

CHAPTER 3

Oracle Fusion Middleware
and SOA Suite 11g



St. Matthews is facing several real business challenges, as we have seen in the previous chapter. The hospital has well-defined views as to how to approach these challenges from the business perspective as well as the implementation or technological perspective. Service-Oriented Architecture (SOA) as an overarching design principle is a key element in this approach. To apply the concepts laid down in St. Matthews' architecture blueprints, the hospital needs to put in place a technology stack that supports those architectural concepts. Chapter 2 made the opening moves in this direction.

St. Matthews has evaluated a number of SOA and middleware offerings from various vendors—including open-source products—and has picked Oracle Fusion Middleware (FMW) as its preferred stack. Among the selection criteria, support for open industry standards ranked very highly, along with product functionality, operational (administration) effort, and maturity. Because it is a long-running and rather satisfied user of Oracle technology—RDBMS and various tools, including Oracle Forms—the IT staff at St. Matthews has a natural bias toward the Oracle Fusion Middleware offering. It hopes and expects that FMW will offer additional advantages, such as even better integration with the Oracle technologies already in use (for example, PL/SQL), that alternative products wouldn't provide, in addition to a smoother learning curve based on the current skill set.

This chapter introduces Oracle Fusion Middleware in general and then focuses on the SOA Suite, one of its key components. It first offers an overview of the history of SOA and middleware within Oracle. Then it paints the broader picture of Fusion Middleware before concentrating on the SOA Suite and the products and technologies FMW comprises. The chapter concludes with the installation of the SOA Suite 11g and briefly touches upon the migration from previous releases of the Oracle SOA Suite. At the end of the chapter, we'll have the SOA Suite up and running, ready for the development of services.

History of Middleware and SOA in Oracle

Oracle aspires and even claims to be the number-one vendor in the middleware market. Whether or not that claim is justified, and regardless of what being “number one” in middleware exactly means, it is very clear that Oracle has undergone a dramatic makeover from not having a meaningful presence in middleware market at all to being at least one of the dominant players. How did that makeover happen? Why did it happen? And what is middleware anyway?

An exact definition of middleware is almost impossible to give. Historically (from the mid-1980s), middleware was very much associated with messaging and queuing, brokers, transaction monitors, and distributed processing. The essence of middleware is in connecting software components and applications, enabling interoperability between various platforms and different systems, and supporting the automated, structured exchange of data across the IT landscape. As such, middleware became the platform for Enterprise Application Integration (EAI). In more recent years, middleware has become the label for technology for (web) services and SOA. The term *middleware* now encompasses almost all software infrastructure required for implementing SOA. It has come to also be used for areas such as identity management, business intelligence, and content management. Note that although some vendors include databases in their definition of middleware, Oracle does not.

The Mists of Time—Until 2001

It is difficult to point out exactly when Oracle started doing middleware for real and what those initial activities were. It is much easier to see what those early efforts led to.

The Oracle RDBMS has been an interoperability platform from very early on: The RDBMS was released for all major platforms, and many not so major ones as well. Oracle Corporation has always invested a lot of effort in porting the C-based Oracle RDBMS software from Solaris—the primary release platform for many years until Linux (temporarily) usurped that position—to a host of other operating systems running on hardware from PCs to mainframes. Oracle database applications written on one platform will be portable to any of the other platforms that the RDBMS runs on. This means, for example, for PL/SQL programs “write once, run everywhere”—the much-touted Java tagline, was realized before Java even existed! Interoperability with non-Oracle technology was not as hot an issue at that time.

A first stab at messaging (infrastructure) was delivered through Advanced Queuing (AQ) in the Oracle 8.0 RDBMS release (1998). AQ is based on database technology, including tables and PL/SQL. It adds to the database the ability to implement, publish, and subscribe scenarios where applications asynchronously communicate messages. Note that AQ is still the backbone of persistent messaging in WebLogic today.

In the mid-1990s, client/server was all the rage, and in that two-tier architecture there was initially no place for middleware—a third tier. However, the rapid expansion of the Internet as well as several major challenges with the client/server architecture, such as maintenance effort and server-side scalability, resulted in the rise of the three-tier architecture and the introduction of the notion of an application server. An application server can be used to execute common business logic for all clients, thereby decreasing load for both the front end and back end. Also, through the type of functionality offered by this *middle* tier, it is the best place to position and do integration (using *middleware*).

The need for integration across different systems in the enterprise—and, later, also between enterprises—became more apparent and formed another force that drove the creation of middleware software. The fact that most organizations use technology from different vendors, and that integration therefore also means interoperability across various technology platforms, was one of the drivers for the development of industry standards.

Oracle announced its full support for the Java platform early on, in 1997. In 1998, the Oracle 8i RDBMS was released with a Java Virtual Machine (JVM) inside the database. Oracle also released several versions of its ill-fated OAS product, the Oracle Application Server. OAS 4.0, for example, released in 1998, debuted support for CORBA, an early interoperability standard. Based on OAS, Oracle announced the Oracle Integration Server (OIS) at the end of 1999, a platform for ... integration! OIS leveraged Advanced Queuing and introduced early versions of technology adapters for integration with packages or views in the database as well as some third-party ERP (SAP, PeopleSoft) systems. Other components new in OIS were Workflow and InterConnect. Until overtaken by the Oracle Enterprise Service Bus (OESB), InterConnect was the primary Enterprise Application Integration product offered by Oracle and it laid the foundation for several pieces of today's SOA Suite.

In 1998, the World Wide Web Consortium (W3C) published the 1.0 release of the XML standard. This turned out to be the foundation for almost every integration and interoperability initiative ever since. Oracle was quick to join the XML crowd. In 1999, the first release (of many) of the XML Development Kit (XDK) appeared. The 9iR2 landmark release of the RDBMS had built-in support for SQL/XML and the native XMLType data type. The XML functionality in the RDBMS was collectively labeled “XMLDB,” a term that today covers many of the more native database features around XML.

Industry Standards: From 1998 until Now

The evolution of middleware technology within Oracle Corporation took place alongside developments in the industry. Both commercial vendors and open-source products were released, competing with Oracle's offerings. More importantly, most vendors were collaborating in various consortia and standards bodies to create the industry standards that would bring such tremendous change to the world of middleware and the promises of interoperability. The true reason why SOA could bring the success and the results promised by various reuse and integration initiatives since the 1980s lies in the widespread commitment to open standards, among the commercial vendors as well as the open-source projects.

Many of the standards around the Web, Web Services, SOA, and interoperability are created and maintained by standards bodies such as the W3C, OASIS, and JCP, in close collaboration with many of the major industry players. Companies such as IBM, Microsoft, SAP, Sun Microsystems, BEA Systems, Hewlett Packard, Fujitsu, webMethods, Software AG, and, of course, Oracle, frequently join forces to further the evolution and widespread promotion of standards. Implementation of and compliance with these standards has become an important part of marketing efforts, and any product that fails to meet the standards' specifications will have problems competing with similar offerings that do support the standards.

XML (eXtensible Markup Language, inspired by HTML and its predecessor SGML) was the first (and foremost) standard in this area, sponsored by Microsoft and published by the W3C in 1998. XML is today the lingua franca as well as the main lubricant of SOA, Web Services, and other interoperability initiatives, as well as the foundation for many more specialized standards. The XML standard describes a set of rules for creating documents with structured data. XML itself is very generic—something like ASCII or comma-separated files with more structure. Its real value starts to shine in conjunction with standards and tools that describe and perform validation of the structure and content of the documents (XSD), retrieve pieces of information from the documents using structured queries (XPath), and transform documents into different structures (XSL-T). You will see many examples of these core XML technologies throughout this and any other SOA-related book.

Hot on the heels of these standards related to storing information in structured documents and manipulating those documents were standards for exchanging information captured in such documents. The first definition of SOAP (the Simple Object Access Protocol) saw the light of day as early as 1998. SOAP describes a simple envelope-style mechanism for combining payload and metadata in structured packages. In 2000, the Web Service Definition Language (WSDL) introduced the now-omnipresent standard for describing the contract for a Web Service—where the definition of a Web Service by now has been stretched to encompass almost any service and operation that deals with structured information. Fairly well known, though not overly successful, is the Universal Description, Discovery, and Integration (UDDI) standard, also dating from 2000. UDDI is intended to underpin directories of Web Services that tools can browse through in order to discover useful services. Despite its lack of immediate success, UDDI has certainly helped to promote the concept of service registries with listings of useful services that potential consumers such as developers can browse through. UDDI, SOAP, and WSDL can be seen as the first generation of the XML-based standards concerning Web Services. In 2004, the WS-I Basic Profile was published to complement this threesome—a set of guidelines on how exactly to apply these core Web Service standards, to ensure full

operability (the original standards allowed for multiple interpretations in certain areas that led to differences between vendors' implementations).

The second wave of standards in the area of Web Services is concerned with more advanced concepts around message exchanges, such as the policies that apply to the message, the security of the messages, sending them in a reliable way to guarantee the reception (in the proper sequence and without duplication) of messages, correlation of multiple messages sent over a longer period, and the specification of return addresses and other concerns. Together these standards are referred to as WS-*. Their names all start with WS, and collectively they constitute a framework for service-oriented message exchanges that make it useful to have a common denominator. The * is usually pronounced *splat* or just *star*. Important members of the WS-* family are WS-Addressing, WS-Reliable Messages, WS-Security, and WS-Policy.

The automation of business processes has always been an important objective for the IT industry. The complexity of many processes and the involvement of multiple IT systems and applications, as well as the required participation of humans, have stood in the way of process automation for a long time.

With the rise of Web Services to overcome the interoperability challenges, fresh opportunities started to open up for business process automation. New ways to describe business processes in a structured way started to appear in 2004. The most prominent examples are Business Process Modeling Notation (BPMN) and Business Process Execution Language (BPEL, defined through BPEL4WS and WS-BPEL). Process definitions include the process routing and decision logic, calls to Web Services, and tasks to be performed by people. As the standards evolved, engines to execute such process definitions were developed by various vendors. BPEL4People and WS-Human Task (2007) have added human task-oriented extensions to BPEL that, by itself, is a rather technical Web Service-oriented language. Table 3-1 provides a chronological overview of some IT industry standards and specifications relevant to middleware and Service-Oriented Architecture.

Service Orchestration and Composite Services

In the previous chapter, we discussed elementary services and business services. The latter are coarser grained, offering more specific and complex functionality, leveraging the elementary services to help provide their functionality. Composing the coarse-grained services takes a combination of flow logic, calls to other services, and logic to process the results of those calls.

The industry had recognized the challenges of service composition or orchestration. In many cases, a number of services need to be invoked to accomplish a certain task, and only when all services have been called and delivered their response is the task done or the composite service complete. If a service call fails—either because of a technical issue or because of a business exception—the task may need to be undone or may need additional steps to overcome the problem. Multiple service calls can be made in parallel. Calls can be made to synchronous services, which send their reply as the return message to the request, and also to asynchronous services, which call back at some later point in time to deliver their response. These clusters of service calls or composites can represent a real business process or implement a composite service. Instances of such composites that perform service orchestration can be long running—up to days or even months when real business processes are implemented. Multiple instances of the same composite can be active at the same time,

Standard	Year of Original Publication	Current Release and Year Published	Standards Body	Purpose
Enterprise Java Bean (EJB)	1997/1999	3.1, 2009	JCP	JEE specification for exposing and accessing remote Java-based business logic
XML	1998	1.1 (2nd edition), 2006	W3C	Flexible yet structured language for creating text documents
SOAP	1998	1.2, 2007	W3C	XML-based protocol specification for exchanging messages with Web Services
XPath	1999	2.0, 2007	W3C	Query language for retrieving information from XML documents
XSLT	1999	2.0, 2007	W3C	Style sheet language for describing transformations for XML documents
WSDL	2000	2.0, 2007	W3C	XML language for describing the Web Services contract
UDDI (Universal Description, Discovery, Integration)	2000	3.0, 2004	OASIS	XML language for publishing a registry of (web) services
XSD	2001	1.0, 2001	W3C	Schema language for defining the valid structure and rules for XML elements (and successor to DTD)
Java Message Service (JMS)	2001	1.1, 2002	JCP	JEE specification that describes a Java API for loosely coupled, asynchronous interactions through Message Oriented Middleware
Java EE Connector Architecture (JCA)	2001	1.5, 2006	JCP	JEE specification for creating adapters to connect Java with Enterprise Information Systems
Security Assertion Markup Language (SAML)	2002	2.0, 2005	OASIS	XML-based standard that describes how security-related information (identification, authorization) can be exchanged
Web Services for Remote Portlets (WSRP)	2003	2.0, 2008	OASIS	Specification for interaction between portals (Portlet consumers) and (remote) Portlets (Web Services with a user interface)
WS-Reliable Messaging	2003	1.1, 2007	OASIS	A wire protocol used in SOAP messages to ensure reliable transport between sender and receiver

TABLE 3-1. *Chronological Overview of Some IT Industry Standards and Specifications Relevant to Middleware and Service-Oriented Architecture*

Standard	Year of Original Publication	Current Release and Year Published	Standards Body	Purpose
Business Process Execution Language (BPEL4WS/ WS-BPEL)	2004	2.0, 2007	OASIS	Executable language for processes that interact with Web Services
Service Data Objects (SDO)	2004	2.0, 2005	OASIS	Data-programming architecture that facilitates working with structured data objects in Service-Oriented Architecture
Business Process Modeling Notation (BPMN)	2004	1.2, 2009	OMG	Standard for ways to graphically describe business processes—and as added bonus simulate or even execute those processes for real
WS-I Basic Profile	2004	1.1, 2006	WS-I	Specification on how to apply standards such as SOAP, WSDL, and UDDI in order to achieve true interoperability across technology stacks
WS-Security	2004	1.1, 2006	OASIS	Specification on how to apply security—for example, through SAML or Kerberos—to Web Services and SOAP messages
WS-Addressing	2006	1.0, 2006	W3C	Standard that provides transport-neutral mechanisms to address and identify Web Service endpoints and to secure end-to-end endpoint identification in messages
XQuery	2007	1.0, 2007	W3C	Programming language for querying collections of XML data (technically a subset of XPath)
BPEL4People and WS-Human Task	2007		OASIS	Extension of BPEL4WS to specify interaction with humans and a human task-definition language
Service Component Architecture (SCA)	2007	1.0, 2007	OSOA	Configuration language for describing composite applications based on service components
WS-Policy	2007	1.0, 2007	W3C	XML language for describing the policies—such as Security and Quality of Service—that apply to a Web Service
WS-I Basic Security Profile	2007	1.1, 2009 (approval draft)	WS-I	Specification on how to apply security standards such as WS-Security in order to achieve true (security) interoperability across technology stacks

TABLE 3-1. Chronological Overview of Some IT Industry Standards and Specifications Relevant to Middleware and Service-Oriented Architecture (Continued)

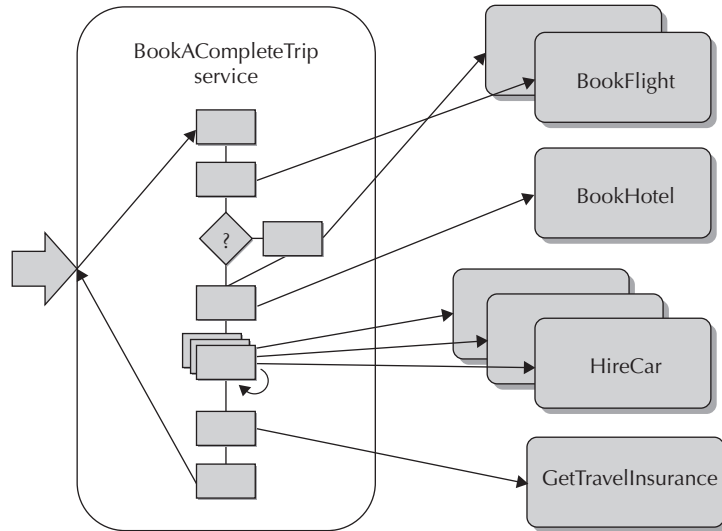


FIGURE 3-1. An example of service orchestration, combining external services and internal logic and exposing a single coarse-grained business service

handling different requests from potentially different clients. Figure 3-1 shows an example of service orchestration where the `BookACompleteTrip` service orchestrates the flight, hotel, car, and insurance-related services.

BPEL was the first language for implementing service orchestration—programs written in BPEL invoke (usually multiple) Web Services, perform process logic, handle faults in service calls, coordinate transactions, and deal with events and timeouts. Programs written in BPEL are known as *BPEL processes*. These processes run in a BPEL container. They are usually started through a call to the Web Services interface published for the BPEL process or initiated by the consumption of an event.

Through BPEL, many organizations successfully created SOA applications: programs or service composites that implemented business logic largely by orchestrating services that do the real work. However, even though BPEL can orchestrate multiple service calls and perform many of the actions required in a service composite, the BPEL container will usually work with other engines to execute all logic in a process and the services it invokes. For example, pieces of computing logic are implemented in Java; a workflow engine is engaged to handle human tasks; adapters are required to interact with databases, message queues, and a file system; and a rule engine implements decision logic that determines the routing through the process at junctions. Service composites usually also involve a fair bit of message transformation, filtering, and routing.

When the BPEL process is really the implementation of a business process, chances are that it will be long running (longer than a few seconds) and will carry some state data—data that it hangs on to for a large part of its lifetime. BPEL processes can also be used to create composite services—the coarser-grained next level from elementary services. In that case, a BPEL process instance will complete much faster and should not be considered stateful.

Note that in many cases the BPEL process could by itself take care of most of the work that can be delegated to other service engines—and before the various dedicated service engines were available, many BPEL processes were developed that actually do all the work themselves. However, for reasons of performance, functionality, and development productivity, it is better to use the best tool for the job. Equally important is the management agility: With all logic programmed into the BPEL process, every little change requires a redeployment of the entire service composite. Especially with long-running processes, such redeployments are not a trivial affair. Better to have the BPEL process collaborate with logic running in other engines that are more easily reconfigured and have different procedures for configuration and deployment. Some engines, for example, support a great deal of run-time configuration.

Recently we have seen a shift away from BPEL: True business processes are increasingly implemented using engines running BPMN. BPEL seems to be primarily an integration engine that combines services to provide the type of coarse-grained service we discussed earlier.

Service Component Architecture (SCA)

With on the one hand the obvious success of BPEL for creating service composites and on the other the ongoing challenges, many vendors working on Service-Oriented Architecture have joined forces to come up with a new standard for creating service-oriented applications.

The objective of this standard was to allow the development of a service-oriented application that could implement every piece of its functionality in the language and run-time engine best suited for it and still have all the pieces integrated in a simple, standardized way, based on the established standards for services. It is assumed—though not required—that the application will invoke several external services. It is also expected that the application will expose a service interface itself. Reuse is an important theme in SOA and is also a key objective for this new standard: Applications developed according to the standard are reusable components that can be included in other applications.

This standard is called Service Component Architecture (SCA), first published in 2007. It is expected to become the dominant guiding principle, according to which vendors build their SOA containers, and thus the framework for development of SOA applications.

The promise of SCA is that developers can use various languages running on different run-time engines to implement various parts of the application—for example, BPEL, Java, another SCA composite application, a rule engine, a workflow engine, and technology adapters to work with databases, queues, and file systems. Each such part of the application is called a (service) component. Each service component publishes a contract that describes its interface through a WSDL document. The developers specify the functional link between these different parts of the application, and it is up to the SCA container or run-time engine to facilitate communication between the components in the most efficient way, usually through a native, binary communication protocol.

The coupling between service components is very loose; they can work together without any knowledge about each other's implementation. This way of creating composite applications is very flexible: It allows replacement of one service component with another that, as long as it fulfills the same contract, could be implemented in an entirely different language running on another service engine.

SCA is not just for easier and more productive development of SOA applications. It also specifies how the behavior of the application can be made configurable to allow administrators to apply changes in the behavior without redeployment of the application. Changes in the location of services called from the application can be changed at run time without impact on the availability of the application. Quality of Service aspects, including security policies and reliability requirements, can be (re)configured during or even after deployment.

SCA helps to simplify the assembly and deployment of composite applications. An SCA composite application can be assembled from a collection of SCA composites and then turned into deployable units.

Not Invented Here (2001–2008)

Around the turn of the century, Oracle had strong support for queuing, Java, and XML in its RDBMS. In addition, it had a number of what one might call “middleware products” that were mildly successful at best. OAS was struggling in a market dominated by IBM WebSphere, BEA WebLogic, Sun IPlanet, and several open-source products. Oracle Integration Server did not get any real traction, and Oracle Corporation seemed to be looking for a product strategy that would make it a serious contender in the now rapidly evolving market for middleware.

During these years, we have seen a slowly but steadily evolving tendency at Oracle HQ to no longer insist on building every technology and product itself. In 1994, Oracle acquired the RDB database from Digital (Equipment Corporation), an acquisition that worked out very well. It demonstrated how Oracle would generally treat products it acquired: continue and frequently even intensify the development of new product releases. Maybe that experience prepared the organization for what was to come in later years.

In 1998, Oracle released JDeveloper 1.0, its first ever Java Development IDE, based on Borland’s JBuilder product. This initial release consisted of over 90 percent JBuilder code, and was a rather feature-poor product in the eyes of today’s Java developers. However, its significance was huge. Next to the traditional development tools created within Oracle, such as Oracle Designer, Oracle Forms, Oracle Reports, and Discoverer, Oracle offered a tool for Java (web) developers. And it would continue to evolve JDeveloper to become the strategic development platform of Oracle. In 2002, the JDeveloper 9i (9.0.3) release meant the end of the last JBuilder remnants in the products. Yet the experience with a product that started out based on a third-party code base probably also contributed to the acceptance of externally developed products at the core of new Oracle initiatives.

In 2001, Oracle radically changed its Application Server tactics. Rather than struggling on with its own OAS product, it struck a deal with a Norwegian company called Orion to license its application server. Orion became the heart of Oracle Containers for J2EE (OC4J) (initially the *J* in OC4J stood for Java), the engine for the Oracle Internet Application Server (iAS). iAS quickly expanded its reach to become the middle-tier platform for such diverse products as Portal, Forms, J2EE Applications, Workflow and InterConnect, Discoverer, Oracle Single Sign On, Oracle Internet Directory, and many more.

Support for Web Services in the Application Server started to appear from 2001 onward, and was complemented by development facilities in JDeveloper almost from the start. Shortly afterward, support for UDDI was added to the Oracle Internet Directory product. The year 2003 saw the announcement of Oracle’s implementation of WSRP (WebService for Remote Portlets), a standard that facilitates the integration of Portlets (services with a user interface) across vendor platforms.

In 2004, the cascade of acquisitions started in what was by that time becoming known as the “SOA space.” Oracle acquired dozens of companies and products, most of them to be folded into the Enterprise Applications portfolio, but a substantial number in middleware as well. All of these acquisitions quickly helped to put together a suite of products and technologies that covered most of the requirements for implementing Service-Oriented Architecture. The acquisition of Collaxa for its BPEL Process Manager (2004) laid the foundation for the later SOA Suite 10g product. Thor, Oblix, and OctetString were acquired in 2005 for their various security, identity and access

management (IAM), and services management related offerings. The Oracle WebService Manager (OWSM, frequently pronounced *awesome*) was based on the technology in these products.

As part of the 2005 acquisition of PeopleSoft, the Business Activity Monitor (BAM) was added to the growing range of middleware products. Also in 2005, Oracle gave up its own UDDI implementation and instead decided to license the Systinet Service Registry from Systinet Corporation (bought by Mercury a little later, which itself was acquired by Hewlett-Packard in 2006). In 2006, a partnership was entered into with IDS Scheer, a leading vendor of business process modeling software (acquired by Software AG in 2009), that allowed Oracle to offer a product—Oracle Business Process Analysis (BPA) Suite—based on ARIS.

Another major chunk was swallowed later in 2005, when Siebel Systems was acquired. In terms of middleware, Siebel brought what was to become the BI Enterprise Edition to the table.

Amid all these influxes from the outside, Oracle developed its own Enterprise Service Bus (ESB)—released in 2006. The Oracle ESB leveraged a lot of the work that had gone into developing InterConnect and a wide range of technology adapters, and did this based on standards for XML and Web Services. The first incarnation of the Oracle Rules Engine was published around that same time. With the ESB, Rules Engine, WebService Manager, and BPEL Process Manager components packaged together, Oracle released SOA Suite 10g in early 2006, accompanied by JDeveloper 10.1.3 with the design-time environment for these components. Even though the various components were not extremely well integrated, the suite, together with iAS 10g, offered a wide range of functionality for developing, deploying, and managing SOA implementations. Table 3-2 lists Oracle's most relevant acquisitions and OEM partnerships around middleware technology (see <http://www.oracle.com/us/corporate/acquisitions/index.htm> for a complete, up-to-date list of product and vendor acquisitions).

With this first generation of the SOA Suite in place, the next step was to be a much more integrated SOA platform with all the various pieces really integrated together. The outline of SOA Suite 11g—the core of the Fusion Middleware platform—was becoming clearer from 2006 onward, with initial technology previews being published starting in late 2007. However, plans were to be changed dramatically in the course of 2008, as Oracle's largest technology takeover until then unfolded. Among its products were the AquaLogic Service Bus (an ESB), BPM Studio (a BPMN-based product for Business Process Modeling), the Enterprise Repository (for governance of services and other IT artifacts), AquaLogic Data Services, Tuxedo (for managing transactions across distributed systems including mainframes), and a number of portal products.

When the Oracle-BEA deal was closed at the end of April 2008, the architects of Fusion Middleware returned to the drawing boards. BEA's products offered valuable opportunities for improving the pending SOA Suite 11g that simply had to be taken advantage of—at the cost of a regrettable delay in its original release schedule. The entry of WebLogic into the Oracle stable meant, for example, the early retirement of the OC4J-based Application Server line: All Oracle's middleware were to be delivered on top of the WebLogic platform.

Other acquisitions that had an impact on the middleware product portfolio include Sunopsis (2006, rebranded Oracle Data Integrator), Stellent (2006), and Universal Content Manager and Hyperion in 2007—a solution for corporate performance management. Early 2010 saw the completion of the acquisition of a big (Glass)fish: Sun Microsystems. Sun was a prize for Oracle because of its hardware and its control over Java. As a bonus, Oracle gained control over many Sun products concerning identity and access management, Web Services, BPEL and integration, portals, as well as another JEE application server: GlassFish. None of these products gets the

Year	Type	Vendor	Product	Purpose
2001	OEM	Orion	OC4J	Java/J2EE application server
2004	Acquisition	Collaxa	BPEL Process Manager	Web Service Orchestration
2005	Acquisition	Siebel Systems	BI EE	Business intelligence, OLAP, reporting
2005	Acquisition	PeopleSoft	Business Activity Monitoring	Real-time analysis of business events
2005	OEM	HP (Mercury) Systinet	Service Registry	UDDI service directory
2005	Acquisition	Thor Technologies	Xellerate	Identity and access management
2005	Acquisition	OctetString	Virtual Directory (Engine)	Identity and access management
2005	Acquisition	Oblix	COREid, COREsv	Identity and access management/ Web Service management
2006	Acquisition	Sunopsis	Data Integrator	Data integration (ELT)
2006	OEM	IDS Scheer	ARIS/Oracle BPA	Business process analysis
2006	Acquisition	Stellent	Universal Content Manager	Content management
2007	Acquisition	Hyperion	Hyperion, ESSBase	Corporate performance management
2007	Acquisition	Bharosa	Bharosa Tracker and Authenticator	Security and real-time fraud detection
2007	Acquisition	Bridgestream	Role Manager	Identity and access management
2007	Acquisition	Moniforce	Moniforce	Web user experience monitoring
2008	Acquisition	BEA	WebLogic, AquaLogic Service Bus, Enterprise Repository, BPM Studio, Tuxedo, Portals	JEE Application Server, ESB, SOA governance, BPMN, transaction management
2008	Acquisition	ClearApp	ClearApp	Management of composite SOA applications
2009	Acquisition	Sun Microsystems	Hardware and infrastructure components: GlassFish, OpenESB with IEP, OpenPortal, OpenSSO, Identity Management, Portal, MySQL	Manifold, including middleware for running JEE applications, implementing SOA, processing events, and managing identities/ authentication and authorization
2009	Acquisition	GoldenGate	GoldenGate	Real-time data integration and continuous data (changes) availability
2010	Acquisition	AmberPoint	Governance System, Management System	SOA management and governance, security, business transaction management

TABLE 3-2. Oracle's Most Relevant Acquisitions and OEM Partnerships around Middleware Technology

“strategic” status at Oracle. This means that while some of their capabilities may be added to the existing Fusion Middleware components, none will replace a current FMW product. As such, the impact of the acquisition of Sun on the SOA Suite 11g was fairly limited.

More impact—at least on operational management, security, and governance of SOA applications—can be expected from the acquisition of AmberPoint, which was announced in February 2010.

Invented Here after All

Not all new middleware technology was seized from the outside world. Indeed, one of Oracle’s current flagship middleware products was built from scratch at Oracle—it simply did not exist anywhere in the world. WebCenter is a product for Enterprise 2.0, the enabler for enterprise-wide collaboration. It delivers the next generation of enterprise portals. WebCenter extends SOA concepts and services with a user interface through its support for WSRP Portlets. It provides the integration point at the user-interface level between many areas of functionality, such as content management, task management and workflow, enterprise-wide search and communication through e-mail, instant messaging (or chat), and VoIP (Voice over IP). It is also a natural fit for user interfaces that expose or leverage the services provided from the SOA.

WebCenter, Oracle Applications, BI Enterprise Edition, BAM Studio, Enterprise Manager, and an increasing number of other Oracle products are developed using the homegrown Application Development Framework (ADF). This framework contains ADF Business Components for smooth interaction with the database and ADF Faces, with a rich Web 2.0 library of user-interface components based on the JavaServer Faces (JSF) industry standard. The rich web applications are agnostic when it comes to their data provider—the ADF Model abstracts the underlying business service—and work equally well with a persistence layer based on JPA and EJB as well as Web Services.

ADF has facilities for very productive, declarative development with several reuse mechanisms and support for advanced data visualization, active (event-driven, server-push) user interfaces, and “design time@run time” (discussed later in this chapter) for application customization and personalization. Development of ADF applications is done with Oracle JDeveloper. The applications run on a J(2)EE application server, typically WebLogic Server.

ADF is used in the SOA Suite to implement the user interface for the human tasks that are part of the SOA applications.

Complete, Open, and Integrated—2009 and Beyond

Around the turn of the century, Larry Ellison stated that enterprises should not go out and select best-of-breed products from a range of different vendors that then would have to be integrated together by “the guys with the glue guns.” Much better, he said, to buy a pre-integrated suite—engineered from the beginning to fit together—that does not require such a “glue gun” approach. That pre-integrated suite, by the way, was supposed to be the Oracle 11i eBusiness Suite.

Clearly this strategy was not embraced by the marketplace, and organizations continued to acquire best-of-breed products that required integration. Oracle itself started to do the same thing, as was described in the previous section. With PeopleSoft (HRM/HCM), Siebel (CRM), Retek (Retail), Portal (Billing), and many other applications, Oracle bought itself an impressive range of best-of-breed products that required ... integration. At the technology level, by the way, it did something very similar. With Stellent, BEA, Sunopsis, Hyperion, and others, Oracle acquired superior, market-leading alternatives to some of its own products. Of course, these products, too, needed integration.

The continuing and even increased need for integration of business applications provided Oracle with a market opportunity. It could make money in the middleware space to support all the integration efforts going on. In fact, how credible would the position of the Oracle Application Server platform be without support for integration and service orientation? Providing solid and functionally rich middleware was probably not just an opportunity as much as a necessity.

Apart from the external drivers, there was an urgent internal driver that was probably the most pressing one: Customers running a combination of modules from Oracle eBusiness Suite, Siebel, Retek, and PeopleSoft demanded of Oracle that these modules work together smoothly—something they obviously had not been designed for. Although this requirement posed a huge challenge, it was a challenge quite similar to the ones facing most enterprises: how to make legacy, custom, and COTS (commercial off-the-shelf) applications work together. Oracle’s middleware had been pushed by the Oracle marketing teams and sales force as *the* solution for such challenges, so now it was time to put their money where their mouths were—or, as is the usual expression within Oracle, it was time to “eat your own dog food.”

Based on the Oracle 10g SOA Suite, the Application Integration Architecture (AIA) was developed. AIA provides process integration packs (PIPs), collections of BPEL processes that implement business processes across various Oracle Applications products (for example, order processing across modules in EBS and PeopleSoft). Underpinning the PIPs is the AIA Foundation Pack that contains Enterprise Business Objects and Enterprise Business Services, which allow organizations to create their own customized business processes on top of the Oracle SOA Suite, spanning multiple modules from different products in the Oracle Applications portfolio and also legacy applications, including SAP modules and custom-built applications. AIA is crucial to Oracle’s strategy with regard to its Enterprise Applications portfolio. AIA’s requirements will further drive the development of Oracle’s SOA products, and its success will provide clear proof of the value of those products as well as a reference implementation with best practices, reusables, and guidelines for organizations using the SOA Suite for their own SOA implementation.

Fusion

With the acquisition of PeopleSoft, Oracle announced its Fusion vision and roadmap. Later acquisitions had their impact, not so much on the vision itself but certainly on the roadmap and timelines. There has been a lot of confusion as to what Oracle’s Project Fusion entailed. At the core, “Fusion” has these aspects:

- The integration of the acquired business entities into Oracle, reorganization as well as staff retention, especially among engineers
- A newly developed next-generation application that is based on industry standards and the latest technology and that takes the best features, flows, and usability traits from the existing application products; this new product is known as Fusion Applications
- Technology for making different products in the Oracle Applications portfolio—such as EBS, PeopleSoft, Siebel, Retek, and JD Edwards—work together, as well as the technology stack for the new Fusion Applications; this technology has been labeled Fusion Middleware

The adage Oracle uses for Fusion Middleware (FMW) is “complete, open, and integrated.” This captures the essence of the objectives with and claims for Fusion Middleware.

Complete means that all capabilities in every middleware area you can think of are provided by Fusion Middleware. And to put it even stronger, Oracle claims that every capability in FMW is

provided by the best-of-breed offering. So even if an organization wants to pursue a best-of-breed strategy, it would have to select Fusion Middleware components in every area because they are the best of breed in their own right. Oracle further strengthens the completeness claim with Fusion Applications as the living proof. Oracle has a unique ability to maintain the completeness in the future, given its resources, dedication, and internal needs.

Open refers to the fact that Fusion Middleware is hot-pluggable. This means that the FMW components can be replaced by alternative products from other vendors. Oracle recognizes the fact that even though it claims to provide a complete, best-of-breed solution in every area, organizations may have current investments or even deviating views as to which product is the best solution in a certain area of middleware capabilities. Another aspect of openness is the support for open standards. Fusion Middleware complies with every major industry standard in the area of middleware—some 195 standards are supported and adhered to in the FMW 11g release. This makes the product open in the sense that it can be interacted with in ways that are common across the industry. FMW is not open source, obviously, but does not tie an organization to Oracle proprietary protocols or hamper interoperability. Custom applications written for use with Fusion Middleware will run with alternative middleware platforms that also support the industry standards.

Now let's consider *integrated*. Not only does Fusion Middleware offer all middleware capabilities (*complete*), it also has all these capabilities nicely integrated and working together. That may sound trivial—if a vendor offers a number of middleware products, you would naturally expect them to work together. However, frequently that is not the case at all—and it wasn't the case for the 10g releases of the Oracle SOA Suite. Fusion Middleware provides that integration across different areas of functionality, such as business intelligence, Web Services, content management, enterprise collaboration, identity and access management, governance, event processing, and custom-developed user interfaces. It helps organizations create business processes that integrate with these different technologies across the enterprise.

Fusion Middleware 11g: The Innovative Foundation for Enterprise Applications

July 1, 2009, marked a milestone in the history of Oracle Corporation. On that day, the worldwide rollout of Oracle Fusion Middleware 11g was initiated, the culmination of many years of helping forge the industry standards, conducting research into interoperability, creating new tools and frameworks, acquiring and absorbing products from external parties, and architecting a complete stack of middleware products. The public unveiling of FMW 11g was one of the biggest product launches in Oracle's history.

Fusion Middleware 11g consists of many different products that provide solutions in diverse areas, from identity and access management, business intelligence, event processing, content management and data integration to a data grid, web application development, enterprise portal and collaboration, business process management, governance, security, and, of course, Service-Oriented Architecture. Fusion Middleware 11g runs on top of WebLogic Server 11g. The design time for most products in the stack is JDeveloper 11g.

Fusion Middleware is not sold as single product with a simple price tag. Oracle understands that many organizations will, at least initially, only use specific components from the wide range of middleware products. Customers buy licenses for specific product suites, bundles of related products in specific areas of functionality. Among the FMW 11g suites offered that are associated with SOA are the BPM Suite, EDA Suite, Governance Suite, and, of course, SOA Suite. Note that these suites have a certain level of overlap. Also note that for most suites, several reduced-cost variations are offered that support usage of only specific products from the suite.

The biggest customer for Fusion Middleware is Oracle itself. The development of Fusion Applications and other products in the Oracle Applications portfolio is all done on top of the Fusion Middleware 11g stack. Most organizations will have less stringent requirements for their development and integration efforts than the ones faced by Oracle's internal divisions. Because the exact same technology is available to external customers as is being used internally, Oracle is providing the proof in the FMW 11g pudding by eating all of it itself.

SOA Suite 11g: The Key Components

SOA Suite 11g is first and foremost an SCA container, a run-time engine that can execute composite service applications. Composite (service) applications are SCA-compliant applications that are assembled from various service components that are wired together internally based on WSDL contracts. Composite applications publish a service interface through which they can be invoked by external clients. This interface is frequently a (SOAP) Web Service interface, but other types of bindings are also possible, such as based on EJB/RMI and JMS. SOA Suite 11g can run multiple instances of every composite application in parallel. It can handle calls into applications, coordinate messages between components within an application, and facilitate calls from the application to external services.

You invoke a composite application that exposes a Web Service binding by sending an XML message to a URL. That message will be processed—possibly resulting in database manipulation, file creation, human task execution, e-mail sending, and event publishing. At some points during the processing of your message, you may receive return messages that can contain the results of whatever the application has been doing.

The composite applications running on the SOA Suite can make use of the following service languages and engines for executing its components:

- **BPEL Process Manager** Orchestrates (potentially) long-running service composites with many interactions with external services, both outgoing and incoming.
- **Decision Service or Business Rules engine** Executes decision logic that can be (re)defined at run time.
- **Human Workflow Service** For engaging humans in making decisions or providing information.
- **Spring-based Java Beans (as of 11gR1 PS 2)** Custom business logic implemented in Java acting on the messages.
- **Mediator** For filtering, transforming, adapting, and routing messages.
- **BPMN** Business process logic defined through BPMN can be executed inside the SOA Suite (by the same engine that also runs BPEL). (This, too, was introduced in 11gR1 PS2.)

Composite applications accept incoming request messages and route them through components programmed using these technologies. Note that other SCA containers may support different service engines—for example, running Cobol, C, and C#—and that Oracle may add new service engines to the SOA Suite as well. Each component performs a service that may alter the message, create new messages, have external effects, or influence the onward processing in the application. Composite applications can call out to external Web Services—and receive asynchronous responses or other incoming messages from these external services. Applications can also make use of the Event Delivery Network to publish business events as well as to consume such events.

Ingredients of a Composite Application

When developers create an SOA composite application, what they are actually doing is working on XML files. What gets deployed to the SOA Suite's SCA container typically is a collection of the following files:

- The WSDL and XSD files that describe the interfaces (contracts) of the application as a whole (the services it exposes) as well as the service components running inside the application.
- The files that are the programs to run in the BPEL and Mediator engines or that define the human task to be performed by an end user.
- Files that describe how the SCA components are wired together to exchange XML messages to be processed at run time.
- Definitions for how XML messages are to be transformed en route from one component to the next.
- Some of the XML files provide the configuration details for the adapters that the composite application can use to communicate to external technology platforms, such as database, file system, e-mail server, and message queues.
- Configuration plans that apply environment-specific deployment details.

Most of the XML, by the way, is hidden from view by visual editors that present far prettier and easier-to-understand renditions of those blocks of XML data.

All of the XML files are bundled together in archives—a JAR (Java Archive) or SAR (Service Assembly Archive, aka SOA Archive, a deployment unit that describes the SOA composite application)—that are deployed to the SOA Suite container.

The SOA Suite is shipped with an impressive set of technology adapters. These adapters speak a specific protocol and language to some external technology platform on one end and act like a Web Service on the other. Examples of these protocols, platforms, and languages include the file system, FTP servers, the database, JMS queues, the eBusiness Suite, SAP, and various B2B exchange types, such as RosettaNet, ebXML, HL7, and EDI(FACT). These adapters make it possible for SOA applications to connect to many different components and thereby service-enable existing assets.

Outside of the SCA container—but still part of the SOA Suite license and prepared for integration with the SOA composites—are several other valuable products: the Oracle Service Bus (OSB), Oracle Business Activity Monitoring (BAM), and Oracle Complex Event Processing (CEP).

Adapters

SOA Suite 11g is integrated with a large number of adapters that the composite applications can make use of for accessing services across various technologies and protocols. These adapters allow the composite applications to retrieve data from, forward messages to, and leverage functionality in many different places in and even outside the enterprise—from database and file systems to EDI trading partners and legacy applications. Some adapters allow for the activation of

composite applications from the outside world. The most important adapters available for use in SOA Suite 11g composite applications access the following targets:

- **Database** For accessing tables and views (query and data manipulation) and calling PL/SQL program units
- **File and FTP** For reading and writing files from a file system and an FTP server
- **Queues** For accessing queues through JMS, Oracle Advanced Queuing, and MQ Series
- **Enterprise Java Bean (EJB)** To communicate with remote Enterprise JavaBeans
- **Sockets** For reading and writing data over TCP/IP sockets
- **Oracle Applications (aka Oracle eBusiness Suite adapter)** For retrieving data from and sending data to eBusiness Suite (11*i* and 12)
- **Business Activity Monitoring (BAM)** For sending data and events to an Oracle BAM server
- **ADF-BC (Business Components)** For interacting with an ADF BC–based Service Data Object service
- **B2B** For the exchange of business documents with e-commerce trading partners based on industry standards such as RosettaNet, HL7, and various EDI protocols; also support for interaction with SAP and other ERP applications

Adapters will usually be called by the service components running in a composite application (outbound). Note, however, that most adapters can also initiate a new instance of an application (inbound). The database adapter, for example, can “poll for changed records,” and any new or changed record can start a new instance. Likewise, the file and FTP adapters can poll for new files to arrive on the file system or an FTP server, or for new lines in an existing file. This adapter, too, can instantiate a composite application instance when new data is read from a file that has changed.

Other adapters that can act as “service clients” that create new instances of composite applications include the EJB adapter, the JMS adapter, and the AQ adapter. Inbound adapters can connect to existing instances of composite applications (see Chapter 6 for details).

Adapters are discussed throughout the book: For example, the database adapter is discussed in Chapter 5, the file adapter in Chapter 7, the JMS adapter in Chapter 12, and the ADF BC adapter in Chapter 20.

Event Delivery Network

Business events are situations of potential interest. Examples are the reception of an order, cancellation of an appointment by the patient, the failure of a credit check, the crossing of a stock threshold (we are critically low on Band-Aids), and the acceptance of a job offer by a new nurse. These events frequently occur in business processes when certain conditions are met or actions have been performed. Events can also come into existence during the execution of a service component. The business events may trigger the start of new composite application instances or could notify already running instances.

However, the burden of informing any potentially interested party of the event should not be on the service component that happens to encounter the situation. The producer of the event—the application or service component that causes or encounters the situation that is deemed to be

of business interest—is not responsible for what happens with the published event, nor does or should it care. This keeps producers and consumers decoupled: Consumers can be added or removed without impact on the producers of events. Likewise, new producers can be introduced without any effect on the consumers. To make this happen, we need a “man in the middle” of sorts, a generic medium that deals with both consumers and producers.

A key part of SOA Suite 11g is the Event Delivery Network (EDN), an intermediary that takes on the responsibility of receiving events from producers and delivering them to interested parties.

Business events are defined across services and composite applications as an extension of the canonical data model. The definition of a business event comprises a name, possibly custom headers, and the definition of the payload. These definitions need to be registered with the EDN.

Service components—BPMN, Mediator, and BPEL—can publish events (occurrences of one of the predefined event types) to the Event Delivery Network. Events can also be published to the EDN from ADF applications and through a PL/SQL API.

Service components such as Mediator, BPMN, and BPEL register their interest in one or more of the centrally defined business events with the EDN. Such an interest can indicate all events of a specific type, but also can include more fine-grained selection rules that refer to the custom headers or payload to filter on specific occurrences of an event. When an event has been published, the Event Delivery Network will make sure that all interested parties will receive the event. Note that it is very well possible that an event is not delivered to any interested party at all. In that case, it disappears into the void.

Chapter 9 discusses the Event Delivery Network and presents several examples in detail.

Oracle Service Bus

Composite applications running in the SOA Suite will frequently need to access services made available in an enterprise service bus (ESB), possibly based on services running in other SCA containers, offered by external parties, or running on legacy platforms such as mainframes. In a similar vein, the services exposed by the composite applications within the business domain may need to be made available to a wider audience; this, too, is typically done through an ESB.

SOA Suite 11g contains an ESB: the Oracle Service Bus (the successor to BEA’s AquaLogic Service Bus, abbreviated OSB).

Chapter 13 describes the OSB and how it can be used along with SOA composite applications.

Business Activity Monitoring (BAM) Server

Oracle BAM provides a framework for creating dashboards that display real-time data as it flows into the BAM server. This is typically data received from physical sensors (security gates, RFID scanners), trace details from computer applications (request logging in web applications, process progress signals from a BPM or workflow engine), or live data feeds with financial data, weather reports, or even sports statistics. Rules can be created in BAM to instruct the framework to highlight deviations and send alerts under specified conditions. BAM is primarily used to monitor aggregates against predefined thresholds for data recently received over relatively short periods (typically minutes to hours, rather than months to years). That, along with the built-in capability to trigger alerts and take actions, is the main distinction between BAM and traditional business intelligence, which tends to be more passive and more historically oriented. BAM tries to facilitate the operational control of business process execution.

Data used by BAM for the actual reports is managed in memory in the Active Data Cache. Data is loaded into this cache in real time via various channels. Probably most important in the

context of the SOA Suite is the BAM Adapter—it is not only the fastest option for streaming data into the BAM server; it is also integrated into composite applications like all other adapters. For BPEL there is an additional option through the BAM sensor action that can be enlisted when adding special tracers to activities in the BPEL processes. Alternative routes for data into BAM are Direct JMS, Oracle Data Integrator, and through the Web Services interface exposed by the BAM server.

Chapter 19 introduces Business Activity Monitoring in detail.

Fusion Middleware Infrastructure and WebLogic Server 11g

SOA Suite 11g runs inside WebLogic Server 11g—the SCA container lives inside the JEE container. The underlying run-time infrastructure of Fusion Middleware 11g is the WebLogic Server platform, managed through the Administration Console. Several web applications are installed into the WebLogic Server domain as part of SOA Suite 11g to support the FMW 11g run-time operations. The Oracle Enterprise Manager Fusion Middleware Control Console is the most important one of these—other examples are the SOA Composer, the Worklist application, and several BAM web applications.

The Oracle Enterprise Manager Fusion Middleware Control Console is the integrated console for virtually all run-time monitoring and administration of SOA composite applications and their instances. This console is an ADF 11g web application that runs on WebLogic and is accessed from a browser by the SOA Suite administrator to work on tasks in these main categories:

- **Configuring** Adjusting properties from SOA infrastructure and service engines down to components in composite applications.
- **Monitoring** Aggregating metrics, performance figures, and faults across applications, components, and service engines; reporting the current state of running instances; providing an audit trail per composite instance; drilling down to the steps through a component; and inspecting the log files.
- **Managing** Deploying, stopping, and starting composite applications; recovering from faults; terminating application instances; unit testing of composite applications; and attachment of policies to SOA composite applications, service components, and binding components. The Oracle WSM Policy Manager is the integrated facility to attach policies regarding security, reliable messaging, addressing, and logging to Web Services and service composite applications.

The WebLogic Server Administration Console is used alongside the Enterprise Manager Fusion Middleware Control for normal JEE administrative tasks such as the configuration of data sources and JMS objects, administration of the security realm, and management of the technology adapters. Figure 3-2 shows the architecture of WebLogic Server and SOA Suite 11g installed on top of it.

User Messaging Service

Another element in the infrastructure is the Oracle User Messaging Service (UMS). UMS provides applications with two-way communication with users across various channels and protocols. Messages can be sent and received through e-mail, IM (XMPP), SMS (SMPP), and voice (VoIP). Most of the time, the message will be initiated by the application, but UMS also caters to

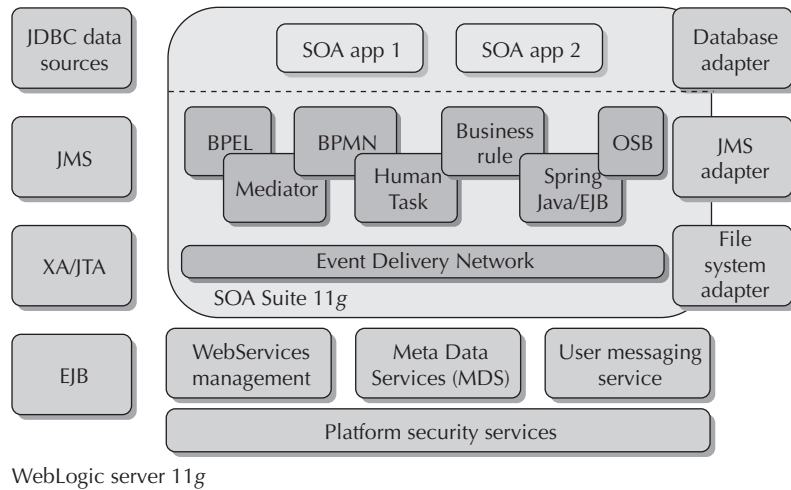


FIGURE 3-2. *WebLogic Server with the SOA Suite installed on top of it*

scenarios in which the user is the sender of the message and the application is on the receiving end. The arrival of the message is fed into the application by UMS.

Notifications will frequently be sent via UMS from BPEL processes, the Human Workflow engine, and the BAM server. WebCenter, ADF, and other web applications can also make use of UMS. UMS consists of a server that works with a number of drivers to connect with message gateways using specific protocols. These external gateways are not part of WebLogic Server or the SOA Suite. Various e-mail servers, chat (IM) servers, and external providers of SMS and text-to-speech services can be integrated.

Appendix C and the book's wiki provide instructions for configuring the UMS services.

Meta Data Services (MDS)

The Fusion Middleware run-time environment has at least one, and possibly multiple, metadata repositories that contain metadata for Oracle Fusion Middleware system components. A metadata repository contains metadata about the configuration of Oracle Fusion Middleware as well as metadata for different types of enterprise applications. Shared artifacts such as XSD documents describing the canonical data model, data value maps that describe mappings between business vocabularies in different domains, reusable transformations, human task definitions, security policies, business rule definitions, and business event definitions are deployed to and managed in metadata repositories. Artifacts in these metadata repositories can be used during development as well as at run time. Meta Data Services (MDS) provides a single interface across all repositories. MDS provides services to validate, version, tag and categorize, discover, and manage artifacts throughout their lifecycle.

A special facility in MDS is its support for customization. MDS can return specialized versions of artifacts that are created from a base version with context-sensitive deltas applied to it.

Design Time

The IDE (integrated development environment) used by developers to create service composite applications is JDeveloper 11g. JDeveloper is an IDE in more than one way. Most facilities required for developing software are integrated into a single workbench, including editors, debuggers, and support for testing, building, and deploying software artifacts. JDeveloper is integrated with WebLogic Server for easy deployment, execution, and debugging of web applications. JDeveloper also brings together the design time for many different products and technologies—from Complex Event Processor, BPM, UML, Java, and ADF to SQL and PLSQL (SQL Developer), and from WebCenter, Data Integrator, XML, and Web Services to all the technologies and service engines of the SOA Suite. That means JDeveloper is also the *integration* development environment.

Oracle Service Bus currently has two design-time environments: One is a browser-based console and the other is part of the Oracle Enterprise Pack for Eclipse. OSB support in JDeveloper is planned.

The slogan “design time at run time” is becoming fashionable, and describes the ability to change the behavior of already deployed applications at run time. Fusion Middleware supports various forms of this run-time application manipulation. For the SOA Suite, some of the interesting DT@RT bits include editing of business rules; manipulation of domain value maps; configuring properties on service composite applications and adapters; and creating, removing, or changing subscriptions to business events. The Oracle Enterprise Manager Fusion Middleware Control Console, the BPM Process Browser, and the SOA Composer are the tools for most of the SOA Suite’s DT@RT, as is the OSB Console.

Related Suites and Products in FMW 11g

Applications running in the SOA Suite or organizations working with the SOA Suite will frequently use other Fusion Middleware products as well. Some of the most likely suspects that you may run into or decide to use alongside the SOA Suite are detailed in this section.

The Application Development Framework (ADF) is a JSF-based framework for developing rich Java web applications. ADF has special integration points with the SOA Suite. To name a few: ADF is used to create the user interface for the human tasks, ADF Business Components (ADF BC) are used to publish SDO services on top of the database (which are used in, for example, BPEL processes), ADF BC is capable of publishing events onto the Event Delivery Network, and ADF applications can consume the Web Services exposed by composite service applications running in the SOA Suite. By the way, ADF was also used by the Oracle development teams to create the Enterprise Manager Fusion Middleware Control Console.

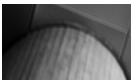
WebCenter has several faces, one of which is its portal capability. Portlets can be seen as a special type of service: a service that has a user interface built into it. WebCenter provides ADF applications with the ability to consume such services, and it also enables ADF developers to expose their applications as Portlet services. WebCenter is also an extension of ADF as a foundation for rich web applications, through a large collection of services that add “integrated collaboration” to ADF applications. This includes support for blogs, wikis, RSS feeds, chat, e-mail, tagging and linking, content integration, management of tasks, activities and events, and enterprise search across all content, services, and application data. WebCenter also adds *design-time at run-time* capability to an ADF application. This enables an application administrator or content editor to change the appearance and content of application components at run time—very much like regular portal products do, but in a more advanced and better integrated-with-ADF way.

Fusion Middleware contains a number of products for governance of SOA artifacts (and other IT assets). At the core of governance is the Enterprise Repository. The Enterprise Repository provides metadata management for technical and software-related SOA assets and sophisticated tools for governing those assets throughout their lifecycle to promote reuse. The Service Registry provides a standards-based (UDDI) reference for the dynamic discovery and use of services and their associated policies at run time. It contains a subset of the metadata managed within Oracle Enterprise Repository that is useful to the run-time infrastructure for dynamic discovery of services and policies.

Oracle BPA is a tool for business analysts and architects to perform process modeling and analysis as well as simulation and publishing of process models. It integrates with both BPM and BPEL: Process models (also called *blueprints*) from BPA serve as the starting point for more detailed, implementation-ready process definitions created in BPM and BPEL.

CEP subscribes to event streams—such as from the SOA Suite Event Delivery Network (although more likely from lower-level and more voluminous sources)—and executes a Continuous Query Language (Oracle CQL) query to search for aggregates, patterns, and exceptions in real-time event streams. The events processed by CEP are usually highly frequent, sometimes physical in nature, and can be quite meaningless by themselves. The results from the continuous queries that reveal a meaningful pattern or an exception are turned into events at a higher, more business-oriented level that can be fed into Oracle BAM or the Event Delivery Network, for example.

Oracle Data Integrator (ODI), together with Oracle GoldenGate, provides a data-integration platform that covers all data-integration requirements—from high-volume, high-performance ELT batches, to real-time, event-driven, trickle-feed integration processes, to SOA-enabled data services. This technology can be used alongside an enterprise service bus to handle large volumes of data that primarily need to be moved from one system to another in not necessarily a service- or XML-oriented way. ODI has support for Web Services as well—both outbound and inbound. In addition, it can be integrated directly with Oracle BAM.



NOTE

ODI intentionally uses the term ELT instead of the more common ETL (Extract, Transform, and Load).

Application Integration Architecture (AIA) is a framework for integrating various products and modules from the Oracle Applications portfolio and for creating cross-module business processes. AIA builds on top of the SOA Suite. Through AIA, JD Edwards, Retek, PeopleSoft, Siebel, EBS, and Fusion Applications—among others—can interact in a loosely coupled way. AIA provides a reference architecture for implementing SOA that can also be used with custom applications and third-party software.

The Oracle Identity and Access Management offering has a substantial number of products that help implement and manage scalable security based on open standards for applications and services. Through Oracle Platform Security Services (OPSS)—an abstraction layer that implements a security and identity and access management API—applications can use a uniform set of services, without having to deal with implementation details of the underlying security infrastructure. OPSS is the platform that provides security to Oracle Fusion Middleware, including products such as WebLogic Server, SOA Suite, WebCenter, ADF, and Oracle Entitlements Server, to name a few. Note that OPSS can run on various JEE application servers, including JBoss and WebSphere, in

addition to WebLogic. SOA Suite and WebLogic Server interact with OPSS for securing Web Services and composite applications.

The Universal Content Manager, with supporting tools such as SiteBuilder, is another component of Fusion Middleware. This product will typically not have direct interaction with applications running in the SOA Suite. The same applies to the FMW products for Business Intelligence (BI EE) and Enterprise Corporate Performance (Hyperion).

Oracle has bundled many of its FMW products in suites, such as the SOA Suite. These suites comprise a logically related collection of products sold under a single license. Having said that, most suites can be bought with tailor-made licenses that apply to a subset of the products in a suite. And besides, there is a lot of overlap between the suites that is accounted for when you acquire more than one of them. The most relevant product suites around the SOA Suite are the BPM Suite, EDA Suite, SOA Governance Suite, BPA Suite, and Data Integration Suite.

Getting Started with SOA Suite 11g

This book is by no means intended to be only theoretical. Yes, it does tell a story and hopefully explains a great deal about the SOA Suite by showing examples and describing the concepts behind and workings of the service engines, underlying standards, and technologies and supporting tools. However, you—and your fingers—will only start to learn for real once you start practicing what is preached in this book. So now is the time to get into gear. It's time to get yourself a fully operational SOA Suite (version 11g) so you can start developing, deploying, and running composite service applications as well as fully appreciate what it is really like to create a service-oriented application.

This section describes the steps to take for installing and configuring SOA Suite 11g. It only provides high-level instructions, however, because the details can be found in several excellent installation manuals. The book's wiki provides an online chapter complement with an extensive installation instruction using many screenshots. The complement guides you through the installation of a complete SOA Suite 11g environment that goes with the examples in this book.

It is assumed that you will install into a development environment to start dabbling with the SOA Suite—not a full-blown, highly available, clustered production environment.

Installation of SOA Suite 11g

The SOA Suite is available for various operating systems, including Windows, Unix, and Linux. You can install and run the SOA Suite on a machine that has at least 2GB of memory, some 10GB of free disk space, and a dual-core 1.5 GHz processor. If you intend to run JDeveloper on the same machine, you should have at least 3GB of memory.

Part of the SOA Suite infrastructure is the metadata repository that needs to be installed in a 10g or 11g Oracle RDBMS. Note that SQL Server 2005 and 2008 are also supported, as may be other databases at some point. Before you start the installation, you need to make sure you have access to such a database—with 1GB of disk space for the creation of new tablespaces. The database parameters for `processes` and `open_cursors` should be set to a value of 500 or above. You need database user credentials with DBA or SYSDBA privileges.

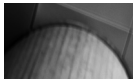
The documentation for the installation (as well as all other details) of the SOA Suite and the other components in Fusion Middleware can be found online at <http://www.oracle.com/technology/documentation/middleware.html>.

Downloading the Software

Before you can start with the installation, you need to download the required software from Oracle Technology Network. Go to the OTN page for Fusion Middleware Software: <http://www.oracle.com/technology/software/products/middleware/index.html>. Download the following components:

- **Repository Creation Utility** 300MB
- **WebLogic Server 11g** 800MB
- **SOA Suite 11g** 1.5GB
- **JDeveloper 11g** 1MB
 - JDeveloper extensions for SOA and BPM - 450Mb
 - Oracle Service Bus 11g - 900Mb (optional)
- Oracle Complex Event Processing 11g (optional)

The design and run-time environment for SOA Suite 11g is illustrated in Figure 3-3.



NOTE

This software (the full production versions) can be used for free under the OTN Development License for self-education or for prototyping and development of applications. All you need is a free account on the Oracle Technology Network.

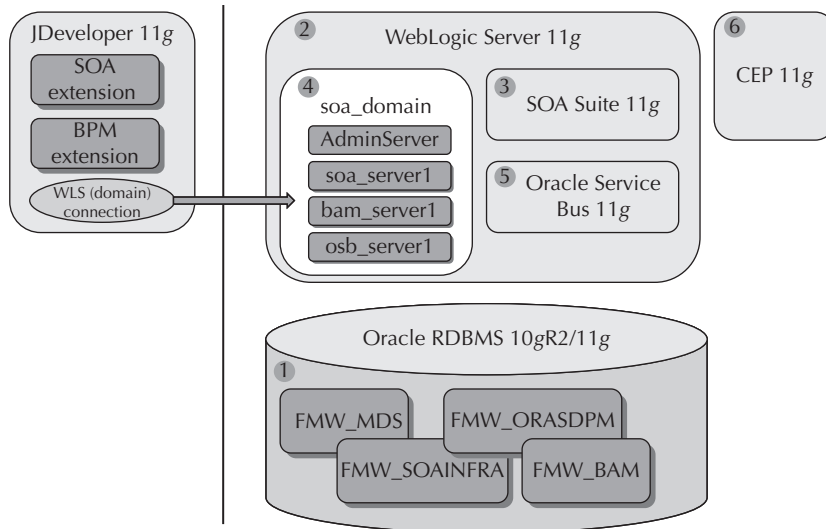


FIGURE 3-3. Installation applies to three tiers: the JDeveloper design time, and the run-time setup of the Database and the Middle Tier

Installation Steps

This section presents the installation steps for the SOA Suite. Further details can be found through references on the the book's wiki.

1. Run the Repository Creation Utility (RCU) to install the metadata repository. The RCU is started from the `rcuHome/bin` directory. The RCU creates all tablespaces, schemas, and database objects required in the metadata repository for SOA Suite and BAM. Select at least the SOA Infrastructure and the Business Activity Monitoring component under SOA and BPM infrastructure. The Metadata Services should be selected automatically upon making those choices.

The default (or minimum) tablespace size can be a bit large for a development environment. You can decrease the size of the data file associated with each tablespace to save on disk space.

2. Install WebLogic Server and create the middleware home. Run the downloaded executable file for WebLogic Server 11g (the internal release number is 10.3.x). Select the option Create A New Middleware Home. Accept the option to perform a typical installation and accept other default values by clicking Next until the Finish button is enabled. Then click the Finish button. Now WebLogic Server will be installed. You could start WebLogic Server when the installation is complete to verify the successful installation. However, let's first install SOA Suite 11g and create the SOA domain inside WebLogic Server.

3. Install the SOA Suite. Extract the downloaded ZIP file to a temporary directory. Run the executable `runInstaller` (Linux and Unix) or `setup.exe` and pass the parameter `-jreLoc`, specifying the location of a Java 6 run-time environment (for example, the one installed along with WebLogic Server in `MIDDLEWARE_HOME\jdk160_11`).

The Installer Wizard appears. It performs a number of checks—available disk space, hardware requirements, and so on. Then it asks for the install location. In the Oracle Middleware Home field, specify the absolute path to your existing Oracle Middleware Home directory; this is the directory created when you installed Oracle WebLogic Server. It presents a summary and allows you to start the actual installation by clicking the Install button. When you click that button, the SOA Suite software is installed in a directory structure starting at `SOA_HOME` that lives under `MIDDLEWARE_HOME`.

4. Configure the SOA Suite. At this point, we have the metadata repository prepared in the database and a clean install of the WebLogic Server. The SOA Suite software has been installed, but not yet configured. There is no SOA container running inside WebLogic Server just yet. The next step entails configuring the SOA Suite inside the WebLogic Server. We do this using the Fusion Middleware Configuration Wizard.

This Configuration Wizard is located in the `SOA_HOME/common/bin` directory (for Linux) or the `SOA_HOME\common\bin` directory (for Windows). Go to this directory; then run the `config.sh` script (for Linux) or the `config.cmd` script (for Windows) to start the Configuration Wizard. Unless you have very specific reasons for deviating from the default settings, you should accept them as they are for this development environment.

The wizard will create a new WebLogic domain called `soa_domain`. You have to provide the credentials for a new user who will have the Administrator role. The default recommended values to accept are *weblogic* for the username and *weblogic1* (the last character is the number one) as password. Note: Another frequently used password is *welcome1*. You may come across it in tutorials and installation instructions or other documentation.)

WebLogic Terminology

The product we just installed is called WebLogic Server 11g. However, at this point we cannot run it because it has no instance to start and run.

An installation of WebLogic Server can be used to run one or more domains. For the installation of the SOA Suite and the BAM server in the personal development environment, we will assume for this book that you will work within a single domain. A domain is a logically related group of servers that can share certain resources. A server is a unit that can be started and stopped independently of other servers. Servers host the components and associated resources that constitute your applications—for example, JSF pages and EJBs. Every domain contains a special server: the Administration Server (AdminServer). You use the Administration Server, programmatically or through the Administration Console or WLST, to configure all other server instances and resources in the domain.

All other servers in the domain are called managed servers. When a managed server starts up, it connects to the domain's Administration Server to obtain configuration and deployment settings. However, a managed server can start up independently of the Administration Server if the Administration Server is unavailable. Several managed servers can be linked to form a cluster. Note that a cluster runs within a single domain.

The installation of the SOA Suite that is described next involves an AdminServer, a server for the core SOA Suite components—usually called `soa_server1`—and a second, managed server that runs the Oracle BAM Server and web applications; this server is called `bam_server1` by default.

Specify database connection details for the metadata repository you created using the Repository Creation Utility in step 1. On the Summary page, click Install to start the creation of the new SOA domain with three servers inside: AdminServer, `soa_server1`, and `bam_server1`. Note that you can run this Configuration Wizard at a later moment to apply additional configurations to this domain.

The new domain is created in the directory `MIDDLEWARE_HOME/user_projects/domains/soa_domain`.

5. Install the Oracle Service Bus 11g. (This step can be considered optional at this point, as the OSB is only required for Chapter 13.) Run the `setup.exe` from the OSB 11g download. Provide the JRE location. Install the OSB 11g into the same Middleware Home used for the SOA Suite 11g. Next, run the configuration wizard to extend the `soa_domain` created during the installation of the SOA Suite 11g with a managed server `osb_server1` that contains OSB 11g.

6. Install Complex Event Processing 11g. Again, an optional step, Complex Event Processing is only introduced in Chapter 19. The CEP installation process creates a separate lightweight container that runs CEP. Additionally, Eclipse should be installed as the development environment, with the CEP plug-ins to complete the CEP IDE.

7. Start the AdminServer and the managed servers for SOA and BAM. Before we can start doing anything at all with the SOA Suite, we need to start the servers in the new WebLogic SOA domain. We will use three command-line scripts to get these servers going.

Starting the SOA Server without Entering the Credentials

If you do not want to have to type in the username and password of the administrator account every time you start up the server, you can do the following—after having started the server at least once to have the directory structure created:

1. Create a new directory called “security” under `MIDDLEWARE_HOME \user_projects\domains\soa_domain\servers\soa_server1`.
2. Create a new text file called `boot.properties` in this directory.
3. Add two lines to this file with username and password key-value pairs, like this:

```
username=weblogic
password=weblogic1
```

When the server is started, the credentials are read from this file. Note that the file will be changed into a more secure, encrypted pair of values. The same instructions apply to the BAM server and the OSB server.

First, to start the AdminServer, locate the `startWebLogic.cmd` script (Linux: `startWebLogic.sh`) in the `MIDDLEWARE_HOME \user_projects\domains\soa_domain` directory. Run this script from the command line.

When the AdminServer is running, you should start the SOA server and optionally the BAM server and/or the OSB server. Go to the directory `MIDDLEWARE_HOME \user_projects\domains\soa_domain\bin`, which was created by the SOA Suite Configuration Wizard. Open a command window and enter the following command to execute:

```
startManagedWebLogic.cmd soa_server1
```

(On Linux, use the script `startManagedWebLogic.sh`.)

To start the BAM server, use this same command for the `bam_server1`, and for the OSB server replace `soa_server1` with `osb_server1`.

NOTE

You will be prompted for the username (weblogic) and password (weblogic1) of the administrator credentials used to boot the server, so do not go away right after starting the script.

8. Access the Oracle Enterprise Manager Fusion Middleware Control. With the servers running, now is a good moment to check out the Enterprise Manager. This console is where most of the administration tasks take place with regard to the SOA Suite and the composite applications. The Enterprise Manager Fusion Middleware Control, shown in Figure 3-4, supports various actions, such as deploying, starting and stopping, and testing composite applications, as well as inspecting completed and running instances of the applications, including fine-grained details from individual service engines and every single step in BPEL process execution. The console is available at `http://localhost:7001/em`. Use the username `weblogic` and the password `weblogic1` to log in to the console.

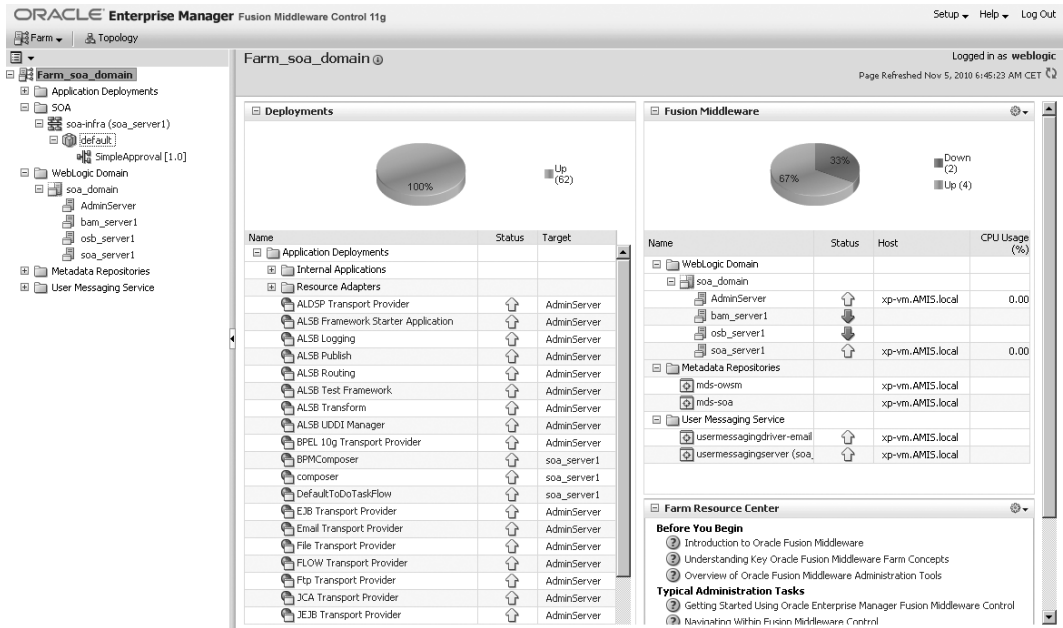


FIGURE 3-4. The Enterprise Manager Fusion Middleware Control

The SOA node is the starting point from which you can inspect the list of deployed composite applications—although, of course, initially there are none.

The classic WebLogic Server Administration console is also available and can be accessed at <http://localhost:7001/console>. You will need this console to create and edit JDBC data sources and JMS queues, as well as to configure various technology adapters and the identity store used by WebLogic Server and SOA Suite.

Other web applications running at this point include the following:

- The BPM Worklist application, which displays the tasks assigned to users by the Human Workflow service. This application can be accessed at <http://localhost:8001/integration/worklistapp>.
- Running at <http://localhost:8001/soa/composer> is the SOA Composer, an application that supports live editing of Business Rules and Domain Value Maps.
- BPM Process Composer is the application where business analysts and business process developers meet up to create and edit Business Process Models; it is available at <http://localhost:8001/bpm/composer>.
- When the OSB server is started, the console—both for development as well as for administration—can be accessed at <http://localhost:7001/sbconsole>.
- When the BAM server is started, the BAM web applications are available from the start page at <http://localhost:9001/OracleBAM>.

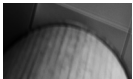
9. Take further configuration steps. The installation and configuration of the SOA Suite is now done to the point where we can start deploying and running the composite applications. There are some additional configuration tasks you may want to perform, either now or at a later moment. You could, for example, configure JMS queues or JDBC data sources in the WebLogic Server Administration Console.

This may also be a good moment to configure the User Messaging Service (UMS) to allow the composite applications to send and receive notifications via e-mail, IM, SMS, and other communication channels.

Details for the configuration of JMS, JDBC, and UMS can be found online in Appendix C.

Installing JDeveloper

Go to the JDeveloper Downloads page on OTN (<http://www.oracle.com/technology/software/products/jdev/htdocs/soft11.html>) and download the latest JDeveloper Studio Edition release. After downloading the executable EXE, BIN, or JAR file, run the file to start the installation. Accept the default settings in the Installation Wizard and have both JDeveloper and an integrated WebLogic server installed.



NOTE

We will not use the integrated WebLogic Server for running the SOA composite applications because it does not have the SOA server installed.

Adding JDeveloper extensions When the installation is done, start JDeveloper. Before you can design SOA applications, you need to install the SOA Suite extension—and for BPMN components the BPM Studio extension. Select the option Check For Updates from the Help menu. The Check For Updates Wizard comes up. Make sure that the box for Oracle Fusion Middleware Products is checked on the Source page. The list of available updates will include the latest Oracle SOA Suite Composite Editor 11g plug-in as well as the BPM Studio extension. Check both boxes and click Finish. You will now probably have to provide your OTN credentials. JDeveloper will then start to download the plug-ins (some 450MB in total). After the download completes, you need to restart JDeveloper in order to activate the two plug-ins.

You can also add the SOA Composite Editor and BPM Studio extensions to JDeveloper from local files that you download from the Oracle Fusion Middleware Products Update Center (http://www.oracle.com/technology/products/jdev/101/update/fmw_products.xml). This can be useful when you have to install the same version of the plug-ins on several clients or when the download from JDeveloper fails for some reason (possibly firewall related). Just download the ZIP files for the SOA Composite Editor and the BPM Studio. Then run the Check For Updates Wizard, and on the Source page select the radio button Install From Local File. Select the downloaded ZIP file to install the extension from.

Create a connection to the SOA Suite To deploy SOA applications directly onto the SOA Suite, we need to configure an application server connection in JDeveloper to the WebLogic Server `soa_domain`.

To create this connection, start JDeveloper and go to the Resource Palette. Click the New icon, select New Connection, and pick the Application Server connection type from the list. Enter **FMW11g_SOASuite11g_local** as the name for the new application server connection and select WebLogic 10.3 as the connection type. Click the Next button. Provide the authentication details:

Again, use the `weblogic/weblogic1` username/password combination. Next, you need to indicate the WebLogic hostname (`localhost` for local installations) and the port for the AdminServer (7001 by default). Also enter the name for the domain you connect to; the name suggested in the on line installation instructions is `soa_domain`. Click the Next button again. When you test the connection, you should receive a number of success messages, one for each of the different ways of connecting to the domain.

Click the Finish button to close the Create Connection dialog. The new connection is now available on the Resource Palette.

See the on line chapter complement for detailed screenshots.

Sample Application and Fusion Order Demo

Oracle provides a demo application—called the Fusion Order Demo (FOD)—as a showcase for Fusion Middleware applications. This demo application is an end-to-end application example developed by the Fusion Middleware Product Management. It demonstrates common use-cases in Fusion Middleware applications, especially the integration between ADF, SOA composite applications, and WebCenter, as well as the usage of various service engines and adaptors inside SOA applications. The business scenario demonstrated in FOD is a web shop where customers can order products. Every new order triggers various process flows that handle the approval, logistics, and payment details.

You will find the Fusion Order Demo on OTN (<http://www.oracle.com/technology/products/jdev/samples/fod/index.html>). Installation and configuration instructions are provided on this page.



NOTE

To see every aspect of this demo in action, you would have to update JDeveloper with the WebCenter plug-in. However, for inspecting the SOA Composite applications in the demo, this is not required.

Create and Run the “HelloWorld” of Service Composite Applications

This section walks you through the steps to create, deploy, and test-run (with SOA Suite 11g) the world’s most basic SOA composite application. At the end of this section—after maybe ten minutes’ worth of work—you will have your first application running in the SOA Suite. (Note that detailed, step-by-step screenshots for this section are available in the on line chapter complement on the book’s wiki.) The steps are as follows:

1. Fire up the engines. First, start the database that hosts the metadata repository and then the WebLogic servers in the SOA domain (AdminServer and the managed `soa_server1`) using the command-line scripts.

Locate the `startWebLogic.cmd` script (Linux: `startWebLogic.sh`) in the `MIDDLEWARE_HOME \user_projects\domains\soa_domain` directory. Run this script from the command line or terminal.

When the AdminServer is running, you should start the SOA server. Go to the directory `MIDDLEWARE_HOME \user_projects\domains\soa_domain\bin`, which was created by the SOA Suite Configuration Wizard. Open a command window and enter the following command to execute:

```
startManagedWebLogic.cmd soa_server1
```

(On Linux, use the script `startManagedWebLogic.sh`.)

2. Start JDeveloper. Be sure to choose Default Role if you are prompted to select a role.

3. Select New from the File menu. From the New Gallery that is presented next, select the SOA Application item in the Applications Category (under the General node). Click the OK button to continue.

You will be prompted to provide a name for the application—for example, HelloWorldSOAComposite. Leave the Application Package Prefix field empty and click the Next button. On the next page, enter **HelloWorld** as the name of the project and click Next again. JDeveloper then asks you what type of composite application this will be; pick Composite with BPEL on the Configure SOA settings step. Click Finish to have the application, project, and service composite created.

4. The Create BPEL Process dialog appears. Specify the name for the new BPEL process (HelloWorld) and the template (Synchronous BPEL Process). Make sure the box Expose As A SOAP Service is checked, and accept the defaults for Namespace, Service Name, and Input and Output (variables). Click OK. The Create BPEL Process dialog is shown in Figure 3-5.

5. The BPEL editor opens up. You will see the basic structure of the BPEL process with a Receive activity and a Reply activity, by default configured to receive a single string and return a single string. You need to add one activity to set the value of that string result: Drag an Assign

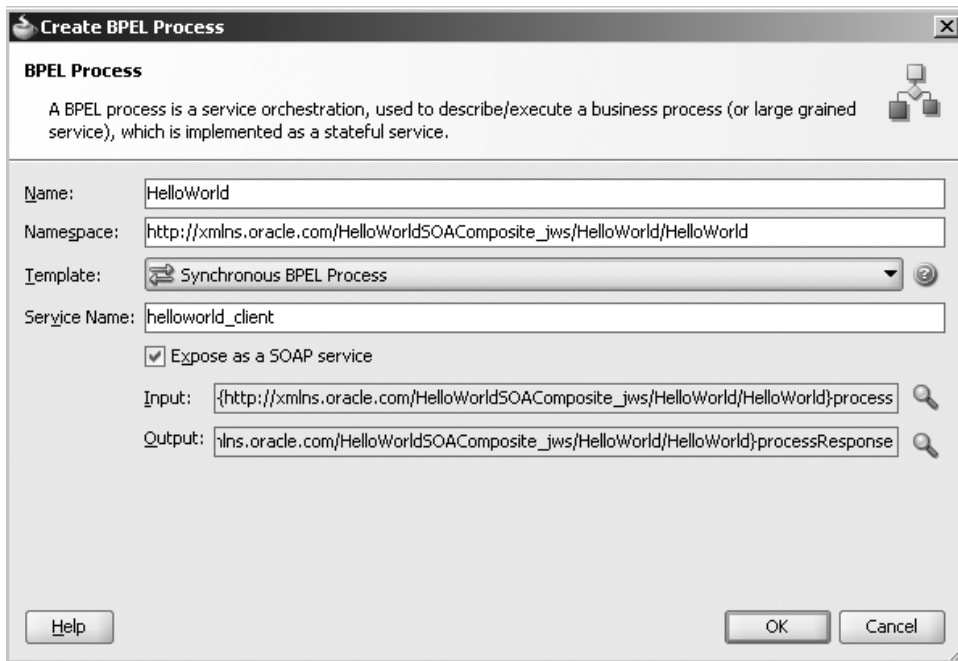


FIGURE 3-5. Configure the new HelloWorld BPEL process.

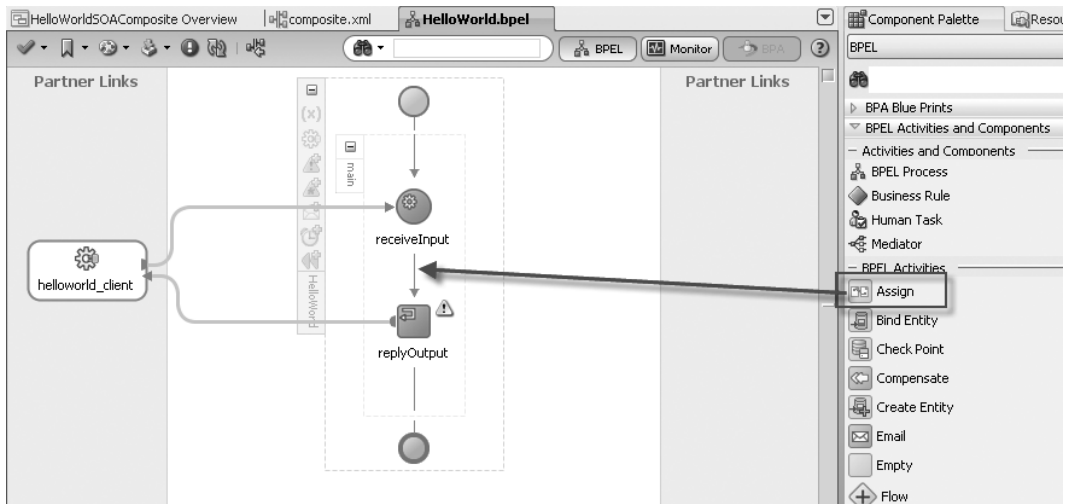


FIGURE 3-6. Add an Assign activity to the BPEL process.

activity from the Component Palette and drop it between the Receive and Reply activities already in the process, as shown in Figure 3-6.

Double-click the Assign activity to open the editor. Select the second tab, labeled Copy Operation, if it is not already selected. Click the green plus sign and select Copy Operation from the drop-down list. On the right (or To) side of the window, expand the outputVariable node, the payload child node, and its client:processResponse node, and then select the client:result node. That is the target of the Copy operation.

On the left (or From) side of the window, choose Expression in the drop-down list. Click the calculator icon to open the XPath expression editor. Type the following text in the expression box:

```
Concat('Hello dear',)
```

Position the cursor between the comma and the closing parenthesis. Expand the inputVariable in the BPEL Variables tree, all the way down until you can select the node client:input. Select that node. Click the Insert Into Expression button to add [an expression to extract] the value of the input variable to the expression. Click the OK button to close the expression editor. Click OK again to close the Create Copy Operation dialog and then one more time to close the Assign Editor. Figure 3-7 shows the creation of the copy operation in the Assign activity.

You have now created a valid BPEL process—one that receives a request message that contains a single string and returns a message that will contain the concatenation of “Hello dear” with that same input string. It’s not much, but it constitutes a real BPEL process inside the HelloWorldSOAComposite application.

6. To test this application, deploy it first. Right-click the HelloWorld project. From the context menu, select Deploy and its nested option, HelloWorld.

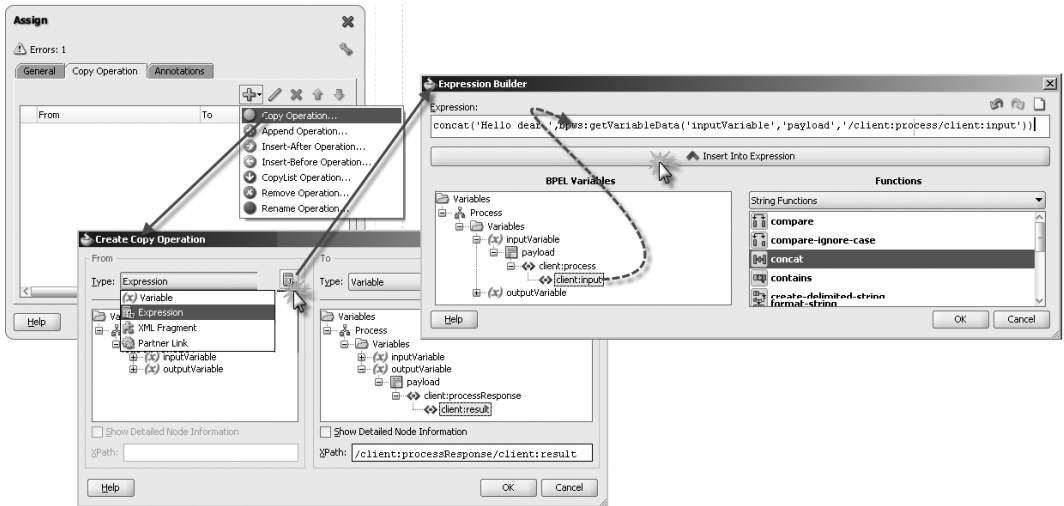


FIGURE 3-7. Configuring the Assign activity

The Deployment Wizard appears, which is a multistep dialog. In the first step, select Deploy To Application Server (instead of deploying to an SAR file). Click Next. Accept all the default settings in the second step, and click Next. On the third page, select the FMW11g_SOASuite11g_local connection to the WebLogic server with the SOA domain. Click Next. On the next page, select soa_server1 as the target server for deployment. Click Next, and the Summary page appears. Now you can click Finish.

The SOA composite application is built, resulting in an SAR (Service Archive) file. The archive is now handed to soa_server1 in the SOA domain on the WebLogic server. The message “Deployment Finished” should appear in the Deployment sub tab of the console window after several seconds (up to one minute).

7. Deployment is complete. With the deployment done, we can access the composite application’s Web Service interface from tools such as the HttpAnalyzer inside JDeveloper or soapUI (an open-source tool frequently used for testing Web Services). We can also open the Enterprise Manager to first inspect the deployed composite application and then test it.

Open the Enterprise Manager (<http://localhost:7001/em>). Expand the SOA node and its child, the soa-infra node, under the root node Farm_soa_domain. The node for the HelloWorldSOAComposite application should be listed. Select that node.

The right side of the page is refreshed to present the details for this composite application. Click the Test button to call the service exposed by this composite application. Enter a value for the input field—for example, your own first name—and click the button labeled Test WebService. The Web Service exposed by the HelloWorld application is invoked. This will create a new instance of the composite application. After a few seconds, the result from the service should be displayed, something to the effect of “Hello dear Lucas.” Figure 3-8 demonstrates the test run of the HelloWorld application’s service.

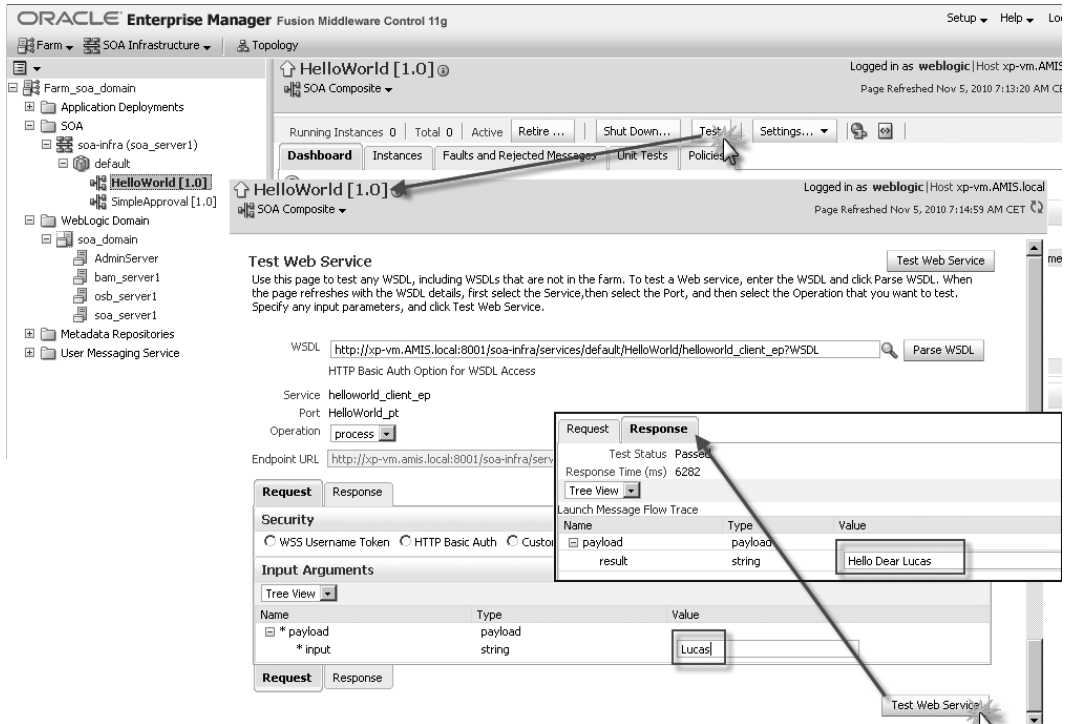


FIGURE 3-8. Running the SOA composite application *HelloWorld* in the Enterprise Manager Fusion Middleware Control

When you click the node for the *HelloWorldSOAComposite* application, you will see the new instance listed. You can drill down on this instance to find out more about the components in this instance that have executed (how long they took to complete, for example) and all trace details for activity inside those components (such as the activities in the BPEL process). You need to click the instance ID itself to see more information (not just on the row where the instance information is shown).

At this stage, you have achieved quite a bit. The SOA Suite 11g run-time environment has been installed and configured. Also, it actually works because it was possible to deploy and run a very simple composite application on it (whatever exactly that may be). The design time (JDeveloper with SOA Suite extension) is also up and running, appropriately configured with a connection to the SOA Suite container.

Migrating from SOA Suite 10.1.3

Many organizations have adopted Oracle SOA Suite 10.1.3 in the recent past, using BPEL Process Manager, the enterprise service bus, and/or Web Services Manager. Some even started with the 10.1.2 release. Such organizations typically have made considerable investments in

their environment, the SOA applications, and the skills required to develop the applications and administer the infrastructure.

With SOA Suite 11g, these organizations may feel like they are up against the “dialectics of progress”: They were the first to adopt Oracle’s SOA offerings and as a result they now have to make additional investments to upgrade to this latest release. However, much of the investment is not lost, but instead can simply be applied to SOA Suite 11g. And these early adopters are best equipped to appreciate many of the improvements available in 11g over the previous releases of the SOA Suite. Finally, Oracle has provided various tools that support the migration. As a result, it may not be as earth-shattering, risky, or costly as it appears from a distance.

Note that there is no supported migration path from SOA Suite 10.1.2 to 11g; you will have to perform an upgrade from 10.1.2 to 10.1.3 first.

The migration to SOA Suite 11g involves several aspects:

- The environment or run-time infrastructure (from OC4J to WebLogic Server)
- The development tools (JDeveloper)
- The security framework and identity and access management tools
- The SOA applications developed on 10.1.3
- Any long-running BPEL processes with open instances
- Client applications that hook into the SOA Suite via (Java) APIs
- The skills, processes, standards and guidelines, and best practices

You can find more on migration from Oracle SOA Suite 10g to release 11g in Appendix A.

Summary

Oracle SOA Suite 11g did not appear out of thin air. It is the next step in a long evolution in the IT industry at large (and Oracle Corporation in particular).

This chapter gave you a glimpse of the rise of software for integration and later on middleware, in general, within Oracle. We discussed the advent of industry standards, starting with XML and encompassing the Web Services standards, and more recently the standards for business processes and service components. These standards are essential to success of Web Services—the foundation for interoperability—and Service-Oriented Architecture. Oracle plays an important role in the specification process and the promotion of most industry standards.

Oracle itself is an interesting example of integration: The company has acquired and subsequently absorbed several dozens of other companies and their software offerings. Many important parts of today’s software portfolio have roots in products from these “scalps.” The most striking acquisitions in the area of middleware are Collaxa (2004), BEA (2008), and Sun Microsystems and AmberPoint (both in 2010).

A many-year process of innovation, integration, and interaction with customers, including the important internal Applications Development teams, has finally resulted in Fusion Middleware. On July 1, 2009, Fusion Middleware 11g was launched. FMW offers a wide palette of middleware technology, ranging from business intelligence, Web Services, and content management, to enterprise collaboration, identity and access management, governance, event processing, and custom-developed user interfaces. SOA Suite 11g is an important element in the FMW 11g stack, with interactions with many of the other areas within FMW.

The SOA Suite has at its heart the SCA container that runs SOA composite applications. These applications are built from components that run on specialized engines: BPEL, Mediator, BPMN, Business Rules, (Spring) Java, and Human Task. The components can interact with external Web Services and technology adapters to reach out to the database, file system, messaging infrastructures, and so on. The SOA Suite provides a framework for negotiating events between applications, offering a very decoupled way of making different applications interact. Other products in the SOA Suite are Oracle Service Bus, Business Activity Monitoring (BAM), and Complex Event Processing (CEP).

Organizations that have adopted earlier generations of the SOA Suite will have to go through a migration process when they want to take up the 11g release. This migration applies to several aspects, including the infrastructure, applications, other software assets, and the skills of staff such as developers and administrators.

This concludes Part I of the book. The next part introduces the components of the SOA Suite in detail and demonstrates how to create composite applications with them.