

Three Basic Approaches and One Central Repository to Service-Enable Your Enterprise Applications

A winning competitive strategy requires sustainable, profitable growth that's fueled by innovation — but this is easier said than done. Innovation is always slowed down by the need to make improvements to a solution without breaking it in the process.

Take a typical sales order process, for example. If a company wants to add a credit limit check to the process, this change traditionally involves developers taking apart existing applications and replacing thousands of lines of code, which is not a quick job. But an enterprise can more easily execute this credit limit check — or any other process change — if it can:

- Reuse existing IT assets
- Streamline operational and administrative IT processes to reduce TCO and increase cash flow
- Innovate business processes and continuously adapt them

This means that business and IT teams have to communicate fluently with each other to understand the goals and hurdles from both a business process perspective and a technology perspective. To do this, IT teams must consider innovative new functionality in the context of working business processes and put it all into a tidy technology framework that enables business change. But how?

SOA's a Good Start, but Don't Implement Technology at the Expense of Working Processes

From a software point of view, both IT and business requirements can be fulfilled if business applications follow a service-oriented architecture (SOA), where standardized processes are constructed out of independent services made available on a network.

The constructs of the typical SOA approach, however, tend to focus exclusively on the technology framework — and neglect the larger business process. A combined focus on both application and infrastructure is needed to make this business-IT symbiosis work.

SAP uses its business experience and enterprise software to provide exactly this combination: Enterprise SOA is SOA plus something called *enterprise services* — aggregations of Web services glued together by business logic to help execute a business process. In other words, enterprise services are Web services with an enterprise-level business meaning. Enterprise SOA merges SAP's enterprise application content with the open SAP NetWeaver platform so that partners and customers can simply plug in reusable application snippets and enable flexibility within business processes — even active ones.

For example, the credit limit check mentioned earlier can be rendered as a

Regular Feature

SAP NetWeaver Unleashed



Sindhu Gangadharan, SAP AG

service that plugs into the existing sales order process — without replacing code or breaking the business process. The same credit limit check service can then be reused in a data order or purchase order process as well.

The term "business content" refers to SAP business know-how and industry best practices that are made available to customers as a set of service-enabled objects and components.

To get started with enterprise SOA, companies must service-enable their enterprise's applications. Service-enablement refers to the isolation of an enterprise application's specific functionality and its implementation through open standards to make the functionality accessible as a service. Such service-enablement is the basis for enterprise SOA-compliant applications.

How Exactly Does IT Go About Service-Enabling Applications?

Depending on a company's needs, IT teams can follow a combination of three approaches for exposing applications or functionality as Web services:

1. Your development team is building a new application or part of an application and wants to expose it as a Web service. For example, if you are a large enterprise that wants to provide its subsidiaries with a billing application, you only need to build the billing application's underlying services once, and then the same services can be reused according to the different subsidiaries' requirements. This is called **service-oriented development**.
2. You want to expose existing functionality — such as pricing engine functionality, which is part of your standard ERP functions — as a Web service. We'll term this **point-to-point, services-based integration**.
3. You're looking to service-enable legacy application functionality that isn't technically savvy enough to be made directly available as a service — for example, an order management application that resides in a legacy

system. Here, **brokered services-based integration** is your best option.

These three approaches are the basis of SAP's *Enabling Enterprise Services* IT scenario (see **Figure 1**). As you've learned in previous *SAP NetWeaver Unleashed* columns, IT scenarios are packages of tools, roadmaps, best practices, and other deliverables designed and delivered to help customers make use of the out-of-the-box content and capabilities that are built into SAP offerings through the SAP NetWeaver platform.¹ Specifically, SAP's *Enabling Enterprise Services* IT scenario is a way of packaging the required capabilities, tools, infrastructure, and knowledge that companies require to extend their solutions by providing and integrating enterprise services into their business processes.

Let's look at each of these approaches for service-enabling application functionality — otherwise known as *scenario variants* — in more detail. We'll also explore how to leverage the functionality and IT processes delivered within these

¹ For a refresher on IT scenarios, see the article "IT Scenarios Provide a Guided, Business-Oriented Approach to Maximizing SAP NetWeaver Use" by Claudia Weller in the July-September 2005 issue of *SAP Insider* (www.SAPinsider.com).

scenario variants to snap in enterprise services and take those initial steps toward enterprise SOA.

Developing Agile, Services-Based Applications

Using the *service-oriented development* approach, customers can implement enterprise services right away by using the services that are part of enterprise suites such as mySAP Business Suite² or by creating custom services as required.

To support service-oriented development, SAP provides a new *Enterprise Services Repository* (ESR) that stores all the underlying metadata of the application objects, for example, data types, message interfaces, and service definitions (see **Figure 2** on the next page). The first implementation of the ESR and its associated editors comes in SAP NetWeaver 2004s and is based on the SAP NetWeaver Integration Repository, which evolved from SAP NetWeaver Exchange Infrastructure.

² For the first of such services from mySAP ERP 2005, visit the Enterprise Services Workplace at <https://www.sdn.sap.com/irj/sdn/developerareas/esa/esapreview>.

IT Scenario: Enabling Enterprise Services		
When You Want to...	Variant	Description
Build new applications or parts of applications that can be easily exposed as reusable Web services.	Service-oriented development	This scenario variant enables customers to design, implement, and build enterprise services based on the Enterprise Services Repository (ESR).
Simply expose existing functionality for standards-based, online consumption.	Point-to-point, services-based integration	This scenario variant provides for exposing ABAP and Java functionality on SAP NetWeaver Application Server as Web services via standard protocols. The service provisioning is done without mediation by an integration broker.
Service-enable legacy applications that lack the underlying technology to make themselves available as services directly. Instead, you want to use a brokering methodology.	Brokered services-based integration	This scenario variant enables you to run a Web service scenario using the integration broker as the integration/communication engine between the service consumer and the service provider. This especially makes sense if the integration broker's additional capabilities will be used.
Figure 1	Three Basic Approaches for Enabling Enterprise Services	

The SAP entities created and maintained in the repository are based on a governance process that ensures quality and supports reuse. The repository is also open for customers to supplement the SAP service descriptions with custom services relevant to their own business needs and environment. So, whether you're taking an *outside-in* or *inside-out* approach to service-oriented development, both the SAP-provided and the local services can be accessed centrally in an identical manner.

An *outside-in approach* is best for service-enabling newly built applications. It starts by creating the interface design in the Enterprise Services Repository, using the Data Type Editor to define your data parameters — customer number, for example, using the credit limit check scenario. After defining a corresponding message interface in the repository and reporting the underlying data types, you automatically get a WSDL document for that interface. This WSDL document is the basis for generating the programming stubs (proxies) in the desired implementation language. These services can be either synchronous or asynchronous services.³ SAP Business Suite architects use exactly this methodology when developing new applications (see sidebar).

The *inside-out approach* starts with existing application functionality. It quickly gets some services that you need up and running, since you can wrap existing functionality using the Web services standards (such as WSDL or SOAP) and then import the interface description into the Enterprise Services Repository. Services created from the inside out follow a synchronous message response pattern.

Note also that the Enterprise Services Repository is designed for much more complex scenarios than the credit

³ *Synchronous* means that the process waits until a response comes back from the request message; *asynchronous* means that the sender does not wait for the response but continues with the remaining processing steps.

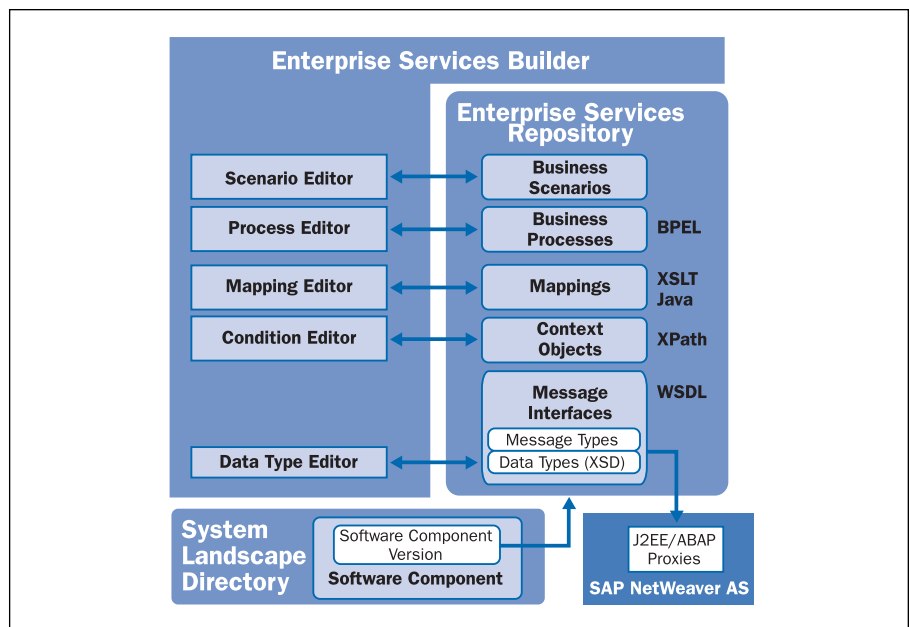


Figure 2 Snapshot of the Enterprise Services Builder and Enterprise Services Repository

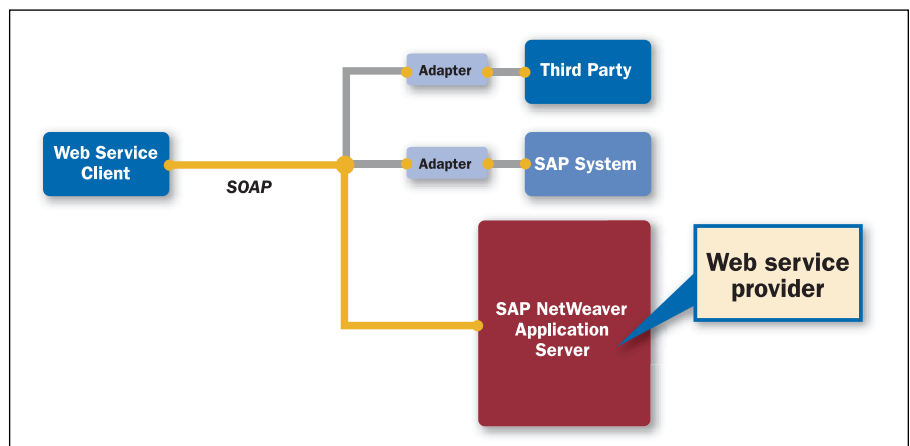


Figure 3 Point-to-Point, Services-Based Integration

limit check example we've seen here. For example, if you want to check a customer's history or see what kinds of transactions they've executed in the past, you can also have an orchestrated business process in between or as part of your service call. The repository is designed to accommodate business processes of all complexities.

Rendering Existing Functionality as Web Services

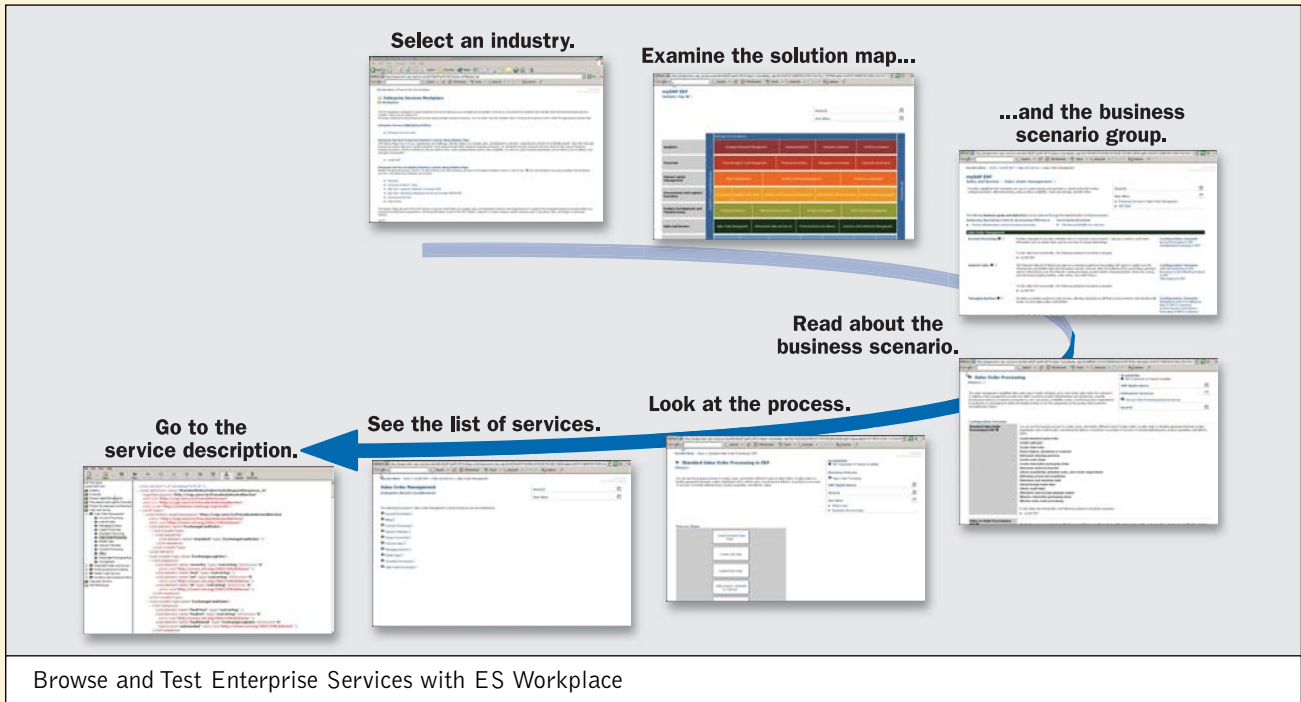
The *point-to-point, services-based integration* variant is about exposing existing ABAP and Java functionality as Web

services once they are created using the methods described in the previous section. With this approach, services are called in a point-to-point mode via a SOAP call (see **Figure 3**). It allows for synchronous communication in either ABAP or Java runtime environments. An integration broker is not used as a negotiator between a Web service client and the Web service provider.

This scenario variant is designed to set up how the service will be provided and consumed. In service consumption, the consuming application calls the proxy layer to compose and send a

See What Enterprise Services Actually Look Like

To preview the enterprise services being built within SAP using outside-in development (available with the release of mySAP ERP 2005), sign on to the Enterprise Services Workplace (ES Workplace) on the SAP Developer Network (<https://www.sdn.sap.com/irj/sdn/developerareas/esa/esapreview>). The new ES Workplace lets you browse and test enterprise services. You can drill down from industry to solution map to business scenario, all the way to the service and WSDL (from the Enterprise Services Repository) for that service, as shown in the figure below. Customers, especially on the business side, can really see what enterprise services SAP has to offer for their particular industry or application area.



message based on the message exchange pattern of the Web service description. In synchronous scenarios, the service consumer receives a response message. This runtime process is used to format and send a request message to the service provider. Because the client is generated from the WSDL service description (contract), the message will conform to the message exchange pattern of the requested service.

Service providing takes care of receiving and processing the request message from the service consumer and calls the proxy to execute the server functionality, which is nothing but the business functionality exposed by the service. Companies can enhance their solutions by providing them as services using ABAP and Java Web services, which

are provided within SAP NetWeaver using the *Web Services Framework*.

The Web Services Framework is used by IT architects and developers who are actually developing the service. The framework consists of:

- The development environments for both the ABAP and the J2EE engines
- Tools that support UDDI registration
- A distributed, interoperable SOAP runtime (ABAP and J2EE engine)

The framework's wizard-based interface (see **Figure 4** on the next page) guides developers through the service creation process — choosing an endpoint, entering the service name, choosing operations, configuring the service, and so on.

Service-Enabling Legacy Applications

The *brokered services-based integration* approach enhances simple point-to-point Web services communication. Using this approach, the consumer of a service sends a message to an integration broker, which then processes the message and forwards it on. The message receiver does not have to be a Web service provider, and it can also be a system that does not support the SOAP protocol.

This brokered integration of Web services enables multiple services to be orchestrated within a business process. The result of the orchestration is what is presented to the initial Web service call. In this way, it is possible to provide a virtual interface for those applications — legacy applications, for example — that

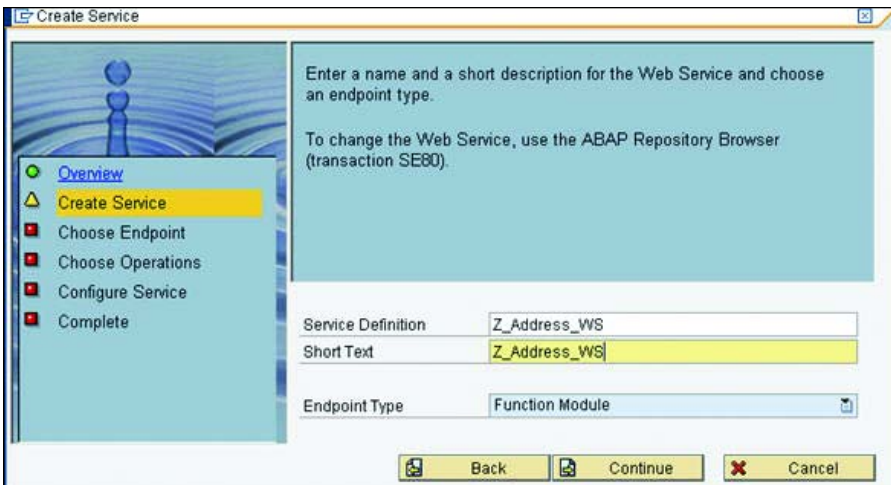


Figure 4 Service-Providing in SAP NetWeaver Using the Web Services Framework

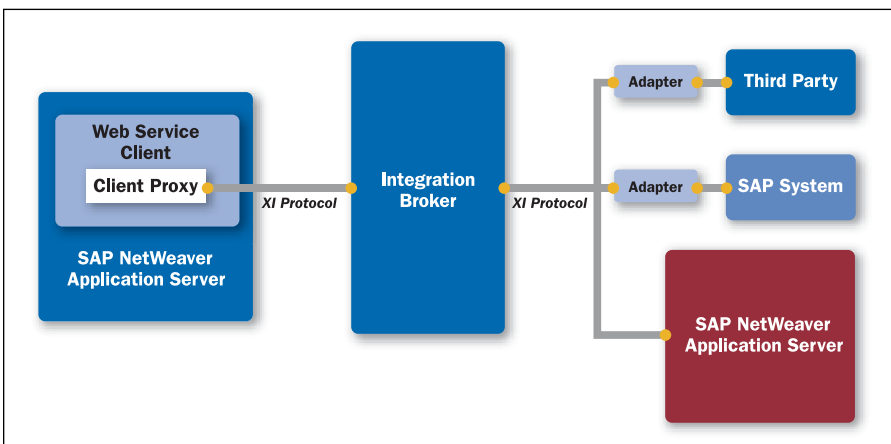


Figure 5 Service Brokering Using SAP NetWeaver's Integration Broker

with SAP NetWeaver 2004s, business and IT teams can follow three simple approaches to create, invoke, and consume Web services according to their business process needs:

- With service-oriented development, customers can rapidly develop service-based applications, providing for greater agility in meeting customer and partner demand.
- With point-to-point, services-based integration, customers can expose proprietary business logic to service clients inside and outside the enterprise boundary in a standards-based way.
- With brokered services-based integration, customers can provide a virtual interface to those backend applications that are not equipped with the capabilities to make their specific business functions available as Web services.

SAP NetWeaver's Enterprise Services Repository allows IT departments to support easy enterprise services development, reuse elements within the repository, and enforce strict governance models for service-oriented development — all from a single, central environment.

For more information on enabling enterprise services, please visit <http://service.sap.com/it-scenarios> and www.sap.com/solutions/businessmaps. Remember to take a look at the ES Workplace at <https://www.sdn.sap.com/irj/sdn/developerareas/esa/esapreview> to see what services SAP offers and begin incorporating them into your enterprise's processes. ■

Sindhu Gangadharan is a Senior Product Manager at SAP and has been lending her multiple talents to the SAP NetWeaver platform in recent years. She has been with SAP for the last seven years, working on various SAP integration technologies. Sindhu specializes in process integration and the enterprise SOA blueprint. You can reach her at sindhu.gangadharan@sap.com.

are not able to make their application functionality available as a standard Web service (see **Figure 5**).

This variant can also be applied when point-to-point Web services communication needs to be enhanced with the addition of routing and mapping options — for instance, if specific service calls need to be routed to a particular receiver system. These additional capabilities are provided by the integration broker:

- *Routing:* The message can be dynamically routed to different receivers depending, for example, on the payload.
- *Mapping:* The request/response message can be mapped to the format of other service interfaces

and then sent to older versions of service providers, for example.

- *Business process management:* You can build in complex processes, such as also checking customer purchase history when conducting their credit limit check. These processes can be designed and executed on the integration broker, handled by the Business Process Engine.

Conclusion

To achieve a competitive business strategy that is fueled by innovation and doesn't break existing processes, companies must start taking their first steps toward enterprise SOA now, if they haven't already. Using the new Enterprise Services Repository available