



The ABCs of AIA

Oracle's Application Integration Architecture

Oracle's Application Integration Architecture (AIA) is a revolutionary new approach to systems integration. Or is it? This paper will explore how AIA fits into the continuum of system integration methods and will explain, as non-technically as possible, how it really may be the groundbreaking approach to integration for which organizations have been waiting. By detailing the basic concepts, components, and functions of this architecture, we will demonstrate how it can provide the foundation necessary to support an evolution to an efficient, flexible, scalable, and highly reliable integration platform for your Oracle applications.

Moving Beyond Point-to-Point Integration

The challenge of system integration has plagued organizations since the development of the first computer-based applications many years ago. Over time, we have seen many “revolutionary new ways” to improve system integration capabilities. Yet, for all of the progress that has been made in technology, many organizations remain surprisingly dependent on traditional methods to achieve “integration.” For example, how many of the following “integration methods” are still part of your business processes?

- Manual re-keying of information from a report into another system
- Hard-coded extract of information from System A to populate interface table of System B
- Hard-coded point-to-point interface/inquiry from a legacy system to Oracle E-Business Suite
- Export of flat file to Microsoft Excel
- Upload of an Excel file into a batch interface
- Export of flat file and ftp to a third-party system
- Export of flat file for upload into a legacy batch interface
- Batch load of a flat file received via e-mail
- Duplication of master data shared between legacy systems due to disparate system environments

Despite the awareness of better, more efficient approaches to support system integration that have presented themselves over the years, most organizations today still employ a costly point-to-point integration infrastructure within their Oracle

E-Business Suite environment. There are several reasons this practice is so prevalent:

- The age-old battle between doing it now versus doing it right
- The organization has always used point-to-point solutions and hasn’t thought about alternatives
- The IT staff is familiar with the existing systems and can implement point-to-point integration more quickly (by copying and modifying similar code, thus multiplying maintenance requirements)
- The infrastructure and discipline necessary to support more efficient, reliable, and re-usable integration methods does not exist within the organization
- The new technologies involved in these different integration methods are unfamiliar to the IT staff

The result of these point-to-point integration scenarios is illustrated by Figure A:

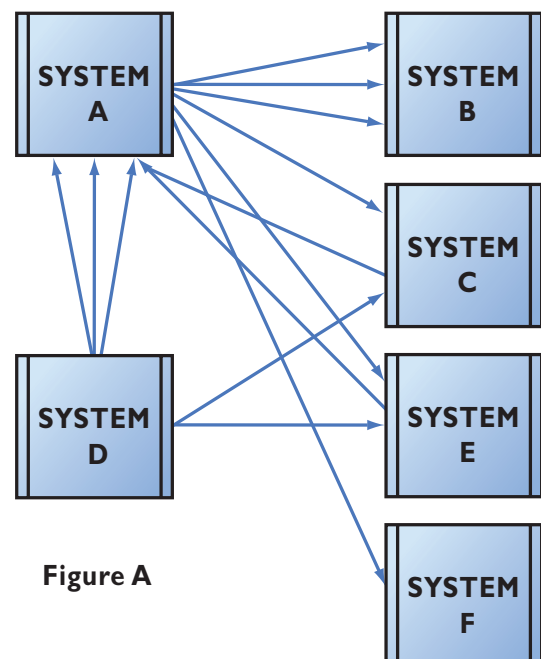


Figure A

In this type of an environment, point-to-point interfaces have been built from system to system, and each one is most likely tightly coupled with the source and target systems. Business rules and logic are embedded in the individual components. Any change to a core system (e.g. replacing System A) would require significant re-development and regression testing of the interfaces, as well as downstream impacts on other systems. Many organizations reach the point where replacing System A becomes too risky, complicated, and costly. The end result is a non-agile, complex web of legacy systems that perpetuates inflexible, outdated, and inefficient business processes. Does this scenario sound familiar?

Organizations that find themselves in this situation often struggle with how to evolve. Is it even possible to change this environment without a radical, high-risk, big-bang system re-development effort? Wouldn't it be great if there was a comprehensive integration solution that would support a low-risk evolution into an advanced service-oriented architecture?

The goal of this paper is to help organizations understand how Oracle's Application Integration Architecture (AIA) can provide this solution. The concepts of AIA are not necessarily new, but the delivery of such a complete architectural foundation is groundbreaking. Never before have all of the components for success been available in one cohesive infrastructure. Prior to AIA, there has always been something missing... the lack of some key component that was the Achilles heel to a successful implementation of the optimal integration strategy.

Through a clear, non-technical description of the various components of AIA, we hope to show how an organization can benefit significantly from what Oracle AIA can offer. Rather than proposing a big-bang, high-cost initiative, we will propose some simpler ways to start on the journey... to walk before you run, and to leverage what AIA provides to help you get there quickly.

Meeting the Demands of System Integration - Why AIA?

Today's organizations are at a critical juncture. Never before have there been so many demands on our system infrastructure. There are a myriad of packaged software solutions, each bringing "best of breed" functionality to their users. New technology capabilities and usage methods require applications to be upgraded more frequently to take advantage of new functionality, security, and reliability. Rapidly changing business environments place constant pressure on organizations to become increasingly agile to address ever-changing customer, supplier, regulatory, and profitability demands.

How do these demands impact system integration requirements?

- Increased use of "best of breed" applications to meet unique and complicated business processes creates additional integration points
- The increasing volume of packaged applications necessitates more upgrades and system interdependence



- The speed of information is increasing, and the need for real-time analysis puts pressure on slower batch integration methods
- The constant quest for cost reductions requires maximum efficiency for system maintenance and new system deployment
- Complexity of integration scenarios is increasing (eg. workflow orchestrations; real-time error handling that crosses application boundaries)
- There is greater need for master data management

The bottom line is that business complexity and competition is increasing at a time when there is unprecedented demand on systems to be more agile and cost effective. Traditional integration strategies have not proven capable of keeping up with these demands in the past, and they will fail more visibly in the future. At the same time, organizations are under intense pressure to reduce the business risks introduced by upgrading or replacing existing systems or adding new applications due to acquisitions or other business needs.

In order to meet the demands of system integration in the future, organizations must implement a strategy that addresses the following objectives:

- Maximize re-use of business technology components to minimize maintenance and development costs
- Support the ability to use/swap-out applications to take advantage of “best-of-breed” with the least risk to the applications environment
- Create an environment that minimizes the cost of regression testing in support of upstream and downstream application changes
- Create an agile environment that can rapidly adapt to changing business and regulatory requirements

It may seem an impossible list of objectives, but it is achievable with the following approach:

- Establish a “loosely-coupled” application integration approach
- Adopt a strategy that allows the organization to evolve away from “tightly-coupled” point-to-point integrations and that doesn’t require a big-bang replacement effort
- Establish and leverage cross-industry standards where possible
- Build the integrations on an architecture that is application neutral
- Re-use integration components where possible
- Adopt an integration strategy that can address packaged software, custom legacy systems, as well as third-party applications
- Build a framework of re-usable features that are common to all integrations such as security and error handling
- Incorporate a robust testing scenario capability to ensure high-quality deployments
- Take advantage of off-the-shelf, pre-built integrations where possible

Oracle’s Application Integration Architecture can be used as the foundation of a strategy to address all of these goals.

The best way to understand the benefits of AIA is to start with a simple example of a typical point-to-point integration, and build conceptually on that model. Traditional methods of integration can be simply illustrated in Figure B, which depicts how a requesting application (e.g. a custom Order Management system) might be integrated with a responding application (e.g. a Receivables system) to provide information about a customer balance during an order approval process.



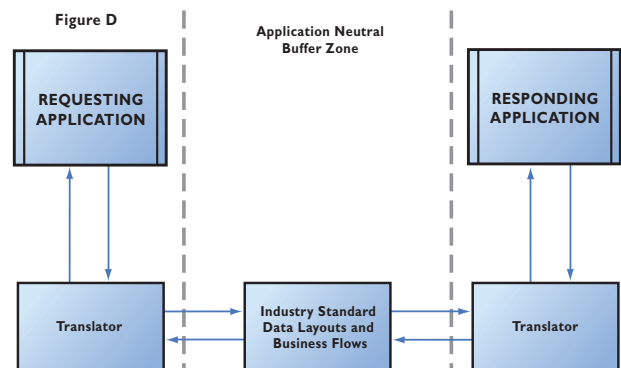
Figure B

Figure B typifies how most integrations are implemented today. Each integration process (the request and the response) must know the details of the other system’s data definitions, processing logic, and interface formats.

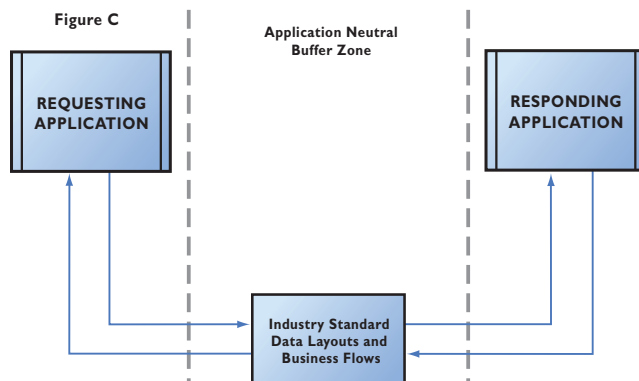
Any change to either system will likely require modification to both of the integration processes. In this sense, we refer to this integration as “tightly coupled” as the integrations are completely dependent on both systems and all processing logic is built in to the integration. This type of integration approach leads to the complex and unwieldy environments found in most organizations.

The key to achieving a more robust, flexible, and scalable architecture is to move away from these tightly-coupled integrations to an architecture built on

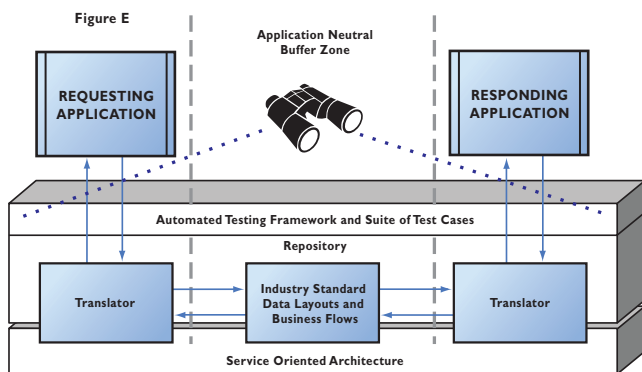
a “loosely-coupled” integration strategy. The evolution away from this tightly-coupled approach starts with an architecture that establishes an application-neutral buffer zone between applications, and the utilization of industry standard data layouts and business flows. Figure C depicts the first step in this evolution:



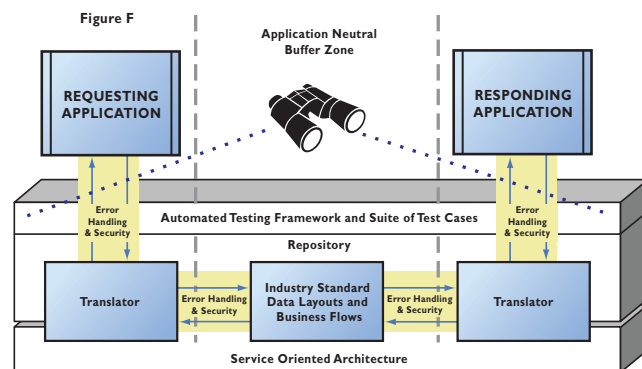
In order for any application to interact with a set of industry standard data and process models, they have to use a common vocabulary. To maintain this loosely-coupled architecture, the data definitions known by a requesting or responding system must be translated to the standard definitions established by the architecture framework. This translation is performed by application-specific translators placed on each side of the buffer zone. These translators convert data from their respective applications to the standards established by the framework and vice-versa. Figure D illustrates where these translators fit into the picture:



One by-product of a loosely-coupled architecture is a proliferation of objects to manage and the need to establish the technology and development standards capable of tying everything together. To address these issues, the many objects and processes created need to be managed within a repository that supports visibility across the architecture and supports efficient re-use and extension of components. An automated testing framework is critical to ensuring that changes to any of the objects within the repository can be tested adequately prior to deployment. Additionally, the integrations need to operate on a flexible, application-independent platform. A Service-Oriented Architecture (SOA) fulfills that requirement. Figure E shows how these additional functions support these needs.



The last components missing from this architecture are integrated error-handling and security capabilities. Figure F shows the entire architecture concept with all components in place.



This loosely-coupled integration strategy makes intuitive sense, so why has it been so difficult for organizations to achieve in the past? The concepts really aren't new... it's just that implementing them seemed like a daunting task, and there was no established path to follow. Many early efforts failed. Without a standard foundation to build on, any effort to achieve this loosely-coupled integration architecture has been difficult to justify from an ROI standpoint.

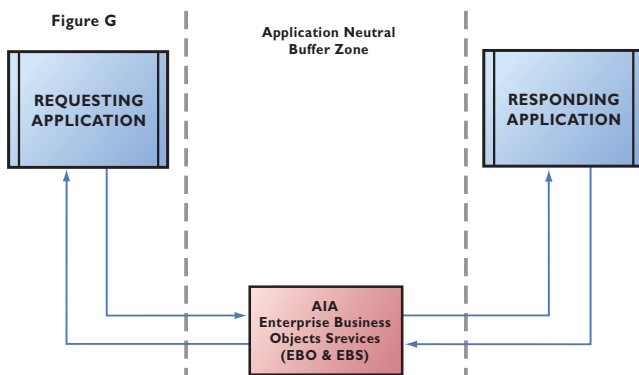
Herein lies the true value of AIA. With AIA, the foundation is now built, and all of the capabilities described above are available today. Finally, incremental progress towards a loosely-coupled, standards-based integration architecture can be achieved in a very cost-effective manner.

Overview of AIA

AIA is a collection of infrastructure components and tools packaged with methodology guidance for the purpose of creating loosely-coupled, standards-based integrations. AIA is designed as a service-oriented architecture with all of the interoperability features inherent in service-oriented designs. The concepts described above represent major components and capabilities of AIA, but not the specific terminology. AIA is based on several key components that are represented in Figure F, but the components have specific roles within AIA. Successful utilization of AIA starts with a basic understanding of these components and how they relate to each other. To illustrate how these concepts are delivered through AIA, we will progress through the same series of diagrams, explaining each AIA component in the process.

A key foundational requirement to enabling a loosely-coupled architecture is the establishment of standard data and process definitions that can be leveraged

by all systems. One of the significant advantages of AIA is that it delivers this capability through pre-built **Enterprise Business Objects (EBO)** and **Enterprise Business Services (EBS)** based on best-in-class cross-industry standards. These objects and services are application independent and extensible. They provide the basis for the common vocabulary that will be used within the architecture. Figure G below depicts these components:

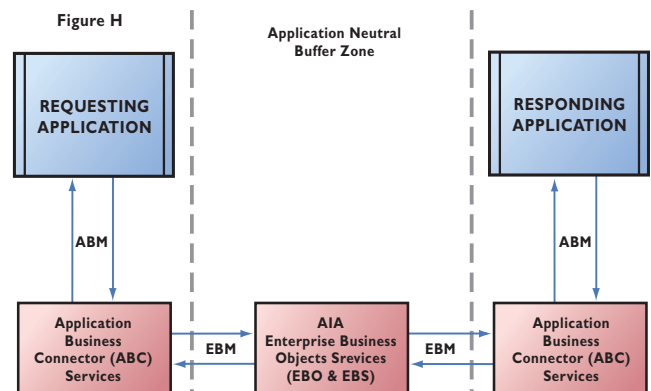


The AIA Enterprise Business Objects & Services provide the standards and abstraction layer necessary to support enterprise-wide integration in a loosely-coupled fashion. Enterprise Business Objects represent common business entities such as Customers, Suppliers, Purchase Orders, Sales Orders, Invoices, etc. Enterprise Business Services encompass a library of standard processes and operations that would typically interact with an EBO such as “create invoice” or “query supplier.” These EBSs comprise a rich library of services based on the standards established by the EBOs. However, these services do not understand the operational vocabulary used by specific applications. Likewise, enterprise applications do not know the vocabulary of the services and objects provided by AIA.

For example, the Supplier Name field in a requesting application will undoubtedly be different than what

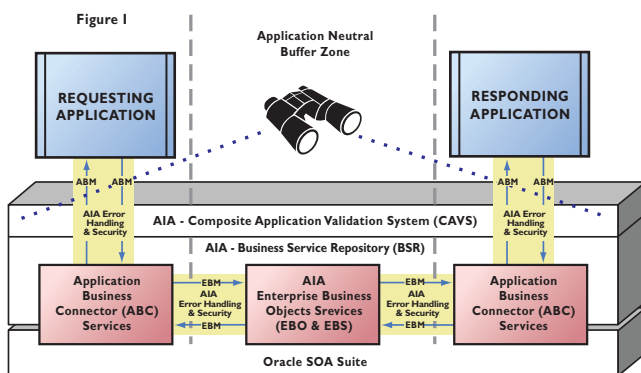
is defined in the Supplier EBO. The Supplier Name in the responding application will also be defined differently. Thus, in order for a requesting application to communicate with a responding application, a translation to a common vocabulary must be performed. This common vocabulary is established by AIA through the Enterprise Business Objects. The translation from application-specific terminology to EBO standards is accomplished in AIA via **Application Business Connector Services (ABCs)**.

The use of ABCs is the key to achieving the loosely-coupled integration sought within the architecture strategy. Requestor and provider applications no longer need to be tightly coupled to communicate with each other. The ABCs communicate with the requesting and responding applications using **Application Business Messages (ABM)** based on the application-specific terminology known and understood by those applications. These messages are translated into standard messages understood by AIA, specifically **Enterprise Business Messages (EBM)** used by EBSs. An EBM will transport all or part of a specific EBO based on the context of the message. Only the information needed will be included in the message, thus keeping message traffic efficient throughout the process. Figure H illustrates these key components within AIA.



AIA is built on a SOA framework and leverages all of the native capabilities of that architecture. As mentioned previously, pursuing a loosely-coupled integration strategy will create more “moving parts” than a tightly-coupled approach, due to the abstraction of objects and the re-use of standard components. AIA provides a Business Services Repository (BSR) to support effective management of the library of objects and services created for an implementation. The BSR allows clear visibility of all components making re-use much easier. It also supports powerful features that allow users to perform an impact analysis when any component is changed or extended, revealing all of the dependent upstream and downstream objects that will be affected by the change.

Testing and deployment of complex integration scenarios has always been a difficult task. One of the most powerful features of AIA is the end-to-end integration testing framework provided by the **Composite Application Validation System (CAVS)**. This framework will allow an organization to develop all or part of an end-to-end test scenario and simulate input and output from all applications involved in the integration flow. The complete Applications Integration Architecture is depicted in Figure 1.



Exploring the Value of AIA

Now that we’ve covered the basic concepts of AIA, let’s explore AIA’s value related to a common integration need – keeping customer master data current. Suppose you have an E-Business Suite (EBS) and a custom CRM system implemented at your organization. The E-Business Suite contains the master customer data, and you need to make sure that the CRM is always up-to-date with the most current customer information. You would take the following steps using AIA to create that integration:

- Install the AIA Foundation Pack and the Oracle SOA Suite.
- Start small and design one integration scenario
- (e.g., new or updated account data in EBS is pushed to CRM).
- Use the AIA Enterprise Business Object (EBO) for customer account.
- Write the EBS-side ABC (that translates the EBS customer account data to the EBO) and publish it to the AIA Business Services Repository (or leverage an existing ABC provided by a Process Integration Pack).
- Write the CRM-side ABC (that translates the EBO to the custom CRM system’s version of customer account data) and publish it to the AIA Business Services Repository.
- Set up testing scenarios using AIA CAVS.
- Capture inserted/changed customer account data in EBS and send a message to the CRM system via AIA.
- Test and move into production.

You may wonder, where was the value in that? The process involved several steps and a learning curve; there was most likely a simpler way to accomplish that specific integration. Although this observation is true, the real value of AIA comes in what happens next. Consider the following scenarios your organization may face

- **Pushing master data to additional systems**—Perhaps customer data is also needed in other downstream systems such as a customer portal or a special pricing analytics application. To keep the data current in these systems, you already have the integration half built. You only need to write the downstream-system-side ABCs to translate the customer data from the application-neutral EBO to the specific needs of each of the other systems. The initial AIA integration scenario has set the stage for creating similar scenarios to push the customer master data to other places. Each of the downstream systems may need a different subset of customer attributes. As long as the EBS-side ABC translated the superset of these attributes to the EBO, EBS-side ABC doesn't need to be touched as each new downstream system integration is added.
- **Growing your integration library incrementally**—As you gain more experience with the AIA infrastructure, you begin to add additional integration scenarios between the EBS and CRM systems integrations such as bi-directional syncing of customer data maintenance of shared data attributes, or available-to-promise inquiry, to name a few examples. These integrations can be added one at a time, and since each component in an AIA integration scenario is a discreet artifact, you can easily add integration scenarios without needing to touch existing ones, thus keeping the risk of change low. At the same time, the AIA BSR keeps a clear record of all integration scenarios (and their components) and makes it easy to re-use ABCs in multiple scenarios where appropriate. As a result, you avoid the development and maintenance overhead of duplicated program functionality.
- **Swapping out applications to best-of-breed**—Suppose your legacy CRM system is swapped out for a new best-of-breed CRM off-the-shelf implementation. The EBS-side ABCs remain the same, and only the CRM-side ABCs must be rewritten. The interface-first design (also known as contract-first design) enforced by AIA means that the re-writing of those ABCs will be straightforward, and the CAVS testing suite is in place to help uncover any issues prior to production.
- **Application upgrades**—After creating customer account integrations between EBS and five other systems, you upgrade your EBS implementation to the latest version. With point-to-point integrations that upgrade probably would have required you to review, update, test, and implement five different sets of custom integration code. With AIA, only the one EBS-side ABC will be directly impacted. Although AIA doesn't eliminate the need for re-testing all five integrations, the CAVS testing framework is an asset in that testing, and the AIA-enforced design reduces the likelihood of defects. Additionally, the governance and visibility provided by the AIA BSR gives you greater confidence that there are, in fact, only the five known integrations and not an additional forgotten integration that will surface as a production problem.



- **Process orchestrations**—After some period of time, you begin to add more complex integrations between the CRM system and EBS. For example, you add bi-directional syncing of customer data so that a customer can also be added from the CRM side of the integration as a by-product of booking an order for a new customer. Or you check inventory and perform a credit check before submitting an order. AIA allows you to create BPEL-based multi-step processes in the application-neutral portion of the infrastructure. Putting these process orchestrations here rather than as application-side customizations allows application upgrades to occur on either side of the integration with minimal re-coding to support these complex processes. Only the ABCs associated with the upgraded application need to be reviewed and possibly re-coded. The BSR will facilitate impact analysis to locate all potentially-affected ABCs and the CAVS system will facilitate testing.

As you can see, the real value of AIA comes in its ability to allow your applications to evolve and remain agile over time while providing the governance and tools to reduce both costs and risks associated with cross-application integration. This value only increases as AIA becomes the enterprise-wide standard and the multitude of point-to-point, tightly-coupled, possibly-undocumented, disparately-designed integrations within your organization are replaced with the governance, visibility, and flexibility provided by AIA integrations.

A typical dilemma faced by organizations is when to invest in this architecture and when to just architect a simple point-to-point integration using traditional methods. This decision should be made based on many factors, including practicality, budget, risk, and future integration needs. For example, let's consider a simple point-to-point batch-oriented flat file interface that exists today in an organization between Peoplesoft HRMS and the Oracle E-Business Suite. In this example, the E-Business Suite is being re-implemented, and some minor modifications to the interface will need to be completed. Is this the time to make the AIA investment and re-write the interface to leverage AIA, or should the minor modifications just be made to the existing interface?

In this example, we believe that business benefits will drive the decision. If the existing batch-oriented interface meets all current and likely future business requirements (e.g., data latency, content, future system upgrade requirements), then we would not recommend making the AIA investment at this time. However, if the current batch method does not provide frequent enough updates to employee data in EBS, a real-time update capability is valued, and/or there is an impending Peoplesoft upgrade or re-implementation scheduled in the future, then investing the time and effort to initiate the AIA architecture would be justified.

Even after the AIA architecture has been implemented at an organization, the decisions continue. Let's consider that same example of the simple interface between Peoplesoft and the Oracle E-Business Suite with its need for minor interface modifications. What if AIA has already been implemented? Should you re-write the interface to leverage an existing AIA implementation or make the simple modification to the existing interface? In this

case the answer may be very different. Once the AIA installation has been established, the incremental cost to move individual integrations to it will be small, and, as we have seen, the real value of AIA comes after reaching a critical mass building an organization's integration library. So, while the temptation may be to simply make the minor modification to the existing interface, the better business decision will likely be to re-write the interface to leverage the existing AIA environment.

Getting Started

How can an organization start to take advantage of AIA's many benefits? Depending on your particular needs, Oracle has provided several options. The AIA Foundation Pack delivers all components necessary to take advantage of AIA and contains the same tools that Oracle uses to deliver its pre-built integrations. Oracle has recently introduced some additional industry-specific Foundation Packs (for Insurance, Communications, and Utilities), but for most organizations, the standard, cross-industry AIA Foundation Pack is the place to start. It provides all of the components reviewed in this paper, including:

- Enterprise Business Objects
- Enterprise Business Services
- Business Service Repository
- Composite Application Validation System
- Error Handling Framework
- SOA Governance Tools
- Reference Process Model
- Pre-defined Reference Architecture and Methodology

The Foundation Pack will allow an organization to leverage the EBOs and EBSs provided to create

deployable integrations between any number of systems, including Oracle, third-party, and custom-developed applications. Depending on the type of integration scenario targeted as the highest priority to develop, the components of the Foundation Pack may be the place to start. In this case, an organization would need to build the Application Business Connectors to work within AIA.

In addition to the Foundation Pack, Oracle also provides pre-built integrations in the form of Process Integration Packs (PIPs) and Direct Integrations (DI). These pre-built integrations build on the Foundation Pack components by providing complete integration scenarios between Oracle's enterprise systems. For example, there is a "Design-to-Release" PIP that provides out-of-the-box integration from the Agile PLM product to the Oracle E-Business Suite. Oracle provides a DI that integrates their new Oracle Utilities Customer Care application and the E-Business Suite Financials or Peoplesoft EFM. There are many other PIPs and DI integrations available, and more are being built and delivered every day.

Depending on the type of integration required, it may make sense to start with a PIP and modify it, rather than building the integration from scratch. For example, if an organization needs to build an integration between a custom (or non-Oracle) CRM system and the E-Business Suite, it may be more cost-effective to purchase the "Oracle CRM On Demand Integration Pack for Oracle E-Business Suite." Starting with that PIP, the organization would simply modify and/or extend the AIA components that were pre-built to interact with the Oracle CRM application to work with the custom/non-Oracle CRM system instead.



Summary & Conclusion

Oracle's Application Integration Architecture is a groundbreaking approach to deploying enterprise-wide systems integration. Using AIA, organizations will be able to deploy integrations between all types of systems in a loosely-coupled, scalable, and high-reliability manner. The concepts of AIA are clear and straightforward, and will support an evolutionary approach to a world-class integration architecture platform. Understanding the components that comprise AIA will help organizations leverage this architecture more quickly and overcome many of the difficult integration issues that have plagued them for years.

For More Information

There are many excellent resources for learning more about AIA. Key Oracle White Papers and web links for reference include:

General information on AIA:

<http://www.oracle.com/applications/oracle-application-integration-architecture.html>

<http://www.oracle.com/applications/oracle-aia-resource-library.html>

Information on prebuilt integrations:

<http://www.oracle.com/applications/prebuilt-integrations.html>

Foundation Pack Data Sheet:

<http://www.oracle.com/applications/aia-foundation-pack-data-sheet.pdf>

AIA FAQs:

<http://www.oracle.com/applications/application-integration-architecture-faq.pdf>

The Authors:

Mike Butler is the Vice President of Oracle Applications for TUSC. He has more than 27 years of business systems experience and has been working with Oracle Applications for more than 18 years. During that time, Mike has designed numerous Oracle E-Business Suite solutions for organizations around the world. He is a former Board Member of the North Central Oracle Applications User Group (NCOAUG).

Susan DiFabio is a Management Consultant at TUSC specializing in system design of enterprise-wide applications and complex data integrations. Throughout her 25 years in consulting she has focused on cost-effective frameworks, re-usable components, and business-focused solutions. Most recently, Susan designed and oversaw the implementation of a multi-faceted insurance data integration application for a major risk-management information services company.