, and any types of file. All attachments are put into an internal category foldering system, shown as a graphical file/folder hierarchy (similar to Windows Explorer), which together with an attribute keyword search and version control, makes them easy to find and reuse. All attached files can also be automatically virus-scanned and are secured by the standard application role-based security, controlling all basic actions, such as view, add, rename, and delete.

Flexfields

The concept of flexfields originated from E-Business Suite, where it proved a powerful and popular feature. Very simply put, flexfields provide a customizable, yet supported, way of defining implementation-specific data entry fields for use across the application features.

Traditionally, flexfields come in two types. Key flexfields are multi-part identifiers used to represent a specific value that's built up from one or more other values (known as *segments).* A simplified example of a Key flexfield might be the four segment value for an account code (such as 0129-210-1200-99), where the first segment (0129) is the account number, the next is the department, the next the division, and finally the cost center.

The second type of flexfield is the *Descriptive flexfield*. This provides additional database and user interface fields (known as *attributes*) that can be used however and whenever an organization requires them. This is a quick and easy way to include additional information associated with transactions (such as an order).

In addition, Fusion Applications introduces a third kind of flexfield, the *Extensible flexfield*. This is somewhat similar in purpose to the Descriptive flexfield, but it allows for the creation of entirely new objects and the definition of custom relationships between objects and attributes. This offers organizations many more options for use.

Flexfields are exceptionally important in implementation phases, and although they introduce some complexity, their maturity and well-defined structure should make them a simple piece to administer.

Profile Options

All software applications require setup and configuration to control precisely how they work. The functional features and processes inside enterprise applications are exceptionally flexible, something that is essential for use across different organizations and between different industries. The ApplCore layer offers a standard way to perform application setups and configurations, known as *profile options*. A profile option is a name-value pair that is set within a common user interface, stored in the database, and pulled into the user context at runtime as needed. A simple example (that you'll see again later on) is the profile option *FND: Log Enabled* that controls whether application diagnostic logging is turned on or off, and therefore has two possible values: Y or N.

Profile options are set within the context of a *level*, adding the extent to which the configuration applies. In our logging example, this profile option can be set at *site* level, meaning it applies to the whole application and all users thereof, or at *user* level, applying to just one specific chosen user. Again, you'll see examples of this later on.

Similar to flexfields, profile options have been part of E-Business Suite for many years, and although some minor evolution has occurred in Fusion Applications, much of the structure remains mature.

Audit

Fusion Applications implements the Fusion Middleware Audit Framework to provide full access to audit capabilities on the components operating in the technology stack, from security processes to specific product applications. ApplCore provides a standard audit policy as a baseline of audit constituents, which should be reviewed and extended upon implementation, using the extensive tools available. The framework also provides standard reporting capabilities, as well as integration to Business Intelligence (BI) components for setting up notification and alerting rules.

Internationalization

Fusion Applications is available in eight languages in version 1.0, with more to be added in additional releases. National Language Support (NLS) preferences are stored for each use and also include time zone, date and number formats, and calendar style. These preferences are managed by the technology stack, allowing Fusion Applications to load the appropriate runtime context quickly at login, and then adjust all the pages and components using its embedded metadata framework on top (described later).

In relation to internationalization, it's helpful to have a basic understanding of exactly how Fusion Applications manages the user interface text. Two techniques are at work inside the application. The first, *resource bundles*, handles all the ADF-Faces user interface component labels (known as *strings*) from the text on buttons and tabs, or simply from the entry-field labels. These are deployed as files (.xliff), imported at runtime, and linked to ADF component attributes and their underlying code objects.

The second technique handles all the *messages* that are shown to the end users when they take an action on the system. Examples might be a confirmation message that a change was saved and a warning that a process cannot be completed. The text for these messages is simply queried directly from the database based on the current session language. These messages also commonly contain tokens for inserting dynamic values such as the current transaction number. As you'll see, more serious error messages are even used to trigger system management functions automatically, such as extra logging and the creation of an *implicit incident*.

Taxonomy

Underpinning Fusion Applications is a structure that ties all application technical components together into a logical hierarchy. With such a huge system of thousands of individual artifacts (e.g., code and definition files) this structure, known as the Fusion Applications Taxonomy, is essential to promote accurate object reuse and modularity, and to provide detailed dependency information.

The taxonomy is a simple hierarchy with four basic levels, starting with Product Line (Fusion Applications), then Product Family (e.g., Financials), then Product (e.g., General Ledger), and finally a value known as the Logical Business Area (LBA), which represents a leaf grouping of components based on their use, for example, GL Calendars. The LBAs can be nested also, creating further levels where required. All components are then assigned an appropriated taxonomy value.

The taxonomy is used in several places in Fusion Applications, albeit under the skin of the visible activities. Provisioning and Patching are the main consumers of taxonomy information; however, there are other features that use the structure to interrogate and display information related to Fusion Applications, such as the Enterprise Manager Topology diagram displayed in Figure 7-1 later.

Fusion Middleware Extensions for Fusion Applications

The next level down in the technology stack is Oracle Fusion Middleware (FMw); although another small subset of components exist in the standard Fusion Middleware products, they have been especially enhanced for use with Fusion Applications. These components are similar to the ApplCore layer, but strictly speaking, they're technology components implemented for applications, as opposed to pure application pieces. The two most significant parts of this layer are the Enterprise Scheduling Service and Approvals Management.

Again, most other Enterprise Applications contain many more similar such components, specifically developed to meet application-specific requirements. Although these might have been fit-for-purpose at one point in time, parts can quickly become outdated if they are not constantly refreshed, representing a significant dependency and maintenance overhead. Upon closer inspection, you'll find that it's clear that many of the services provided are not really unique to any one application, and the latest releases of Oracle's Fusion Middleware provides most of these services to support all enterprise application development and execution.

Enterprise Scheduling Service (ESS)

This feature provides a method of executing units of work at specific times and in specific sequences, similar to the batch processing of older enterprise resource planning (ERP) applications. This requirement may seem rather old-fashioned in today's always-on, super-powerful systems. Most applications administrators know that enterprise applications often contain terabytes of live data, and performing large updates can seriously impact resources and performance; those non-urgent requests are better scheduled for quieter times.

ESS represents a mechanism whereby its internal *scheduler* accepts specific units of work, known as *jobs*, to be started at certain dates and times within specific resource constraints (time, Java threads, and so on). The jobs themselves are registered by the *client* business application (such as General Ledger, Payroll) and are not actually run by ESS itself, but when the specific time is reached, ESS performs a callback to the client that kicks off the work.

ESS has features such as its own log records; detail on-job request dependencies (shown as a tree hierarchy); and the simple ability to review (and purge) past, current, and planned load. All ESS jobs are secured via the same RBAC policies and roles that exist within the client application itself.

Approval Management Service

The Approval Management Service (AMX) abstracts the transaction document approval processing from any specific implementation and then provides a range of management services to use as required. The technology to do this is based around the FMw Human Workflow product. This includes prebuilt integration to the notification capabilities and the especially designed Fusion Application Worklist user interface component.

AMX supports all manner of approval processes including the following:

- Both static and dynamic approver list generation
- Serial and parallel approval paths
- Graphical presentations of approval hierarchies
- A wide range of business rules, identity, and decision services
- Context sensitivity to derive runtime attribute values
- User interfaces for easy configuration and predeployment testing

The creation of approval rules tends to occur mostly during implementation; however, applications administrators should again be aware of the component parts and the processing mechanisms involved.

Oracle Enterprise Crawl and Search Framework

Fusion Applications leverages Oracle's Secure Enterprise Search (SES) technology to provide keyword and category-based search results across a number of data sources (database and non-database), all while making sure the right information is shown to the right person. The internal crawler process first gathers information based on its own security implementations, and the resulting index is stored in an especially hardened database. Finally, all search results are carefully secured using a range of options including access control lists, query time authorization, and combinations of the two.

On top of this base functionality, the Enterprise Crawl and Search Framework was created especially for Fusion Applications. It provides a set of standard APIs into the SES platform, it integrates with the applications-based RBAC security model, and it has a range of standard user interface display options. This helps ensure a complete and consistent search experience across all of Fusion Applications.

As with managing any data repository, there are some significant considerations for applications administrators here, especially regarding managing data sources and their security, scheduling internal processes, and optimizing performance. Further details and activities are explained in Chapter 5.

Core Fusion Middleware

Oracle has continued to evolve its Application Server, Middleware, and Development Tools, through acquisition and new product creation. They are currently represented by a cohesive and integrated platform for developing and running applications, namely Fusion Middleware. This layer is application agnostic, meaning that many of its open standards–based components can be used with any compliant enterprise application. Although the references in the Appendix provide more detailed content on Oracle Fusion Middleware itself, following are a few key pieces to give you a quick summary of their use in Fusion Applications.

WebLogic Server

Although it leverages specific components for specific tasks, Fusion Applications is primarily a set of Java EE Applications, deployed within the WebLogic application server Java runtime environment. Since such a large number of these deployments exist across Fusion Applications, the scalability features inside WebLogic provides a solid platform while retaining flexible implementation options. For those unfamiliar with WebLogic Server, the following simplistic summary of the Fusion Applications architecture should prove helpful.

WebLogic includes three main groupings:

- **The managed server** This is essentially the Java Runtime Environment (JRE) within which application Java code executes in one or more Java virtual machines (JVM).
- **The cluster** This is a group of one or more managed servers that may be running across one or more machines. Capacity adjustments can be made quickly and easily by adding managed servers or physical machines to a cluster. Clusters are also used to ensure availability, as one managed server can take over should another fail.

- **The domain** This is a container in which all the managed servers and clusters are grouped. The domain allows servers with similar requirements to be managed together. This management is accomplished by a dedicated managed server instance that must exist in each domain, known as the *administration server*. Additional Fusion Middleware components are also commonly configured at the domain level.

As explained at the start of this chapter, Fusion Applications is divided into product families based on the particular business features they implement. Each product family has all its Java EE applications deployed to one WebLogic domain. This domain contains several clusters within which applications are deployed (roughly one per managed server), plus the administration server that handles all internal management functions (including running Fusion Applications Control). The detailed configuration needs and dependencies of each product family domain are handled as part of the Fusion Applications installation and provisioning process.

In addition to having clusters for each Fusion Java EE application, two more are required within each product family domain. The first is used to host the SOA Server infrastructure (SOA-INFRA) and its related composite applications. The second additional cluster is for an instance of the ESS, itself running as a Java EE application (known as *ESSAPP*).

The only additional WLS logical container to mention is the *farm*. A farm is the top-most grouping, within which domains, clusters, and all administration and managed servers are held. It is specifically implemented for use by Fusion Applications Control to allow application administrators to manage the system as a whole.

Clearly, the management of the WebLogic environment represents a sizable portion of the applications administrator's role, and later chapters provide plenty of background information and descriptions of the standard tools available. The discussion also adds some insight into how to use some helpful extra features and introduces some recommended management practices.

ADFbc

As mentioned, working away behind Oracle's ADF is ADF Business Components (ADFbc). It continues the support for the MVC architecture by providing a range of back-end services. Specifically, it provides the data binding capability for use in the Model layer, and it supports the creation of the components for the Business Services layer.

ADFbc's data binding occurs in complete accordance with the open-standard JSR-227 recommendation. This states how declarative data access should be defined between user interface components and the underlying functions of the Business Services layer, running in a J2EE environment. In ADFbc, this occurs via the data control and its list of data-aware user interface components.

As its second core capability, ADFbc allows for the creation and execution of business services, and it does this by implementing a specific set of code objects. Fusion Applications fully adheres to this, and the two most fundamental objects in this layer are the *entity object* (EO) and the *view object* (VO). Simply put, an entity object provides an abstraction of the physical data model in the database, built to represent a logical business object and its properties (such as persistence). For example, each line on a Requisition can be represented by a *RequisitionLineEO* object. The code logic performs actions on the EO object itself, and the changes are automatically propagated to the underlying database by the ADFbc platform. Relationships between EOs are defined by another object, known as an *association*.

View objects are simply containers for queries that pull data from the database (SQL), with any relationships between them defined as *view links*. An example from Fusion Applications is *RequisitionLineVO* that simply queries item lines for a given requisition ID. View objects are also rolled-up into parent containers known as *application modules* (AM), and these load related VOs at runtime, plus contain additional properties such as transaction control definitions.

As mentioned, this is all abstracted and deployed as open-standard XML metadata files because it removes any hard coding of implementation detail, meaning that ADF-based applications can be implemented in a range of different J2EE environments. For Fusion Applications, this standard-based, best practice development architecture represents the most efficient and effective way to develop and deploy enterprise applications.

So how do we go one step deeper, into the real application processing logic and the technology stack involved there? In simple terms, once the entity object defined in ADFbc is updated it will raise a new *business event* through the use of something known as the *Event Delivery Network* (EDN). Other code objects will *subscribe* to that event and be listening for it. Once triggered, the event will pass along any associated data and begin the next phase of processing. A simple example might be when an order is updated the EO raises an associated event, which in turn fires a human workflow process to notify the buyer to approve the change. Later sections continue drilling into this execution process thread, but we need to pause at this point to look at another important technology stack component.

Business Process Execution Language (BPEL)

In simple terms, although SOA-based web services (and other components) form a modular architecture, they still need gluing together to deliver an execution flow that closely matches to what the business user is trying to achieve (the BPM flows). Rather than the traditional method of hard-coding this wiring in the code, BPEL steps in. Using standards-based XML formats (such as BPELWS), it allows for the declarative definition of business process execution flows. This is partnered with a language upon which the middleware platform can execute to chain together discrete services and programs, forming what is known as *composite* applications.

In Fusion Applications, all processes are built using JDeveloper's BPEL Designer, and the BPEL Process Manager forms the execution engine that runs within the WebLogic SOA-INFRA cluster. As mentioned, BPEL processes may be kicked off by an event, such as a button click on the user interface, and the subsequent processing runs in its own context and according to the predefined logic.

Clearly, it is important for the Fusion Applications administrator to be able to monitor, manage, and troubleshoot all the executing process flows in the system, and later you will see how to use tools like the BPEL Console to review the process execution.

It's easy to make comparisons between BPEL and the Oracle Workflow component that's used in the Applications Unlimited products. This is a somewhat fair comparison, although BPEL is a more complete architecture for embedding business processes, plus it has much broader support for different service providers. Oracle's BPEL Process Manager runs a separate subcomponent called *Human Workflow* that, similarly to Oracle Workflow, provides dedicated features to handle interaction with business users within a process flow. Human Workflow provides many features to support the assignment of *tasks* to specific individuals or groups of *participants*, including the management of priority and deadlines for completion, the sending of notifications and reminders via various communication channels (not just e-mail), as well as standard methods for processing responses and passing the results back into the BPEL process flow. Figure 2-4 shows a simplified BPEL process that involves Human Workflow (the NotificationTask node) to notify a user of a pending task and to process the response.

As you may have worked out, the AMX and the Worklist component are both part of BPEL Human Workflow. They leverage several of its advanced task administration features, such as parallel approvals, escalation, and delegation.

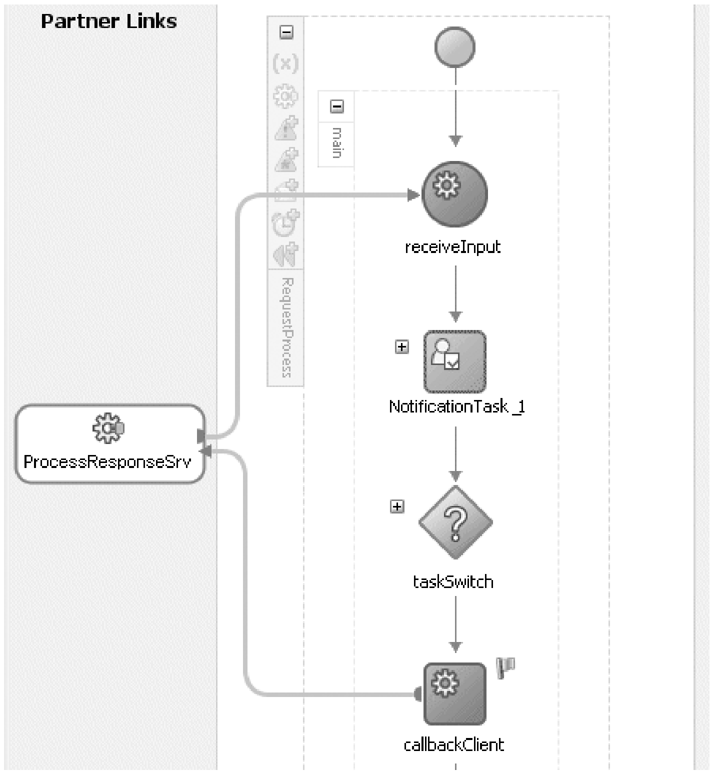


FIGURE 2-4. *A simple BPEL process with Human Workflow*

One other BPEL component used in Fusion Application is the *Activity Guide*. This is similar in concept to regular BPEL process flows, except instead of chaining together programmatic services or human workflows, it is primarily focused on high-level functional features and user interface components only. A simple example is the linking together of the steps involved in on-boarding a new employee into a company. The Activity Guide for this would include setting up ADF taskflow elements such as the payroll and benefits information, and performing legal and contract administration. Activities Guides can also be used to support disconnected tasks that are external to the Enterprise Application, such as creating an e-mail account or even issuing a security badge.

Fusion Applications includes full support for activity guides and presents them intuitively within its dashboard pages, so all related actions can be carefully monitored and tracked. Using the employee on-boarding example, the dashboard includes a simple progress bar with an associated tasklist, showing each item's status with drill-down links for more detail.

One final BPEL component of which applications administrators should be aware (but by no means is it everything) is *Oracle Business Rules* (OBR). This forms a method in which to apply complex policies, regulations, or computations, based on the well-known Rete algorithm. Like the Human Workflow components, this is represented as a specific node type within a BPEL process and governs how the execution process will flow. Each Business Rule node has its inputs and outputs linked to a reusable rule definition—a *ruleset*—created declaratively in the Rules Designer and stored in a rules library. OBR provides many advanced features for the implementation of organization-specific logic, although its use does require a certain level of expertise. The Application Administrator should be aware of this moving piece, understand what it does, and know how to manage it.

One additional comment regarding Fusion Applications specifically is that although the BPM levels (particular Level 4 tasks) may represent logical containers for BPEL processes, there is no consistent physical link between BPM and BPEL in version 1.0. This may expand in future releases, where the whole business process, including the execution logic, is held and managed in one all-encompassing business process repository.

Oracle Mediator

As discussed, BPEL primarily represents a way in which to organize the discrete components of a composite application into one or more flows that mirror the business process activities the organization needs to complete (such as procure-to-pay). Although BPEL Process Manager forms the engine for process execution, it does not handle the invocation, management, and monitoring of the services it calls. This is done by Oracle Mediator. It works inside the SOA infrastructure, performing several of these orchestration-type functions. At the most basic level, Mediator forms the XML message handling mechanism through which almost all activities occur.

Following are a few basic examples:

- **Service invocation** The messages passed between the components within a web service execution are all handled by the Mediator, supporting both synchronous and asynchronous invocation.
- **Business events and the Event Delivery Network** Where events are raised upon specific action, the associated messages are handled by the Mediator.

- **Message transformation** Since many components have different interfaces, Mediator performs the transformations required to get them working together.
- **Routing rules** Mediator can also validate and perform message routing based on the content of the messages, as part of specific rule definitions.

For applications administrators, Mediator also handles any internal exceptions (such as network interruptions), as well as those returned by the web services it calls. It provides a detailed error handling and reporting process together with its own logging mechanism.

WebCenter

Moving aside from business logic processing and back into user interface technologies, Oracle WebCenter sits in the Fusion Middleware stack and provides several services that integrate with and complement those already discussed.

Oracle WebCenter complies with the JSF standards of the ADF component, further enriching the end user experience and enhancing productivity. It focuses specifically on bringing Web 2.0–type features into Enterprise Applications, producing what some call *Enterprise2.0*. Fusion Applications, to leverage many parts of WebCenter. Following are some examples:

- WebCenter spaces and portals for management dashboards
- WebCenter Composer to perform runtime user personalization and more extensive customizations, based on a declarative and metadata driven architecture
- WebCenter Services provides various social media services, including the following:
  - Wiki and blog for the publication of user content
  - People Connection for graphically showing relationships
  - Tags for adding keywords to objects and data to help with searching
  - Instant Messaging and Presence (IMP) to promote collaborative working

With so many flexible and powerful features readily available via WebCenter, applications administrators may need to consider specific implementations carefully and monitor usage for such things as resource capacity control and even potential misuse.

Business Intelligence

As mentioned in the introduction, one of the key drivers of Fusion Applications is the move beyond processing transactions toward providing organizations with the tools needed for informed decision-making, whether it be optimizing order fulfillment schedules or making accurate sales forecasts. The implementation of these analytical type features has traditionally been a separate, dislocated effort, with the resulting tools found in another system, often away from the live transactional data. Fusion Applications integrates its intelligence capabilities into the work area interface, allowing patterns to be spotted and actions to be executed all in one place, greatly improving accuracy and efficiency.

Fusion Applications leverages several key technologies to manipulate raw data into a solution that is responsive, secure and accurate, while retaining comprehensive coverage.

- **Oracle Business Intelligence Enterprise Edition (OBIEE)** The OBIEE server provides extended analytical capabilities and access to various data sources beyond only the transactional data, most significantly Hyperion Essbase, an Online Analytical Processing (OLAP) server that provides scalable and responsive multidimensional data warehousing. Together, these form the foundation of the BI technology used across the whole of Fusion Applications, from the components embedded in ADF pages to more detailed analysis workbenches, scorecards, and reports.
- **BI Publisher** Previously called XML Publisher, this component takes customizable report templates and uses the XML transformation of real-time data to produce extremely high-quality reports. It's easy to use and supports even Microsoft Word–based declarative templates, and can be used to create multiple page reports that can contain a range of embedded analytical functions.

- **Oracle Business Intelligence Applications (OBIA)** As with the existing Applications products, Fusion Applications leverages a set of prebuilt intelligence reports, dashboards, scorecards, and other solutions for use within specific application products, such as Human Resources, Financial Accounting, and Supply Chain. These components have evolved from detailed analysis of best practice for BI, making sure that the right people in the organizations have access to the right information upon which to base effective decision-making.
- **Oracle Transactional Business Intelligence (OTBI)** Specifically developed for Fusion Applications this provides real-time reporting on live business data while leveraging the advanced display capabilities of the OBIEE server. This is most often used in the embedded analytics mentioned earlier.
- **Enterprise Performance Management (EPM)** Specifically designed for upper management, Oracle provides advanced solutions that leverage the entire set of application data and, based on an organization's own guidelines, thresholds, and targets, perform complex computations to provide business performance information in a notify-by-exception, actionable, and easy-to-monitor format. Traditionally, this has been through things such as KPIs and scorecards; however, Fusion Applications also includes a range of innovative methods and in-situ user interface components. EPM includes strategic and forecasting analysis tools based on operations such as supply chain optimization, financial budgeting and planning, asset and workforce management, and sales and marketing effectiveness. Several technical components are at work here, including Crystal Ball, the Hyperion EPM solutions for departmental and industry verticals, as well the Essbase modeling capabilities. The applications administrator needs to understand the moving pieces and ensure supporting management plans are appropriately created, undertaken, and maintained.

Oracle Data Integrator

As mentioned, ODI's role in the Fusion Applications technology stack is primarily based around data synchronization. ODI extends the standard ETL process of data migration into a process that first extracts and then, as one more flexible step, performs both the load and transform activities together (E-LT). ODI is also based on the SOA runtime architecture, meaning that service interactions are accomplished in the same way as the other Fusion Middleware components; for example, ODI can be called from a BPEL process.

ODI capabilities are leveraged in several places in Fusion Applications, and the following list touches on a few:

- To populate data in the BI data warehouse
- To migrate data from legacy systems being replaced by Fusion Applications
- To synchronize common data across the Pillar Replication Framework
- As part of integration to an external system (such as within a coexistence strategy)
- In MDM to synchronize the single source of truth

All these are touch-points for application administrators, and therefore we will go into more detail on the basic ODI processes and ODI's management in Fusion Applications, and mention some useful extended capabilities to meet organization-specific needs. Further specialist reading is provided in the Appendix.

Identity Management and Security

Security is at the heart of Fusion Applications, leveraging Oracle's already world-class solutions across each and every part of the application, from user login, to logic processing, and down into data storage.

User Authentication

Login authentication controls access to function and data security, as explained earlier in the chapter. Fusion Applications uses the Fusion Middleware authentication processes, via Java Authentication and Authorization Service (JAAS), to perform all its user authentication.

This includes several different technical components. It starts with basic user credentials being stored in one or more Lightweight Directory Access Protocol (LDAP) servers, such as Oracle Internet Directory (OID). For centralizing security, these LDAP servers may be distributed across different physical servers using something like Oracle Virtual Directory (OVD). The user credentials are then mapped to predefined security *roles*, managed by Oracle Identity Management (OIM). Where any changes are needed, the Authorization Policy Manager (APM) tool allows full role customization. In addition, where Single Sign-on (SSO) is required, Fusion Applications readily integrates with Oracle Access Manager (OAM).

At the application front-end, for most users the provisioning process is done by roles being granted to employees based on their work assignments, although in some cases (such as external supplier users), manual creation via OIM may still be required. Fusion Applications also supports importing users and roles from legacy systems via OIM and the HCM applications' Import Workers feature. Fusion Applications initially installs a few base users, although it does include a complete set of policies and roles to support each and every business process. Again, customization can be done using the graphical tools available inside OIM and the more general tool, Oracle Platform Security Services (OPSS).

As mentioned earlier in this chapter, Fusion Applications includes extensive features to ensure proper adherence to regulatory standards and controls, especially as part of its GRC product family. Obviously, this has interfaces into security and indeed contains some specific tools for that purpose. One example is the Application Access Controls Governor (AACG) that helps with segregation of duties inside GRC.

Middleware Security

With so many components in the Fusion Middleware technology stack, a full discussion of security would warrant a book of its own (see the Appendix for a bit more information). Suffice to say, Fusion Applications uses all the best pieces. The following list provides a brief summary of some of the main security technologies it employs:

- All web services are secured using a standards-based WS-Security–compliant architecture. This is fully accessible through Oracle Web Services Manager (OWSM).
- In addition to the services it calls, BPEL processes have several other moving parts and therefore include a number of security components, from Secure Socket Layer (SSL) for communication, to the application of user security context for function and data security in process execution. The BPEL Process Manager is the primary tool for monitoring and administering business process security.

- ESS secures jobs by revalidating the security access levels from the requesting users' security context. It validates both their function and data security privileges (roles, policies, and grants) against the requirements of the objects being accessed in the job (such as Java, PL/SQL, and C programs).
- The User Interface and Business Logic components, such as ADF, ADFbc, and the BI components, are all secured by the user authentication (JAAS) mechanism. This common security implementation helps all these components perform their integrated tasks seamlessly and efficiently.

As mentioned, security is an exceptionally important part of the applications administrator's role, and with the middleware forming such a large part of Fusion Applications, plenty of further discussion is warranted.

Database

The security technologies at work in Oracle Database 11*g* are well publicized, and Fusion Applications makes use of the best of these in a fairly standard way. So what is specifically notable for Fusion Applications? As mentioned, PII represents data that needs securing to protect individuals from being identified, contacted, or located by anyone unauthorized to do so. Fusion Applications contains about 50 fields that are categorized as *Internal-Public* and about 30 categorized as *Confidential.* To enforce strong security, it leverages the Oracle Database Vault capabilities when storing and communicating PII data.

A Processing Walkthrough

This section provides a simplified illustration of how a user action will trigger the parts of the technology stack, demonstrating how the parts work together at runtime. As a suite of composite applications, Fusion Applications does not actually have the linear execution paths that are more characteristic in traditional enterprise applications (user interface to database and back again). The different components running inside Fusion Applications often work at the same time, frequently as part of the same user interface page and triggered from the same event. Figure 2-5 shows an extension of the technology stack overview provided, adjusted to show a high-level view of the relationships between some of the parts discussed so far.

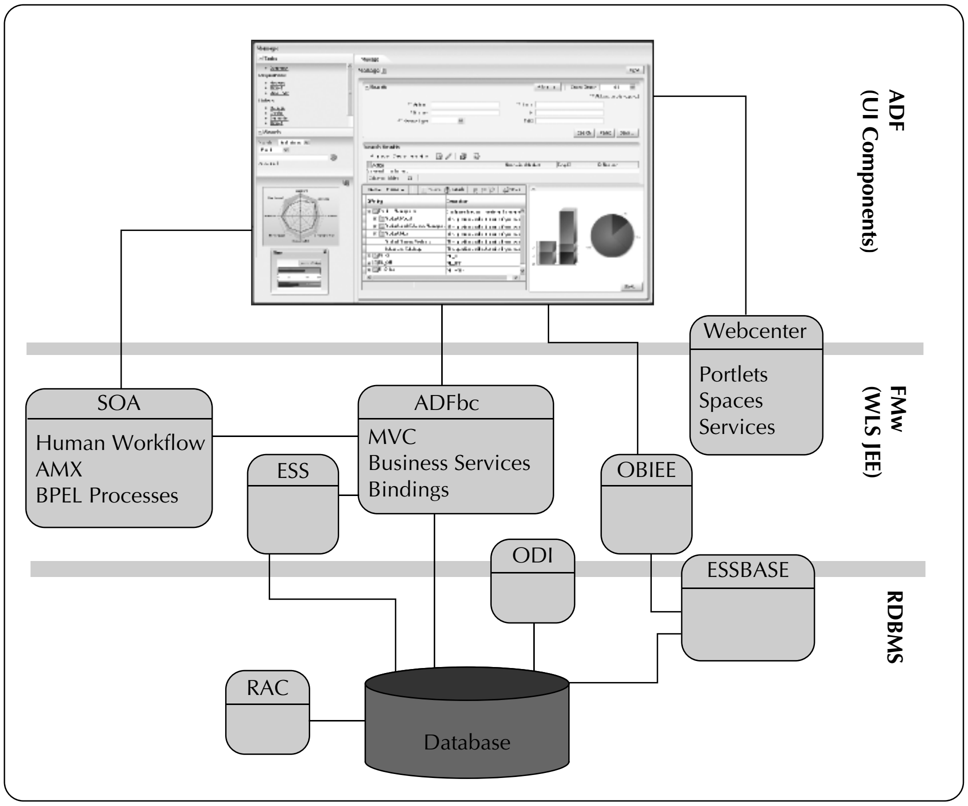


FIGURE 2-5. *How the technology stack works together*

To provide at least a rough outline of a common execution path, the following is a broad sequence of activities that could occur through the technology stack. It is broadly based on submitting a page of data by the business user, such as creating a new purchase order.

1. The user visits the application URL and basic session information is established (cookies, and so on).
2. The user logs in, invoking the full suite of identity management capabilities from OID and into OIM for the association of function and data roles (and their policies) to their session. The associated dashboard, with only authorized regions, data, and capabilities, is provided to the user.

3. The user completes the form and clicks the submit button. The request invokes any embedded ADF validations (Groovy, and so on) and fires the controller code. The controller methods commit some data (via entity objects) at this point.
4. The controller and the page then raise an associated business event within the ADFbc layer and any associated subscriptions are triggered (via EDN). Where the next step represents a large piece of work, the processing may split so that the user session is released to continue forward and the back-end processing runs behind the scenes.
5. Event subscriptions invoke the Mediator to handle subsequent messaging to and from an associated BPEL process flow.
6. The BPEL Process Manager starts the flow and executes the associated steps based on the data it has received. As it runs, it retains context data and runs associated web service calls in and out of the BPEL process, also orchestrated by the BPEL Process Manager and the Mediator. Many different actions and processes may be spawned, including Human Workflows, ESS jobs, and child BPEL processes.

As various threads of processing run, the associated data is updated through the system, so as soon as an activity occurs, the new data will be reflected in the application pages.

Coexistence Overview

Not all organizations will want to replace their entire enterprise applications in one go. It's a hugely expensive task, and unless benefits and return on investment (ROI) can be proven, it's likely to be out of the scope of near-term strategy plans. That said, many organizations will want to adopt the advanced features and more productive working environment offered by Fusion Applications, especially in business areas where significant improvements could be made. This is possible right away with version 1.0, using prepackaged coexistence integrations. These represent several functional features that have been specially designed to run in the Fusion Applications instance while readily integrating to complimentary existing Applications products. By leveraging the SOA, as well as technologies such as ODI, Fusion Applications is natively structured for this kind of process integration.

The first release includes planning for ten of these coexistence integration scenarios, and they focus on some of the best Fusion Applications features, without the need for a full-scale implementation or migration. It's seen as a relatively simple first step to adopt one or more of these integrations, which can bring immediate benefits to business users. Coexistence also represents the start of a flexible piece-by-piece implementation program that can be progressed at times most suitable for each organization.

The following list provides the coexistence scenarios that are planned for Fusion Applications in version 1.0:

- Supply chain management
    - Fusion distributed order orchestration, to non-Fusion order entry/fulfillment
- Procurement
    - Fusion shared spending, sourcing, and contracts, to non-Fusion requisitioning
    - Fusion Procure-to-Pay, to non-Fusion general ledger
- Project portfolio management
    - Fusion PPM, to non-Fusion PPM
- Human capital management
    - Fusion Talent Management On Demand, to non-Fusion HR
    - Fusion Workforce Directory Management, to non-Fusion HR
    - Fusion Incentive Compensation, to non-Fusion CRM
    - Fusion Workforce Lifecycle Management, to non-Fusion Financials and CRM
- Customer relationship management
    - Fusion Territory Management, to non-Fusion customer relationship management
- Financial Management
    - Fusion Accounting Hub, to non-Fusion financials

Clearly, coexistence is a great starting point for implementing Fusion Applications; however, this inevitably represents additional work for the applications administrator. Although managing both legacy and Fusion Applications systems in addition to a major integration piece is not for the faint of heart, in the real world, this is not so unusual. For example, during any upgrade or migration, there is normally a project phase when two systems run side-by-side for a period of time before the full switch-over is done and the old system is decommissioned.

The Extension Architecture

Fusion Applications, and indeed all modern enterprise applications, come with a set of features and techniques to change and extended the way in which the standard software works. This is required, because although all the application features are reasonably flexible, they will never be able to satisfy the needs of every organization. Although many business analysts and consultants would push organizations toward standardization and less customization, sometimes an organization's unique attributes control profit margins and set them apart from others in the marketplace.

Traditionally, changes to the standard application increase the total cost of ownership, since applications are costly and complex to build, brittle to change, and complex to support. Ownership costs have been estimated as being around 70 percent higher for customized applications. Fusion Applications offers several solutions that allow organizations to make the most of their unique differences while mitigating most of the normally associated risks.

Configuration

Fusion Applications comes with a wide range of options for setting up both the technology stack and the application features to work in a way that best fits a specific implementation. These configurations are fully compliant with the standard product and are therefore fully supported by the system operation. Some examples of this might include the configuration parameters provided as part of the installation process (inside something called *Flow Designer*), the business setup data used as part of Functional Setup Manager, or simply the profile options that are set within the system.

Application configuration setup usually occurs during implementation, although occasionally adjustments may be made for such reasons as organizational changes or as part of optimization efforts. The applications administrator will play a large role in configuration management.

Personalization

Personalization represents the first level of system penetration in terms of making changes to satisfy nonstandard requirements. Truly effective enterprise applications should be sensitive to the context of each and every user, providing exactly what each needs to do the required job. Fusion Applications focuses on this from security roles and privileges, to the rich user interface design. Taking this one step further, through personalization it's possible for each user to configure the system him or herself so it looks, feels, and works to match users' individual expectations and preferences. This ranges from the size, shape, and content of the components displayed on each page, to configuring specific sets of query results that users commonly find useful.

Traditionally, many of these preference changes were lost upon user logout, but in Fusion Applications, everything is retained indefinitely. That said, no actual changes are made to the original page definitions; instead, each set of changes is stored separately and is quickly applied each time the user accesses the page. This is done using the Oracle WebCenter component known as Composer. For more extensive personalizations, beyond the scope of the Composer features, Fusion Applications leverages the Metadata Repository (MDS). Similarly to WebCenter, MDS stores all personalizations and reapplies only the changes using a layering approach. It supports multiple layers of personalization, applying one after the other, on top of the original base definition. This has many benefits, but most significantly it means the original base definition can be changed by a patch or upgrade without affecting the personalizations, as they're simply reapplied over the new version. Should a patch or upgrade significantly change the base definition, any personalizations that are no longer applicable (removed regions) are simply ignored. Personalizations are secured, and applications administrators can enable or disable the range of options available to end users.

Extensibility

When the changes needed are actually in the processing logic itself, WebCenter and MDS personalizations are simply not up to the job. Extensibility takes over and comes with two main methods for making these more invasive system changes.

The first one is using the flexfield architecture mentioned earlier in this chapter. Fusion Applications supports user defined unique identifiers (key flexfields) and additional data fields, used either in a single usage context (dynamic flexfields) or in multiple different contexts (extensible flexfields). When appropriate, this is certainly the recommended path to take, since no changes to existing code or data objects are required to get started.

The second method is much less declarative and requires the use of Oracle JDeveloper to begin manipulating code objects to make the changes required. As a general rule, most standard objects, such as EOs, VOs, Java classes, and BPEL processes, can be *extended* by new objects. These new objects should implement all the same interfaces plus add any of their own. Objects that cause restrictions in interfaces or changes to processing logic, security, or adjusted database objects are not part of extensibility, and these come under the customization category, discussed next.

The results of extensibility projects, when done correctly, are again applied through MDS repository layers, ensuring that future-proofing and simplicity are retained throughout.

Where the applications administrator is required to manage extended objects, he or she needs a good understanding of the technologies, tools, and techniques involved. Although the administrator might not be doing the actual development work, a detailed understanding can prove helpful when he or she investigates application problems and analyzes the potential impact of system changes.

Customization

Although most of the preceding three methods for adjusting Fusion Applications should suffice for the majority of organizations, inevitably some organizations' requirements can be satisfied only by serious changes to the base software itself. Usually this is a mixture of methods. Organization-wide personalizations are used to adjust the user interface, some extensions to standard objects (such as BPEL flows) get the system processes to match the business operations, and a few customizations are used to change the behavior beyond that of the standard feature configurations.

With Fusion Applications natively running on Fusion Middleware, leveraging a range of standards-based architectures, and integrating directly with the 11*g* JDeveloper integrated development environment (IDE) (via its own Fusion Applications customization role), any customization work required is now much simpler, more logical, and more manageable than it has ever been before.

The Fusion Applications Ecosystem

You are probably beginning to appreciate how each functional application (General Ledger, Purchasing, and so on) is part of a broad landscape of supporting components, each with its own specific purpose. So far, we have discussed the pieces that run underneath the applications, providing the execution platform for their various capabilities. In addition to this is a second set of components that sit alongside and complement the applications, forming a kind of *ecosystem* to support users from both the business and IT operations.

User Assistance

All Enterprise Applications are complex, and as such they include things like online help and documentation to assist new or inexperienced users. Fusion Applications, although designed to be easy to use, also has extensive information to help the successful completion of every task. Several new approaches within this content make it far removed from the traditional experience of wading through reams of static documents to find what you need to know.

Content Structure

A strong focus is placed on providing users with just enough information to complete their tasks—not too much and not too little. Each concise unit of knowledge is discrete and independent, providing a modular structure that promotes the reuse of information in different contexts. As these all work together, they roll-up to form a comprehensive overall coverage. Each unit can take one of seven forms:

- Frequently Asked Questions
- Key Concepts

- Examples of Use
- Task Sequences and Relationships
- Demonstration
- Further Reference Information
- Glossary

The knowledge is available in a range of consumption formats including simple text, diagrams, screenshots, and interactive video segments.

All help content is structured based on the BPM model, so it's easy to move from a top-level business process to each individual activity and find information for all the tasks therein. Obviously, this navigational layout for the content very closely corresponds both to how the application works (such as task flows) and how organizations work, making the content easy to find and always relevant.

Content Style

The user assistance (yes, it's seen as more than just online help now) in Fusion Applications has a few extra capabilities that take it beyond what is possible with traditional static content. First, as part of the advanced user experience (UX), several Web 2.0 (or Enterprise 2.0) features are available, based on integration with the WebCenter component. One of these is the ability to custom tag each item of content, so that keyword searches are more intuitive and the results are more accurate. Another neat feature is that each piece of content can be voted on by users, thereby placing popular and generally useful articles first in result lists.

Another exceptionally important capability is that all of the content can be edited or added to by anyone authorized to do so. The content is just as extensible as that in business process logic or the user interface components. No extra tools or particular skills are required to adjust help content, since the system itself provides easy-to-use features to make changes. This improves the traditional process considerably, since commonly business users would request changes and wait for suitable resource from the IT department, whereas now the administrator need only grant a business user the appropriate security role and the changes can happen as needed.

These changes, like other MDS-based extensibility, are future-proof, meaning that the original base content can be upgraded or patched, and any changes are just reapplied on top.

Content Access

The user assistance content is available in essentially three ways. First is through *embedded help*. One example is the bubble-help available for most fields on each page, which provides a few explanatory words. In addition, some fields and buttons also offer pop-up help icons that provide a little more detail. A final example of embedded help is the format examples and alerts that appear for fields that accept only specific data types, such as dates and numbers. Embedded help provides almost everything required to understand and use the components on each page.

The second type of content is more wide-ranging, focusing on completion of work rather than usage of pages. This covers everything from understanding business process flows, to how to set up and complete detailed tasks. This content is known as *non-embedded help* and is held within the *help portal.* It comprises the seven different forms of content listed earlier (FAQ, Key Concepts, and so on) and is somewhat equivalent to the online help of traditional enterprise applications. When accessing this content, Fusion Applications shares the context of the active page with the help portal, ensuring that everything displayed is relevant. In the help portal, the extensibility, tagging, and other features come into their own, allowing all content to be tailored as required.

The third type of content is somewhat traditional, but nevertheless exceptionally valuable. Some content, such as installation instructions and system administration information, doesn't neatly fit into a functional Business Process Model, and it doesn't really work in the context of the application features. This type of content, mainly related to back-end system management, is provided as static PDF documents. That's not to say that some of it is not accessible from the help portal, as some functional setup documents are, but for the most part it forms a comprehensive offline library.

The Supportability Architecture

Most enterprise application architectures are based around execution and capabilities, helping them deliver a long list of feature functions. Although this still remains true for Fusion Applications, an additional underlying architecture runs through the whole system, known as *supportability*. Supportability can be defined as the ease with which problems can be diagnosed and resolved, or more simply put, how supportable the software is. Fusion Applications has a baked-in range of support-related technologies, utilities, design patterns, and best practices, so that when something fails, the relevant details are captured and provided for further analysis. In the future it's likely that this will form the basis for automated system diagnosis and may be self-healing, the panacea for delivering lower total cost of ownership (TCO). For now, a few specific pieces make the applications administrator's job substantially easier.

Diagnostic Logging and Tracing

Fusion Applications comprises several business logic execution environments, such as Java, PL/SQL, and BPEL, which all work together to provide the application features and functions. Should the code in any one of these environments fail to perform as expected (exceptions, poor performance, or simply illogical results), running inside such a complex technology stack it would be difficult to identify precisely what and where the failure occurred. Fortunately, Fusion Applications contains a unified logging mechanism, so that all code writes its debug output (including exceptions, data, and processing statuses) in a consistent manner, in standard format, and to a uniform location.

Later chapters provide full explanations of how these logging mechanisms work in each execution environment. We'll look at how they can be configured, monitored, and their content used for successful applications administration.

Application Diagnostic Tests

With thousands of business features that can be used in hundreds of different configurations, it's not just the software code that needs support, but also the setup and use of these features themselves. By starting the problem analysis at the highest level—looking at what the business user is trying to do—many issues around misconfiguration and misuse can be avoided. The validation of business features is supported in Fusion Applications by the Diagnostic Test Framework (DTF).

Similar utilities exist in other Applications products, such as E-Business Suite's own Diagnostic Tests, however, for Fusion Applications, this platform provides a very flexible and powerful method for business users to begin troubleshooting from within the product itself.

In summary, this framework offers the following key features to help detect and diagnose problems in application functions and features:

- Each diagnostic test provides a detailed report output. Tests can also be chained together so that one report can contain a full suite of related diagnostics.
- A separate dashboard is provided for managing all diagnostic tests and their execution, accessible from anywhere within Fusion Applications. Test runs can also be prescheduled and batched using the dashboard.
- Each diagnostic test is fully secured based on the standard application Data and Function Security access privileges.
- Tests can accept one or more input parameters, allowing them to be executed for specific sets of data. For example, tests can be run for individual transactions or within a particular context, such as for a particular user or business unit.
- The use of diagnostic tests remains flexible to suit specific purposes and needs. They can perform such functions as validating technical and functional configurations, qualifying the setup of business objects (such as enterprise structures), verifying transaction and data integrity, and checking the results of process execution. Tests can also directly call application logic to validate parts of a process.
- Although commonly used when reacting to a problem, these diagnostic tests can also be used in a proactive manner to check for potential problems that, if not resolved, might have severe downstream effects. For example, prior to running the processes for the close of a financial period, diagnostic tests can be run to validate the health of the functional setup and the transactional data, and any corrections can be made before processing begins.
- Each diagnostic test is associated to a business process, making it immediately accessible to business users, directly within the context of a problem. In addition, like the help content, diagnostic tests can be custom tagged for such purposes as linking a specific test to one specific event.

Diagnostic Framework

With the 11*g* release of Fusion Middleware and the Database Server, Oracle has introduced an underlying platform for the capture of system information upon any serious failure. This is known as the Diagnostic Framework (DFw), and its adoption extends to Fusion Applications as well.

In summary, this framework provides a detection mechanism so that whenever a problem with a predefined set of symptoms (or signature) occurs, a dedicated process is launched and collects all associated system information, such as logs, traces, dumps, and diagnostics. Fusion Applications leverages this framework so that when the product logic handles a particularly serious error, it will automatically raise an associated alert (known as an *incident*) and ensure all related information is captured.

Interestingly enough, in Fusion Applications this framework actually leverages the two previously discussed supportability features, whereby Fusion Applications incidents will include application diagnostic logs and traces, plus they will execute any diagnostic tests that are associated with the problem signature.

Next-Generation Manageability

Although Fusion Applications is more complex than its other Oracle Applications cousins, more tools and utilities are available to manage that complexity. This book is intended to provide information in a meaningful and applicable way. As such, we will look in detail at the various dashboards, consoles, and systems management programs that exist, both in the technology stack as well as those extended to be part of Fusion Applications.

Although the discussion will focus specifically around Oracle Enterprise Manager as the central systems management tool, to encompass the complete Applications Management remit (as discussed in the next chapter), the focus is somewhat broadened. We'll look at various other utilities and techniques that, when combined, offer a deeper overall awareness, covering all aspects, including the health of the business processes and the application processing, performance of all the supporting technology stack components, and monitoring such things as the database, the operating system, and even the network and hardware.