



*White Paper*

# **Service-Oriented Architecture: Aligning Technology with Business Objectives**

Technical White Paper

November 2007

This paper describes Service-Oriented Architecture and ITI's approach to SOA. It also describes the platform, framework, services and tools that implement the ITI suite of products known as *Premier SOA*.



## Table of Contents

Foreword.....	5
The Promise of SOA.....	6
Service Oriented Architecture (SOA).....	6
SOA Principles .....	7
<i>Loosely Coupled Services</i> .....	7
<i>Coarse-Grained Services</i> .....	7
<i>Orchestrated Services</i> .....	7
<i>Synchronicity</i> .....	7
<i>Composite Applications</i> .....	8
<i>What SOA Is Not</i> .....	8
SOA Objectives .....	9
<i>Reuse</i> .....	9
<i>Agility</i> .....	9
SOA Maturity Model.....	10
<i>Level 1: Initial Services</i> .....	10
<i>Level 2: Architected Services</i> .....	11
<i>Level 3: Business and Collaborative Services</i> .....	11
<i>Level 4: Measured Business Services</i> .....	11
<i>Level 5: Optimized Business Services</i> .....	11
<i>A Mature SOA Can Transform Business Processes</i> .....	12
Premier SOA.....	13
<i>Milestones</i> .....	13
<i>ITI's Approach to SOA</i> .....	13
Premier SOA Tool Set.....	15
<i>Message Catalog</i> .....	15
<i>Framework Reference Guide</i> .....	15
<i>Transformation Generator</i> .....	15
<i>Metadata Generators</i> .....	15
<i>Code Generators</i> .....	15
Premier Service Bus.....	16
Premier SOA Framework .....	17
<i>Messaging Engine</i> .....	17
<i>Registry Services</i> .....	17
<i>Transformation Engine</i> .....	17
<i>Security Engine</i> .....	17
<i>Identity Services</i> .....	17
<i>Licensing Services</i> .....	17
<i>Policy Services</i> .....	17

<i>Metadata Engine</i> .....	17
<i>Auditing Engine</i> .....	17
<i>Rendering Engine</i> .....	17
Premier SOA Architecture.....	18
<i>Channel</i> .....	19
<i>Applications</i> .....	19
<i>Premier SOA Framework</i> .....	19
<i>Data Stores</i> .....	20
<i>Premier Service Bus</i> .....	20
Summary.....	20
Appendix: Premier SOA Case Studies .....	21
<i>Case Study: Aggregation</i> .....	21
<i>Case Study: Embracing Best of Breed Applications</i> .....	21
<i>Case Study: Interoperability</i> .....	21
<i>Case Study: Collaborative Services</i> .....	22
<i>Case Study: Transaction Management Across the Enterprise</i> .....	22

## Foreword

Gradually, over the past decade or so, a paradigm shift has occurred in the field of information technology (IT). We are accustomed to computers and computer programs automating and expediting formerly labor-intensive and time-consuming processes. But, when business needs change and require new processes, computer programs also need to change. And therein lies the crisis that prompted the paradigm shift. The very computer programs that speed up business processes have generally required time-consuming, labor-intensive development. Even minor changes in business process can require a major overhaul of many programs across the “silos” of the enterprise. IT development needs to be as nimble as the businesses processes to which it has become integral.

A new approach to IT development – service-oriented architecture (SOA) – has emerged. This approach involves standards-based, interoperable and reusable components that can be assembled into composite applications.

This white paper describes how Information Technology, Inc. anticipated this paradigm shift, successfully negotiated it by developing a mature service-oriented architecture (SOA), and continues to refine the means of making IT more readily responsive to changing business needs.

As a technology leader in the financial industry, ITI understands the need for SOA, which not only enhances IT responsiveness but also facilitates integration. Committed to the secure, open and flexible integration required by financial institutions of all sizes, ITI and Fiserv (ITI’s parent company) have been aggressively pursuing a corporate-wide integration initiative over the past several years. The objective of this initiative has been to foster a standards-based architecture that provides a common means of sharing data, services, and business processes among ITI, Fiserv, financial institutions and third parties.

ITI’s first Web-based services were constructed six years ago, and numerous improvements to the SOA of *Premier*<sup>®</sup> have followed. The goal of the company’s substantial investment in SOA was simple – to deliver the advantages of SOA to financial institutions, and to do it better than anyone else.

## **The Promise of SOA**

For financial institutions, business requirements change at a dizzying rate. Technology should enable rather than impede implementation of marketing strategies and compliance with new regulations. When an organization needs to respond quickly, IT cannot take months to catch up. What's required is business agility, and the promise of business agility lies at the heart of service-oriented architecture (SOA).

The benefits derived from SOA can be dramatic. Ideally, SOA aligns technology with business processes so that components can be changed or choreographed to meet new business conditions. This type of approach allows financial institutions to maximize the value of current IT investments and minimize the risk associated with new development. Financial institutions also benefit from easier integration, lower development and maintenance costs, faster time to market and a greater return on investment.

## **Service Oriented Architecture (SOA)**

Descriptions of SOA often appear similar on the surface. But these descriptions begin to diverge abruptly as implementation details are revealed, because there are many ways to implement SOA, even when an organization is committed to "industry standards."

Most descriptions of SOA begin with a statement about an organization exposing its underlying business processes through services (often Web services). These services are made available over industry-standard protocols that eliminate interoperability issues across disparate computing platforms. After that, the details and techniques tend to apply more specifically to particular implementations, even though the objectives of different implementations remain the same.

SOA has been described as a methodology, an architecture, and as a technique. ITI describes SOA as a design philosophy that maximizes reuse of interoperable services in order to provide greater efficiency and adaptability. The SOA approach supports the creation of loosely coupled business services that can be easily shared between enterprises.

Despite the great disparity among the definitions, approaches and degrees to which various organizations are implementing SOA today, all implementations typically embrace the following SOA principles.

## SOA Principles

In keeping with the principles of SOA, services are loosely coupled and interoperable, standards-based, coarse-grained and aligned with business processes. Orchestration of services is another hallmark of SOA. Composite applications can invoke multiple services.

### *Loosely Coupled Services*

Loosely coupled services support interoperability across disparate computer systems. These systems often use a technology such as SOAP (although comparable technologies such as REST are also used). SOAP is often referred to as the “universal connector,” because it is based on ubiquitous technologies (XML and HTTPS) that are compatible across many computer systems.

Decoupling applications from business processes removes the inflexibility normally associated with tightly-coupled systems. This is often achieved by replacing compile-time API calls with Web service interfaces, which are described through the Web Service Definition Language (WSDL) and bound together at run-time rather than compile-time.

### *Coarse-Grained Services*

Coarse-grained services are better aligned with business processes than fine-grained services. They deliver a business function, rather than a toolkit of technical components whose complex interactions must be understood to effectively create and reuse a business function.

### *Orchestrated Services*

Orchestration provides control over the flow between business processes at a high level. Orchestration (which is often discussed in connection with Business Process Management, Process Automation, and Workflow) empowers financial institutions to dynamically assemble business services to match the ways their organizations operate.

Common orchestration patterns include an “embedded workflow” pattern, often used within applications, and a “standalone workflow” pattern, which is provided by orchestration products external to an application. Both of these patterns can reduce reliance on IT services by providing a high-level interface for orchestrating the interaction of services used to solve a business problem.

### *Synchronicity*

Services may support both synchronous and asynchronous modes. The mode chosen often depends on the duration of service, system availability, and reliability requirements. Complex business processes running across distributed systems frequently require support for “long-running transactions” and “reliable messaging.” This requirement is often met through the use of Message-Oriented Middleware (MOM), which supports capabilities such as Message Queuing (or MQ). Additional capabilities such as “store and forward” and “publish/subscribe” may also be supported through MOM.

### *Composite Applications*

Well-designed atomic business services make composite applications possible. New composite applications result from the aggregation of data and business processes exposed through a collection of services. A composite application often possesses a user interface and differs from a workflow application in that it does not manage long-lived transactions. Composite applications may, however, hold internal state while invoking multiple services.

### *What SOA Is Not*

SOA is not tied to any specific technology, methodology, framework or tool, and SOA does not require Web services, SOAP, XML, MQ or an enterprise service bus (ESB).

### **SOA FAQ**

**Question:** So how do you implement SOA if there are no standard requirements?

**Answer:** You base your implementation on the principles mentioned — loose coupling, interoperability, and coarse-grained services that are closely aligned with business processes.

## SOA Objectives

At the most fundamental level, there are two objectives when implementing SOA: reuse and agility.

### *Reuse*

The first objective is to save money by reusing services (i.e., reuse). The more services are reused, the more return on investment (ROI) is derived from the SOA.

### *Agility*

The second objective is to increase revenue by improving an organization's ability to adapt to changing business needs (i.e., agility). This, of course, provides a strategic advantage in terms of time-to-market.

## **SOA FAQ**

**Question:** How does SOA differ from other similar efforts?

**Answer:** There have been other attempts to achieve reuse such as CORBA. CORBA was based on object-oriented architecture (OOA). The primary difference between OOA and SOA is the concept of a "service." In general, OOA is more granular than SOA, and is aligned more with program functions than with business processes. A "service" corresponds to a unit of business activity, and only those properties associated with that activity are exposed in the interface. By contrast, OOA typically defines the underlying components required to complete the business activity. With OOA, a great many objects would be published to deliver the same functionality delivered by a single service. The consumer of those objects would need to know which ones to call, in what order, how to communicate between them, how to reverse the order in case of an error, and so on. This is much too granular to be described as a "service."

ITI had an API that exposed 162 calls to objects. These calls were replaced with two services, which were then published. There are vendors who have rushed to claim they have SOA, but they have done nothing more than put Web service wrappers around their existing objects. These objects are not services, and such an approach will not achieve the underlying goals of SOA.

Tight coupling is a second differentiator between SOA and earlier approaches. SOA is loosely coupled. Other approaches have typically used more tightly bound Remote Procedure Call interfaces that compromise cross-platform interoperability, one of the chief principles of SOA.

There are, of course, other objectives that have been associated with SOA. Some organizations are satisfied to simply expose existing functionality. Others, like ITI, see an opportunity to improve customer return by pursuing objectives that fall on a higher level of the SOA Maturity Model (described below). As an SOA matures, organizations find that there is more to it than technology and standards. SOA impacts process, organizational structure, governance and design methodologies. The impact reaches deeper into the business as the business progresses up the SOA stack.

As outlined below, ITI recognizes five levels of SOA maturity into which most SOA implementations fall. Many organizations still find themselves dabbling in the first or second tier, while ITI is successfully navigating the higher tiers and continues to pursue the very highest SOA objectives, from which the greatest value can be realized.

## SOA Maturity Model

Several maturity models have been published that seek to provide assistance to those trying to measure the maturity of a service-oriented architecture. The model promoted by a consortium of industry leaders\* (and paraphrased below) is the model that ITI has used to evaluate the progress of *Premier SOA*

### *Level 1: Initial Services*

Web services are based on industry standards and provide interoperability and openness across an enterprise. Allowing the functions of a business process to be exposed as Web services means they can be leveraged in virtually any environment and that the logic is only codified once. The ability to reuse these services pays dividends in terms of maintenance (due to reuse). Benefits include productivity when these services are used as building blocks to assemble new applications.

Enterprise-wide service integration usually involves an incremental approach. A small set of business functions are exposed, and then more are added as value is realized and resources permit. In time, the business processes of the entire organization can be exposed. The availability of services across an enterprise breaks down the application silos that have been constructed over the years and makes it possible to cross lines of business, aggregate data from disparate and multiple sources, and even create new value where none previously existed.

---

\* Sonic Software Corporation, Actional, Progress Software, DataDirect Technologies, *A New Service-Oriented Architecture (SOA) Maturity Model*, 2006.

### ***Level 2: Architected Services***

SOA requirements include support for authentication, authorization, service discovery, logging, session management, and security. These facilities, provided within an architected infrastructure, provide IT cost reduction and control by ensuring that these requirements are not built for each new service. This also results in risk mitigation because of the reduction in development.

### ***Level 3: Business and Collaborative Services***

Business-oriented service integration is not realized by simply putting wrappers around existing objects. Many application objects are too granular to meet the definition of a service. A service should expose an entire business function rather than parts of a transaction. A fund transfer service, for example, has a level of granularity consistent with SOA principles. However, the objects used within the fund transfer (debit account, credit account, settlement account, or the transfer process itself) do not meet that same standard. The consumer of a fund transfer service should be able to expect that an entire business function will be completed in a single service call. That consumer should not be required to master the complex interactions between fine-grained objects.

Systems that merely wrap existing objects may claim to use SOA techniques yet fall well short of SOA goals. Fine-grained services that force a detailed understanding of the application architecture upon the consumer of the services are one of the problems SOA is supposed to resolve. To avoid this problem, ITI has approached SOA by closely aligning services with business functions.

### ***Level 4: Measured Business Services***

Often, as an SOA expands across an enterprise, the organization realizes a need for feedback on the performance and business impact of the architecture. Business activity monitoring becomes important, providing real-time business performance metrics. Businesses may need to achieve specific service-level agreements. *Premier SOA* supports these activities through logging, service handshakes that gather performance data, and a licensing model that includes metering and usage data. These capabilities demonstrate commitment to evolving the architecture from a reactive mode to one based on real-time intelligence.

### ***Level 5: Optimized Business Services***

Optimized Business Services, the highest level of the maturity model, build on the Measured Business Services of the preceding layer. With this capability, the architecture takes action on the events occurring at the business level according to rules that optimize business goals. Using SOA allows such an approach to be more easily implemented and evolved than is possible with traditional architectures.

### ***A Mature SOA Can Transform Business Processes***

Most organizations are operating in the lower tiers of the model. However, *Premier SOA* provides customers with an opportunity to change the way they do business by exploiting higher levels of SOA maturity. As organizations learn to leverage *Premier SOA*, they find that entire business models can be transformed.

Opportunities to assemble business processes and aggregate information suggest a potential for new ways to do business. Organizations might develop composite applications that allow their organization to implement a self-service model to better serve customers and reduce their own support costs. They might find themselves able to resell services to partners wanting to aggregate services and data controlled by their systems. They might break down silos of information in their organization by orchestrating the behavior of a collection of services. These are all examples of how an organization might transform the way it does business.

#### **SOA FAQs**

**Question:** How does SOA differ from legacy architectures that are monolithic in nature?

**Answer:** Modern architectures typically separate application layers into presentation, business logic, and data tiers. Three-tier architecture is intended to allow any of the three tiers to be upgraded or replaced independently as requirements or technologies change. The problem, however, is when developers create a Web-service façade for a component where the tiers have not yet been separated (i.e., business and data tiers may still be together in a single component). These are not services, and will not enable one to achieve the goals of SOA.

**Question:** What makes the *Premier SOA* approach superior?

**Answer:** Applications have already been separated into three tiers. Services are coarse-grained (meaning that they align with the processes of a financial institution), which results in the kind of building blocks promised by SOA. Components are loosely-coupled. Services are interoperable. The architecture supports service orchestration through both embedded workflow and commercial workflow products. The architecture integrates a lightweight service bus, yet Web services (the universal connector) allow virtually any commercial ESB product to leverage *Premier SOA*, which supports secure, reliable messaging. Message-Oriented Middleware (MOM) supports message queues, which provide the foundation for long-running transactions, store and forward, replication, publish/subscribe and other technical requirements.

## Premier SOA

*Premier SOA* is not a radical new approach for ITI. There has been a quiet revolution over the past several years, and ITI is proud that this architectural foundation has been put into place without disrupting normal product release schedules.

In 2001, ITI recognized that emerging technologies such as Web services would have a dramatic impact on the ability not only to share services, but also to share them across the enterprise and across disparate systems. The leap in terms of interoperability, along with the ability to quickly respond to business needs, showed great potential. It was that potential, coupled with emerging SOA technologies, that prompted ITI to embrace SOA.

### *Milestones*

- 2001 – 2002: Initial services constructed. Customer integration includes aggregation of account lists (bank, securities, trust, etc.). Integration includes an Image API which is leveraged by both e-commerce and core applications.
- 2003: Architected services. First significant exposure of the core *Premier* database to external parties (third-party portal product). Framework services include registry for service discovery, as well as authentication, authorization and session management.
- 2004: Collaborative services. Complex cross-enterprise integrations resulting in composite applications achieved by working with external vendors. Examples include the *Premier Customer Profitability Package*. Technical milestones include single sign on, federated identity management and screen pops.
- 2005 – 2006: Measured services. License management and provisioning provide metrics on service usage. Log and event management across applications. Mediation strategy using XML appliances is adopted.
- 2007 – 2008: Plans include addressing optimization of *Premier SOA*, extension of service bus architecture, enhancement of workflow capabilities and improved support for event-driven business services. Application services will continue to be published and exposed to partners.

### *ITI's Approach to SOA*

The approach that ITI has taken to services is document-centric and leverages a number of related protocols, tools and languages. *Premier SOA* takes advantage of the affinity between XML documents, the Xpath language, and the network transport (SOAP) used to exchange documents. This approach is ideal for coarse-grained components which are more naturally aligned with business functions.

Assembling new, composite applications from a collection of services (and from a variety of sources) is one of the goals of SOA. However, bringing together services from disparate sources introduces a new potential for dependency that must be countered. One cannot afford to be tightly bound to components in another system, or to the release schedules of partners. *Premier SOA* is faithful to the principle of loose coupling by natively exposing business functions through Web services rather than relying on adapters and protocol conversion layers. This makes ITI an ideal integration partner.

Unlike ITI, most organizations have not rewritten their applications as Web services. Many find the goal prohibitively expensive. That has given rise to a popular type of product known as an enterprise service bus (ESB). Service bus products enable software to communicate regardless of platform, device, language, programming model or data representation by providing mediation layers between technologies and message formats.

ESBs are an important feature in the SOA landscape. Many have used ESBs as a way to expose functionality through services. They can, however, undermine a “service-oriented architecture.” For example, if an existing system is comprised of fine-grained objects rather than business-level processes, the result is not truly a service-oriented architecture. This kind of implementation may solve a connectivity issue but it does not achieve the most important goals of SOA (reuse and agility), because it requires those who leverage the technology to know too much about the interactions between the parts. Metaphorically speaking, it gives them a box of parts, rather than a ready-to-use appliance.

*Premier SOA* is comprised of a tool set (that helps foster a culture of reuse at ITI), a platform (that supports horizontal requirements such as messaging), and a framework (for common services such as identity management). *Premier SOA* is a prerequisite for using the various business services made available through ITI applications. Services provided by ITI’s professionals – custom development, best-practices consultation, business and project management services, etc. – complement the flexibility and functionality of *Premier SOA*.

ITI has a reputation for strong product integration, but *Premier SOA* takes that integration to another level. *Premier SOA* makes it possible to integrate not only the ITI product suite, but also applications of ITI partners, applications built by ITI customers, and even applications that have yet to be imagined.

## **Premier SOA Tool Set**

The *Premier SOA* tool set is a collection of patterns, tools and documentation that supports *Premier SOA* and includes those items described below.

### ***Message Catalog***

The Message Catalog is a UDDI-based tool used to catalog messages and services.

### ***Framework Reference Guide***

The Framework Reference Guide is a software development lifecycle (SDLC) tool detailing message and service architecture. It details standards and recommendations for the design of schemas and messages, security, and the implementation of services.

### ***Transformation Generator***

The Transformation Generator is a development tool for generating Data Transformation Engine code for the business logic tier (often referred to as scrubbers). The tool generates transformation code between XML messages and native data formats.

### ***Metadata Generators***

ITI has developed a series of generators that will help automate much of what ITI has learned to do well within *Premier SOA*. Using data models and metadata, these tools will generate message schemas, API documentation, help files, and language-based metadata. Because ITI has embraced the spirit of SOA, as well as a number of related industry standards, ITI is positioned to automate development processes, which will expedite development and lessen the risk of error in code or documentation.

### ***Code Generators***

Code generators generate text programs for every Web service. These are used internally, but are also shared with partners who seek to leverage *Premier SOA*.

## **Premier Service Bus**

Services of the Premier Service Bus play an integral part in extending *Premier's* core architecture for large, complex deployments where integration includes disparate platforms and multiple vendors.

The Premier Service Bus provides run-time routing of messages based on message content (e.g., user, group, institution, application, database, etc.). It also calls the transformation engine that transforms data between native data formats (flat files, SQL Server tables, DB2 tables) and XML-based messages. Commercial enterprise service buses (ESBs) provide more fully featured adapters (for converting between technologies, protocols, and data formats).

Because *Premier SOA* exposes business functions as Web services, it does not require a full-featured ESB product for communication between *Premier SOA* applications and services. They all understand Web services and fiAPI messages (fiAPI is the acronym describing Fiserv's corporate API message standards). Yet when integration involves many partners, partners who have not exposed their business functions as Web services, or environments where common message formats are not used, a fully featured ESB may be necessary.

In most current deployments, the Premier Service Bus meets customer requirements, but ITI also has deployments where an ESB is used. The Sonic ESB product, for example, supports the integration between ITI and Precision Computer Systems (PCS), where ITI's *Premier Platform* and ATM services are being consumed by PCS. ITI is an ideal partner in either environment, because most business functions have already been exposed as Web services, which enjoy universal support among ESB vendors.

## **Premier SOA Framework**

The *Premier SOA* Framework is a collection of components, services and tools that implements *Premier SOA*. Those components are described below. Note that framework components described as “engines” are tightly bound dynamic-link libraries (DLLs) and are not typically available to non-ITI applications or partners. A framework component described as a “service” is a loosely bound component that is often available to non-ITI applications or partners.

### ***Messaging Engine***

The Message Services Engine provides run-time services for message handling construction and processing.

### ***Registry Services***

Registry Services provide a (UDDI-like) Web service that service consumers use to locate service providers at run-time.

### ***Transformation Engine***

The Transformation Engine is a run-time engine for transformation between XML messages and native data formats.

### ***Security Engine***

The Security Engine provides run-time services for security-related functions such as encryption, hashing, and encoding.

### ***Identity Services***

Identity Services provide run-time Web services for authentication, authorization and integrity (including federated identity management, digital signatures and message digests).

### ***Licensing Services***

Licensing Services provide run-time Web services for service provisioning and license enforcement (including run-time metrics, monitoring and management).

### ***Policy Services***

Policy Services provide run-time services for policies. Policies include rules related to institutional controls, application-specific specifications, access controls, etc.

### ***Metadata Engine***

The Metadata Engine provides run-time services for metadata. Metadata includes data descriptions related to all data elements rendered in applications (including message data, display data, help data, etc.). This engine includes support for multiple languages and vocabularies.

### ***Auditing Engine***

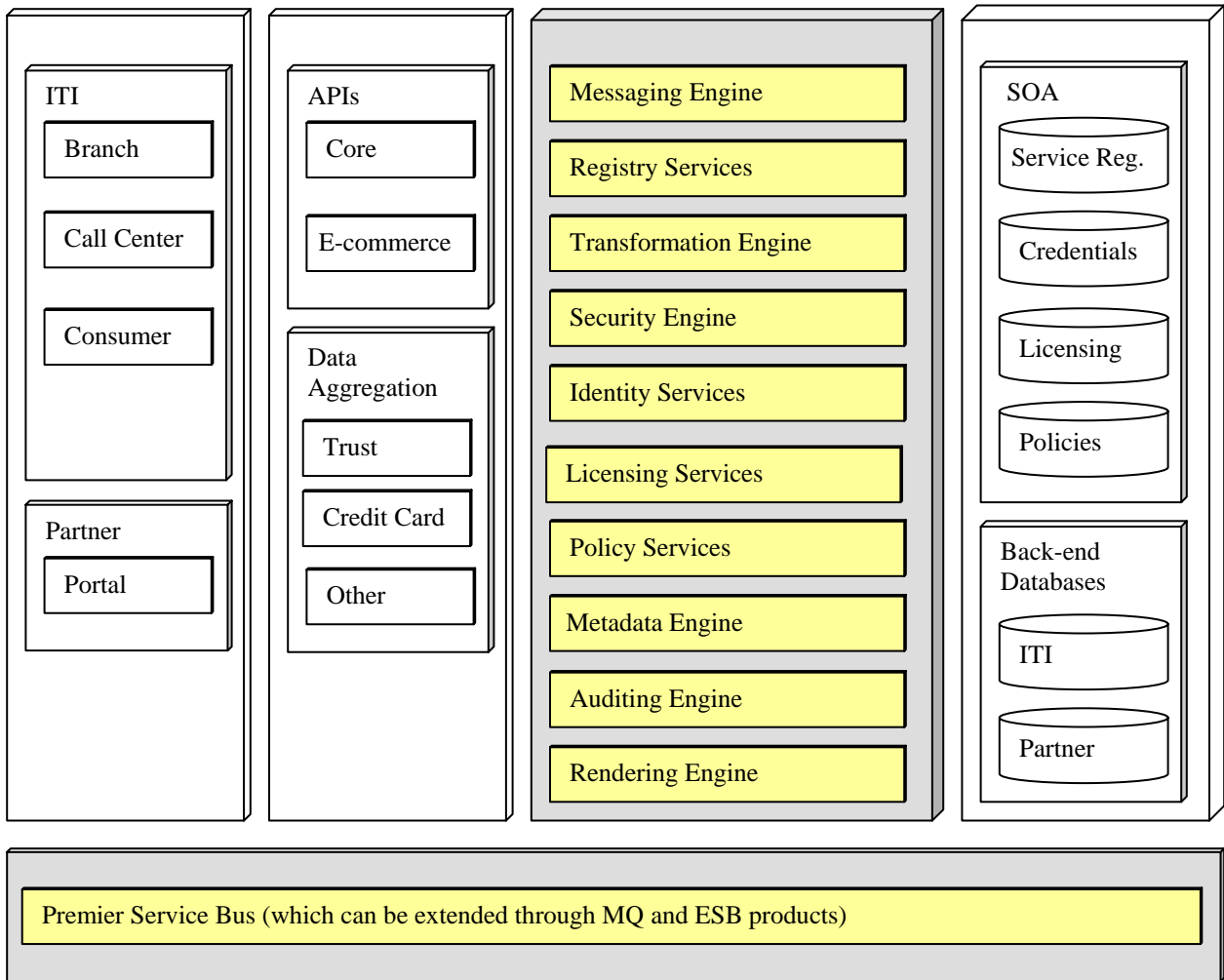
The Auditing Engine provides run-time services and tools for maintaining database audits, error logs, and traces. Related tools exist to manage logs (including the merging/collating of logs involved in distributed architecture). These tools have not yet been made available outside of the ITI development environment.

### ***Rendering Engine***

The Content Rendering Engine is a run-time engine for rendering User Interfaces (such as browser pages) and includes such things as page architecture controls, display widgets, etc. This engine is exposed only to ITI developers through the ITI framework.

## Premier SOA Architecture

The diagram depicts relationships between architectural components (see detailed explanation below).



### ***Channel***

Channels can include but are not limited to browser-based products, rich clients, mobile devices, composite applications and other middle-tier applications.

- ITI: Applications (Branch, Call Center, Consumer, etc.) have access to *Premier SOA* as long as controls are satisfied (access controls, license controls, institutional policies, etc.)
- Partner: Applications (such as a third-party portal product) can leverage *Premier SOA* as long as controls are satisfied. Controls include access controls, license controls and institutional policies, and often require federated identity management controls.

### ***Applications***

Application-level services such as balance inquiries, account maintenance and fund transfer are made available to both ITI and non-ITI channels through “application” services. This is a partial list depicting the types of services that are available:

- Core (e.g., account inquiry and maintenance messages)
- Document Management (e.g., statements, notices, etc.)
- E-commerce (e.g., single sign on, account inquiry, statements, transfers, bill payment, etc.)
- Account Services (e.g., checking, loan, etc.)
- General Ledger
- Transaction Management
- ACH Processing
- Exception Item Processing

### ***Premier SOA Framework***

Services and engines provide the service infrastructure required to build an application based on SOA. All of these facilities are available to ITI applications, and many are made available to third parties.

- Registry (such as get service location)
- Identity Services (such as login, single sign on, and get authorizations)
- Licensing (such as license inquiry and maintenance, including metrics)
- Transformation (used to transform between fiAPI XML messages and native data formats)
- Messaging (used to construct fiAPI XML messages)
- Security (supports W3C encryption [AES], W3C hashing [SHA], Message Digest RFCs, etc.)
- Policy (provides access to institution policies)
- Metadata (one way to get to field-level data such as description, labels, formats, etc.)
- Auditing (logging consistency and control)

### ***Data Stores***

Data Stores include data stored in ITI databases (relational, flat, XML) as well as third-party databases that are either imported through Extract, Transform, and Load (ETL) processes or data made available because integration capabilities have already addressed issues like federated identity management with certain partners.

- Third-party database access (such as trust or securities applications).

### ***Premier Service Bus***

An integrated, lightweight service bus, the Premier Service Bus can be complemented in large, complex environments by enterprise-class service bus products that contain more robust support for data transformation, as well as technology and protocol mediation.

## **Summary**

As noted earlier in this document, there are two fundamental objectives when implementing SOA. The first objective is to save money by reusing services (reuse). The more services reused, the more ROI derived from the SOA. The second objective is to improve an organization's ability to adapt to changing business needs (agility). This strategic advantage allows a business to improve chances for long-term survival.

While *Premier SOA* continues to mature, many promises of SOA have already been realized. *Premier SOA* is helping financial institutions to save money and to respond to changing business requirements. These financial institutions are well-positioned to transform the way they do business. Appended are five case studies illustrating the transformative power of *Premier SOA*.

## Appendix: Premier SOA Case Studies

### *Case Study: Aggregation*

Some of the earliest uses of *Premier SOA* involved aggregation of data. Many of them began in the area of e-commerce and pursued the goal of “wealth management” by aggregating various customer account relationships through a single user interface. *Premier SOA* was exposing business functions (such as consolidated account inquiry) through Web services as early as 2002. The basis of that functionality exists in middleware, and ITI has continued to build upon those early successes – adding services to support functionality such as alerts, document search, and multifactor authentication, to name a few. The majority of these efforts involve real-time aggregation of data (such as account or customer relationship information). But one recent example relies on *Premier SOA*'s ability to transform bulk data from multiple sources into a local database that is then exposed through *Premier SOA* services. The data provided through these services may reside in relational or legacy databases. The nature of the back-end technology makes no difference to the consumer of the business service. In accordance with one of the important tenets of SOA, data can reside on virtually any platform and be based on a wide variety of technologies. The real key to achieving such SOA goals lies in observing industry standards such as HTTP, XML, and SOAP.

### *Case Study: Embracing Best of Breed Applications*

Another case study underscores the fact that *Premier SOA* can be an integral part of best-of-breed enterprise application integration. While ITI might prefer to sell a complete suite of products to every customer, today's financial institutions demand flexibility. Many prefer to assemble a suite of applications from various vendors. *Premier SOA* supplies the flexibility these organizations require. In this particular case study, a non-ITI customer relationship management system needed to leverage *Premier SOA* loan services in order to provide a more complete view of the customer. ITI was pleased to be able to provide the financial institution a more robust loan system than the competitor could provide. Service-oriented architecture makes for strange bedfellows, but the ability to react to customer needs and business opportunities is one of the reasons SOA exists.

### *Case Study: Interoperability*

One of the goals of SOA is interoperability, and other vendors' software products running on various platforms have no trouble leveraging *Premier SOA* Web services, whether integration occurs with third parties or within the Fiserv family. One example is the reselling of the InSight Teller product within the *Premier* customer base. The teller application is able to leverage *Premier SOA* Web services, even though it shares little with *Premier* in terms of technology. InSight Teller was developed with Progress, runs on an IBM pSeries and has an environment that includes Sonic ESB. Enterprise service bus (ESB) products, which use an “adapter” approach to mediate disparities in technology, are not typically a part of the infrastructure of *Premier SOA* deployments. Even so, the integration was easily accomplished, because *Premier*'s business services are exposed as Web services (i.e., the “universal plug”), which are supported by all ESB vendors.

### ***Case Study: Collaborative Services***

As *Premier SOA* matured, it graduated beyond simply exposing business functions as services. The architected services that provide a common infrastructure for such things as authentication, metadata, policies, and logging were completed, which brought “workflow” within reach. *Premier SOA* has its own embedded workflow capability. But the goal for *Premier SOA* was larger than supporting a single, proprietary approach to workflow. This larger goal was to create an architecture that other workflow products (particularly large, standalone, enterprise-class business process management applications) could leverage. It was achieved when a client successfully deployed Adobe Workflow to orchestrate *Premier SOA* business processes. These processes enabled the financial institution to add debit card information to the core account database.

### ***Case Study: Transaction Management Across the Enterprise***

This case study illustrates integration between an Internet banking application, a core banking application, and a brokerage system. The functionality supported includes real-time transactions. One particular aspect of this integration shows how *Premier SOA* has matured to a level that many have not yet reached: specifically, it supports transaction management at the service layer. This is important because transaction management is traditionally performed within the database management system. But, in a service-oriented world, applications can be comprised of services from disparate systems sewn together to meet a particular business need. A common deficiency encountered in such implementations is that database management systems do not provide transaction management across databases – much less across enterprises. In this case, a fund transfer required that a debit be sent to one system and a credit be sent to the other. These systems were geographically dispersed and used dissimilar database technologies, yet there was a need for transaction management (i.e., if the credit side of the transaction failed, the debit side had to be “rolled back”). The *Premier SOA* Message Architecture can establish and manage transaction boundaries at the service layer. Doing this at a service layer rather than at a database layer provides broader reach and more control within enterprise-class applications.