SAPinsider This article appeared in the Jan ▪ Feb ▪ Mar 2008 issue of *SAP Insider* and appears here with permission from the publisher, Wellesley Information Services (WIS), **www.WISpubs.com**.

Service Provisioning in ABAP

A Step-by-Step Guide to Creating a Service with SAP NetWeaver PI 7.1

As companies begin to understand more about service-oriented architecture (SOA) and the benefits it provides to the business – namely, its ability to reuse existing IT assets and nimbly adapt technology to evolving business needs and processes – IT architects are now looking at how to best create and provision services. Companies with core ABAP development groups are especially seeking a concrete service development roadmap. They want to find out how they can use an individual programming language in an SOA environment, which emphasizes openness and standards.

First and foremost, it is critical that IT teams understand SOA's basic principles – that it's a means of creating loosely coupled services that allow for separating the description of the service, its actual implementation, and its binding. This separation is, after all, the pillar of SOA's flexibility. But to take the next step and actually create services, IT architects must understand how a typical service development cycle happens.

With SAP NetWeaver Process Integration (SAP NetWeaver PI, formerly called SAP NetWeaver XI) 7.1, ABAP teams will find all the capabilities they need to develop services – starting with defining a service in the Enterprise Services Repository (ESR), implementing it in ABAP,[1] configuring its service endpoints, and publishing it in an easily accessible services registry. This article will walk you through these steps.

It's important to note that the majority of services SAP has delivered through enhancement packages are implemented in ABAP. This comes as no surprise, really, as ABAP-based development is renowned for its power, simplicity, and efficiency (see sidebar).

Develop a Service in ABAP in 5 Steps

Perhaps the best way to illustrate the process of providing services is to walk through a typical business scenario. Consider an employee working with SAP Supplier Relationship Management (SAP SRM) who wants to create new vendor data within the SRM system as well as search for existing vendors. Since this functionality is also frequently used

[1] Implementing the service in Java is also a possibility. For the purposes of this article, though, we'll focus on ABAP.

Sindhu Gangadharan (sindhu.gangadharan@sap.com) is a Senior Product Manager at SAP and most recently has been lending her talents to the SAP NetWeaver platform. She has been with SAP for the last eight years, working on various SAP integration technologies. Sindhu specializes in SAP NetWeaver SOA technologies and SOA middleware, among other qualifications.

Dr. Susanne Rothaug (susanne.rothaug@sap.com) is a Product Manager on the SAP NetWeaver SOA middleware team. She joined SAP six years ago and has worked on various areas within SAP NetWeaver product management. Currently Susanne focuses on process integration and SOA topics.

Why ABAP and SOA Are a Great Fit

It's true that service-oriented architecture, by nature, is intended to make the programming language used irrelevant. SOA is designed to work with any open standard, so programmers should be able to use their environment of choice when developing services.

Those proficient in ABAP, then, can continue to use it in an SOA environment. In fact, ABAP's a natural fit for SOA, for several reasons:

- ABAP is optimized for business applications to allow high levels of scalability in transaction handling and data size.
- ABAP offers powerful transport and change-management features supporting the development of complex applications even in large development teams.
- The ABAP Development Workbench is a powerful and integrated development environment.

System Prerequisites

The business example highlighted in this article involves the following system prerequisites:

- SAP NetWeaver Process Integration (SAP NetWeaver PI) 7.1
- Vendor management functionality available in an application like SAP Supplier Relationship Management (SAP SRM)

Please note that the additional vendor management functionality requirement is specific to the example highlighted within this article.

within other applications like SAP ERP or SAP Customer Relationship Management (SAP CRM), the employee wants to implement this functionality as an enterprise service. Developing the service involves five steps:

1. Model the service interface in the ESR
2. Generate the provider proxy
3. Implement the service's operations
4. Configure the endpoint in SAP NetWeaver Administrator
5. Publish the service definition to the Services Registry

We're not simply wrapping functionality with services standards to make it available as a Web service. Rather, we are centrally defining exactly what criteria the service needs to fill before we implement the operations.

Importantly, these steps incorporate SAP's best-practice recommendation for a "contract first" way of developing services. The first step in this process involves defining the service interface, or contract, in a central repository – the ESR, in this case. Only then do we move to implementing the actual functionality.

In other words, we're not simply wrapping functionality with services standards to make it available as a Web service. Rather, we are starting from a business perspective and centrally defining exactly what criteria the service needs to fill – determining what request and response messages are necessary to fulfill the business requirement – before we implement the operations.

Step #1: Model the Service Interface in the ESR

Once you determine the granularity of the service you want to create – that is, how broad your required piece of functionality must be in order to address your business need – you can create the service interface for managing vendor data.[2]

Because the service is to be used for two tasks – creating and searching for vendor data – we design the interface with two operations: an asynchronous one for creating new vendor data and a synchronous one for searching by vendor ID.[3] This ability to create service definitions with multiple operations is new as of SAP NetWeaver PI 7.1.

We'll design the interface in the ESR of SAP NetWeaver PI 7.1, which is the central repository and modeling environment for enterprise service objects and process and service definitions.[4] The ESR stores data that is relevant for configuring, installing, and deploying services, but does not provide runtime or platform-specific data. The following designtime objects, then, need to be created: data types, message types, and the service interface and its operations. When available as a service, these objects can be reused – so you won't have to re-create them.

In our example, we must start by creating the inbound, stateless service interface[5] – which we'll name ManageVendor – with two operations:

- The asynchronous **CreateVendor** operation references the already existing message types CreateVendor_MT for the request message and CreateVendor_FMT as the standard fault message type, which is used for handling any application-specific errors that may occur on the inbound side. This fault message, in turn, references the data types Vendor_DT and ExchangeFaultData.
- The synchronous **SearchVendor** operation is based on the request and response message types SearchVendor_ReqMT and SearchVendor_ResMT and also the fault message type. The underlying data types are Vendor_Req and Vendor_DT.

Figure 1 shows the service interface as it is created in the ESR. The design of the service interface corresponds to a generated Web Services Description Language (WSDL) file. **Figure 2** shows an excerpt of the WSDL file that corresponds to the interface in

[2] We recommend that ABAP teams follow SAP's process component modeling methodology to define the service itself. Please note that this article focuses on the technical aspects of defining services rather than the methodology used.

[3] With *asynchronous* services, the client invokes the service but does not wait for the response; the client can continue with other processing. With *synchronous* services, the client invokes a service and then waits for a response.

[4] Please note that this is only one of several deployment options for the Enterprise Services Repository. The ESR can also come with SAP NetWeaver Composition Environment (SAP NetWeaver CE).

[5] A *stateless* interface is one that does not retain knowledge of any data exchanged through the service. Stateless interfaces are preferable for SOA, because they can be reused by many service consumers.

Figure 1. Here you can see the two operations, CreateVendor and SearchVendor.

Once you finish creating these objects, change lists are automatically created in the ESR. These change lists then must be activated; click the change list tab and choose *Activate*. Then proceed to the back-end system for proxy generation.

Step #2: Generate the Provider Proxy

The objects we just created in the ESR are language independent. So to create language-specific objects, we will need to create a proxy object. In this next step, we'll go from the metadata definition to the language-specific definition.

ESR objects are mapped to platform-specific proxy objects. Provider proxies can only be generated for objects in the ESR – the inbound ManageVendor service interface, in our case. We will create and implement the proxies in the ABAP back-end system, in which the service will be running.

To generate the provider proxy for ManageVendor, use the following process steps:

- Go to transaction SE80 in the ABAP Development Workbench and start the Enterprise Services Browser. The browser will connect to the ESR and display its software component versions, including the one in which the example ManageVendor service interface was created.
- Expand the respective nodes of the component versions and the namespace, open the context menu of ManageVendor, and choose *Create Proxy*.
- In the subsequent pop-up window, specify a package and a prefix, and choose *Enter*.
- Based on the prefix that you specify, valid ABAP names are proposed. You can change the proposed names as required.
- The proxy is then generated automatically.

The proxy's *Properties* tab, which is displayed after the generation completes, gives information about the generated proxy interface, such as the implementing proxy class, as well as the generated service definition (see **Figure 3** on the next page).

When the proxy is generated, all the underlying objects, as well as the WSDL file of the Web service based on that proxy, are generated automatically. This means that the service creation process is significantly simpler compared to SAP NetWeaver 7.0; the extra step of running the Service Definition

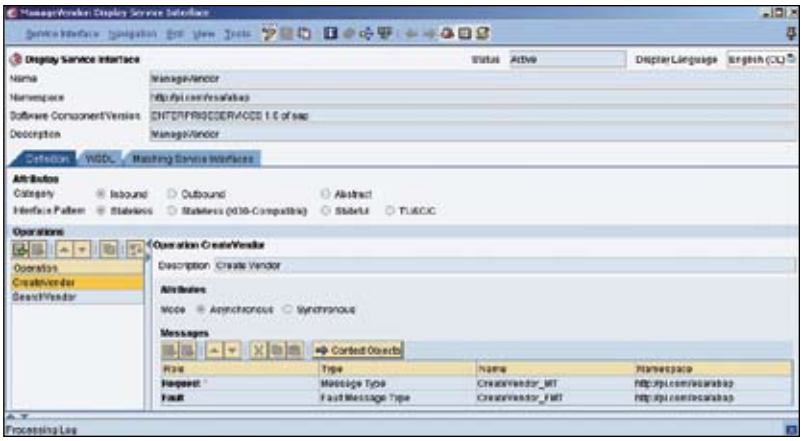


FIGURE 1 ▲ Creating the ManageVendor service interface in the Enterprise Services Repository

```
<wsdl:definitions xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
xmlns:p1="http://pi.com/esa/abap" name="ManageVendor" targetNamespace=
"http://pi.com/esa/abap">
  <wsdl:documentation>
  ManageVendor
  </wsdl:documentation>
  <wsdl:types>
    <xsd:schema xmlns:xsd=»http://www.w3.org/2001/XMLSchema» xmlns=
»http://pi.com/esa/abap» targetNamespace=»http://pi.com/esa/abap»>
      <xsd:element name="SearchVendor_FMT">
        <xsd:complexType>
          <xsd:sequence>
            <xsd:element name="standard" type="ExchangeFaultData" />
          </xsd:sequence>
        </xsd:complexType>
      </xsd:element>
      <xsd:element name="SearchVendor_ReqMT" type="Vendor_Req" />
      <xsd:element name="SearchVendor_ResMT" type="Vendor_DT" />
      <xsd:element name="CreateVendor_MT" type="Vendor_DT" />
      <xsd:complexType name="Vendor_Req">
        <xsd:sequence>
          <xsd:element name="VendorNumber" type="xsd:string" />
          <xsd:element name="Country" type="xsd:string" />
        </xsd:sequence>
      </xsd:complexType>
      <xsd:complexType name="Vendor_DT">
        <xsd:sequence>
          <xsd:element name="VendorNumber" type="xsd:string" />
          <xsd:element name="LastName" type="xsd:string" />
          <xsd:element name="SearchTerm" type="xsd:string" />
          <xsd:element name="Currency" type="xsd:string" />
          <xsd:element name="Address" type="Address_DT" />
        </xsd:sequence>
      </xsd:complexType>
```

The SearchVendor operation

The CreateVendor operation

FIGURE 2 ▲ An excerpt of the WSDL definition that corresponds to the service interface shown in Figure 1

Wizard to turn the provider proxy into a Web service is no longer necessary.

The ABAP objects, though, including the class, interface, and data dictionary (DDIC) type, are not created in the system until you activate the proxy. The ABAP class of a service provider uses an ABAP interface and contains the implementation of the service provider methods. A service provider class has one method for each operation modeled in the ESR. So for an application to use a proxy, the service provider methods will need to be filled with coding.

FIGURE 3 ▼ The *Properties* tab displays information about the generated proxy interface and the service interface

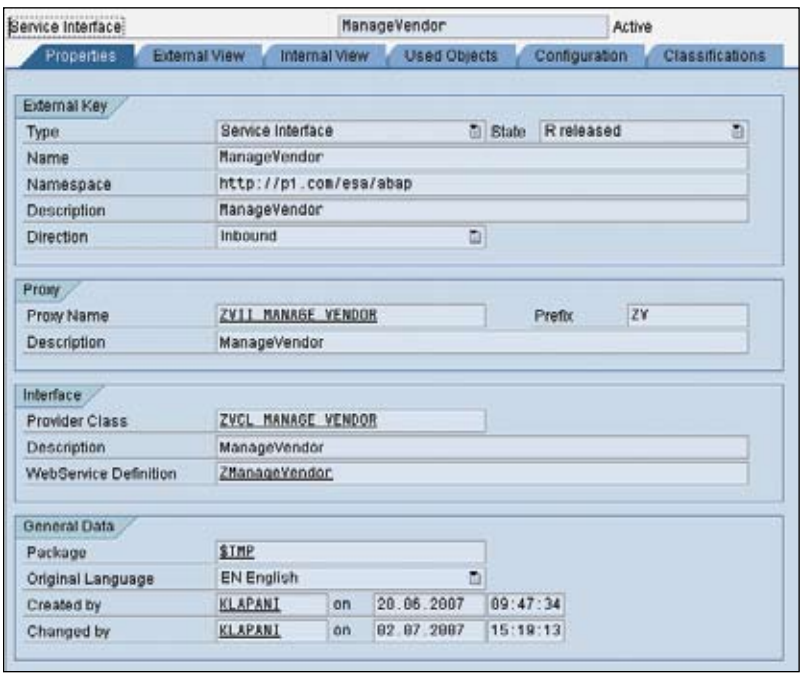
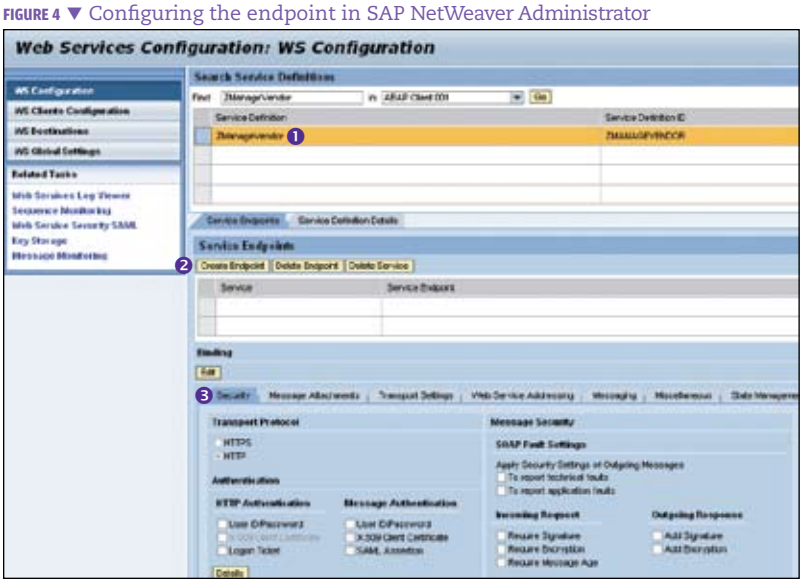


FIGURE 4 ▼ Configuring the endpoint in SAP NetWeaver Administrator



Step #3: Implement the Service's Operations

By double-clicking the interface provider class ZVCL_MANAGE_VENDOR, which is visible under the *Properties* tab in Figure 3, you can display the two operations, ManageVendor and SearchVendor. Then, double-click the method name ZXXII_MANAGE_VENDOR_XX~CREATE_VENDOR to open the editor window. Here you will be asked to insert the implementation of the operation. Provide the coding required for your operation, then save and activate it.

Note that the coding could be a call to existing functionality, such as a BAPI or a function module. This would allow IT architects to use stable and mature functionality in a service-based way – without re-creating the actual implementation.

Repeat this step for the second operation. The proxy now contains the business functionality required to run the service. It does not, however, have an endpoint configuration yet.

Step #4: Configure the Endpoint in SAP NetWeaver Administrator

We'll now move to the WS Configuration section of SAP NetWeaver Administrator to configure the endpoint (see **Figure 4**). First, select the service definition ZManageVendor ❶ that was automatically created during proxy generation, and choose *Create Endpoint* ❷. An endpoint contains a single runtime configuration. You can create different endpoints to define different runtime behaviors for the same service definition. Under the Security tab ❸ you can also assign security settings, such as transport security and authentication, to the Web service.[6]

After saving your settings, the runtime configuration with a port type will be available on the back-end system and can be integrated into an application. Remember, though, that so far this information is only available in the physical ABAP system where the service is running. For a consuming application to be able to discover the service, the service needs to be made available in a more easily accessible place.

Step #5: Publish the Service Definition to the Services Registry

The Services Registry is a UDDI v3-based registry that contains all the information required to run a

[6] See the Security Strategies column by Yonko Yonchev – "Do You Know Exactly Who's Looking at Your Business Data?" – in this January-March 2008 issue of *SAP Insider* (www.SAPinsideronline.com) to learn how to assign security settings for Web services.

We've Covered the Provider Side; Don't Forget the Consumer Side

For the consumer side, SAP provides a number of options — from creating an ABAP consumer proxy that is called by a client program (an option that is well supported in the development environment) to using the new SAP NetWeaver Composition Environment for creating a consuming application in a lean Java EE 5-based environment. SAP can accommodate the needs of the most diverse customer scenarios. Look for an upcoming SAP NetWeaver Unleashed column in *SAP Insider* to discuss the consumer side in more detail.

service; it's an accessible place for discovering services available in your landscape.

To publish your newly created service, go to transaction WSPUBLISH in your ABAP system and select the service definition you want to publish. Simply select the registry to which you want to publish the service, and then choose *OK*. The service will then be listed in the registry and can be discovered by potential consumers (see sidebar above).

Conclusion

SOA discussions need not focus only on Java and open standards; with SAP NetWeaver PI 7.1, providing enterprise services in ABAP is not only possible, but simpler than before. In five steps, IT architects can provide services in ABAP and make them available in a central, easily accessible location.

This five-step roadmap includes SAP's best-practice approach of first defining the contract, or business criteria, and only then implementing the actual functionality. Using this method, you are targeting reuse; since the service will not be completely coupled with only your specific application, you (and others) can reuse it in several other processes as well. For more information, please visit **www.sdn.sap.com/irj/sdn/nw-soa.** ■

Additional Resources...

...from SAPinsider

- "Take Note! What Should — and Should Not — Be Running on the Same Systems in Your SAP Environment? Guiding Principles for Reshaping Your SAP ERP Landscape" by Dr. Franz-Josef Fritz (*SAP Insider*, October-December 2007, **www.SAPinsideronline.com**)
- "SAP NetWeaver Unleashed: Three Basic Approaches and One Central Repository to Service-Enable Your Enterprise Applications" by Sindhu Gangadharan (*SAP Insider*, July-September 2006, **www.SAPinsideronline.com**)
- The **SAP NetWeaver/BI and Portals 2008** conference in Orlando, March 26-28, 2008, for answers to your most pressing questions about enterprise services and how to build, deploy, and consume them (**www.sapnetweaver2008.com**)