Develop composite applications with SAP NetWeaver Composition Environment 7.1

Part 1 — Enterprise services and their usage within a composite application

by Volker Stiehl

Volker Stiehl
Product Manager,
SAP AG

Volker Stiehl received a degree in computer science from the Friedrich-Alexander-University of Erlangen-Nürnberg, Germany. He was a consultant for distributed J2EE-based business solutions and integration architectures at Siemens for 12 years. In 2004 he joined SAP's product management team for a composite application development toolset, where he's also responsible for methodologies and best practices. He holds workshops in composite applications and is a regular speaker at conferences, such as SAPPHIRE, SAP TechEd, and JavaOne. You may reach him at volker.stiehl@sap.com.

Today's business world changes so rapidly that companies must constantly look for improvements in those business processes that bring them a competitive advantage. Typically, you won't be able to find these improvements in your back-end systems, which are made up of stable, basic business processes (e.g., sales order or purchase order processing, supplier invoice processing, and payment processing). Standard products such as SAP Business Suite cover these products fairly well. However, every industry is different and each company has processes that are unique to its particular business. These are the areas in which innovation happens, the areas where companies can differentiate themselves from one another.

In contrast to stable processes, innovative processes are highly dynamic, not standardized, change often, and fill gaps that standard products cannot cover because of a particular company's area of expertise or its unique business logic. In addition, innovative processes should reuse existing functionality in the back-end systems and only add functionality where standard products fall short. Innovative processes are often highly collaborative, so you should look for processes that involve many different process roles.

Composite applications, defined as packaged applications that sit on top of existing enterprise solutions reusing their functionality to form new collaborative business processes, have played an important role in SAP's enterprise service-oriented architecture (enterprise SOA) strategy. Their importance was increased further by an announcement made during the 2007 JavaOne conference, in which the new SAP NetWeaver Composition Environment (SAP NetWeaver CE) was presented to customers and partners for the first time as the design and runtime environment for composite applications. Since then, interest in this kind of application and its development has grown dramatically.

SAP NetWeaver CE was introduced to support the development of composite applications in a model-driven way. In contrast to SAP NetWeaver itself, which was delivered with the components SAP NetWeaver Business Intelligence (SAP NetWeaver BI) and SAP NetWeaver Process Integration (SAP NetWeaver PI), among others, SAP NetWeaver CE has a lightweight layout. It doesn't include the various components, significantly reducing the download and installation times. Another advantage associated with this lightweight approach is the shorter innovation cycle. Because you don't need to synchronize the complete SAP NetWeaver package, you can update SAP NetWeaver CE and adapt it to the latest versions of the integrated standards in a considerably shorter amount of time, keeping developers more up to date.

Why would you want composite applications? How do you build them? How can you benefit from this new method of application generation? If you identify with any of these questions, then this three-article series is for you. It introduces you to composite applications, their characteristics, their architecture, the challenges you typically face during their development, and how you can address them with the SAP NetWeaver CE (Service Release 3) toolset.

To gain more insight into these new challenges, this series shows you how to develop a composite application from scratch. By following this approach, you'll not only learn how to use the different tools individually, but also how to combine the tools in an integrated fashion to create more powerful solutions.

This first installment begins with an overview of composite applications and SAP NetWeaver CE, followed by a short introduction to a simple collaborative scenario: a complaint management scenario. This setting will help you to identify the different pieces that make up a composite application: business objects, services (application, enterprise, or Web), user interfaces (UIs), and processes. For each of these items, I'll introduce the appropriate tool: SAP Composite Application Framework (CAF) for business objects and services, SAP NetWeaver Visual Composer for UIs, and SAP Guided Procedures (GPs) for processes.

Then, from the perspective of a composite application developer, I will explain how to reuse existing functionality with an example that shows you how to consume the real-world enterprise services that SAP Enterprise Services Workplace (ES Workplace) provides.[1] Finally, I'll conclude this article with details on how to simplify the enterprise service's interfaces with CAF so that they become manageable for service consumers.

The second installment, which will appear in the next issue of this publication, continues with building the UI-related parts of a composite application with Visual Composer. It also explains some of the advanced features that you can use in developing composite applications.

The final installment, which will follow Part 2, will show you how to model the process for the composite application. That article delves more deeply into GPs. It will also show you how to benefit from the composite application's architecture by replacing service implementations without affecting the composite itself. Throughout this series, I will discuss tips and tricks, potential pitfalls, and recommendations to make your first composite application a success.

Whether you're a Java developer or a business process expert looking for detailed information about composite applications and their development processes, this series will give you a solid foundation from which to begin your own journey into the world of composite applications.

Introducing composite applications

Key to a company's success is its quick reaction to changing market conditions and implementing innovative processes in a short timeframe to beat the competition. Therefore, a short "time to market" is paramount. If a process is successful, you can rest

[1] You will find the ES Workplace on SDN at https://www.sdn.sap.com/irj/sdn/esworkplace. Logon credentials are required to access information in SDN.

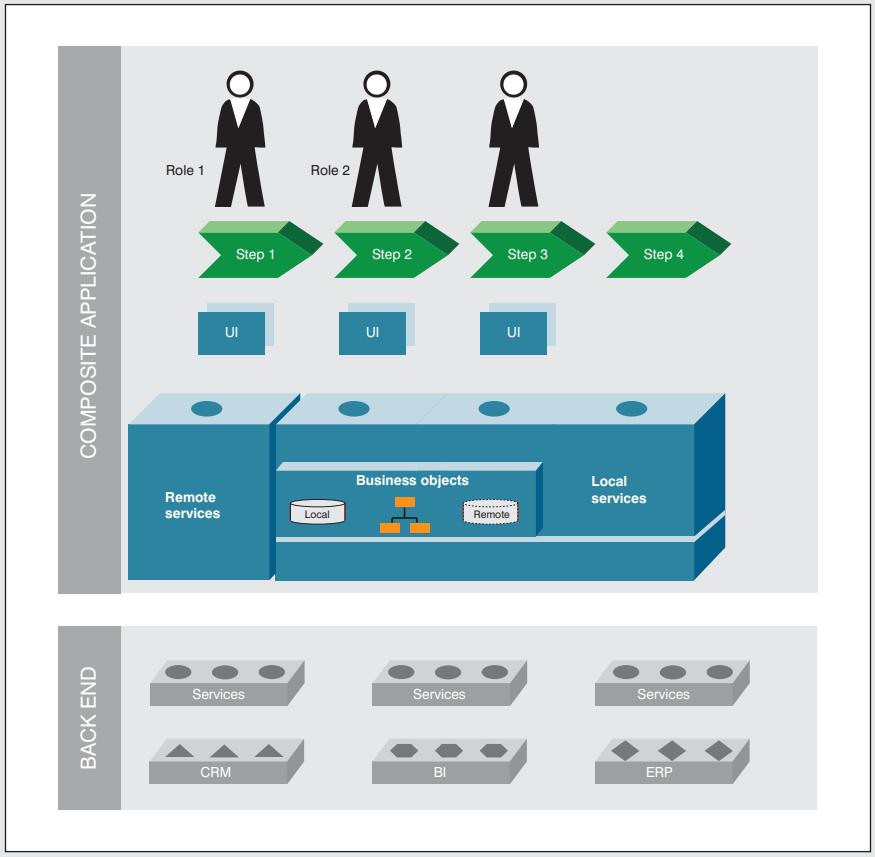


Figure 1 Architecture of a composite application

assured that the competition will adopt it too, forcing you to start the innovation cycle again. This cycle — improve process, competition adopts it, look for further improvements — is typical for innovative processes. That's why you need the architecture and toolset that support these requirements best, bringing companies the flexibility to adapt or innovate processes depending on the speed of their business. SAP's answer to these challenges is composite applications and SAP NetWeaver CE.

The architecture of a composite application, as shown in **Figure 1**, follows a layered approach. Composites sit on back-end systems that provide basic business functionality. The back-end systems are service-enabled, meaning that they deliver their functionality by standard means (e.g., Web services). However, it's essential to understand that service-enabling back-end functionality doesn't make it part of the composite application.

One of the key characteristics of a composite application is that it's loosely coupled with its back-end systems; that's why you see the gap between the composite application and the back end. It's intentional. You don't want dependencies between a composite and its back-end system. Composite applications run only against well-defined interfaces with well-defined business semantics. How you implement the business logic (or semantics) isn't important to a composite; what counts is that the implementation doesn't modify the interface. Understanding this distinction should help you envision how composites work in real time. The advantage of this separation is

that your system becomes more flexible when you plug in an implementation for functionality as it's needed, and your composite has its own life cycle, independent of your back-end systems' releases.

In layering a composite application, the business objects and services layer is at the bottom, followed by the UI layer and the process layer (steps and roles). Composite applications work on business objects, such as order, supplier, customer, product, and invoice. Some business objects are already represented in one of the existing back-end systems; others are unique to the composite and need to be persisted within the composite itself. That's why you need to distinguish between local and remote persisted business objects. However, to the upper layers of the composite, this differentiation doesn't matter and should be hidden from business object consumers.

Only stable interfaces containing the business object's lifecycle methods, such as create, read, update, delete, and find, are exposed to the consumers. Based on these interfaces, you can implement your own business logic on top of these business objects. The business objects are only data containers. You can create, read, update, delete, and find them. That's all. Business logic is built on top of the business objects in the form of CAF application services. These application services are developed in Java and directly use the Java interfaces of the business objects, which are then exposed as local services to the next layer.

If you want to reuse any existing functionalities, you can call them via remote services. SAP recommends that you wrap external services (those coming from other sources) as remote services and that you don't consume them directly in higher layers. This means that a UI shouldn't consume a Web service directly, bypassing a composite application's service layer. This would make the UI immediately dependent on the consumed Web service, exactly what you want to avoid. The same is true for the process layer. If you follow this remote services approach, you can achieve the flexibility you need to replace service implementations later.

Above the business objects and services layer is the UI layer. Based on either remote or local services, you can model UIs to represent one step in the processes modeled in the process layer, which can consume services as well. The same rule applies here: The process layer should consume services only from the business objects and services layer to gain the highest flexibility and independence from the back-end systems.

On the process layer you define process flow, the roles associated with each step telling the process framework who must contribute to this process and when, and the data flow from step to step. To make the data transfer between steps possible, it's important that UI technologies work closely with the process layer; they must indicate:

- Which data they need as input from the process
- What data they will return to the process
- The point in time when the user has finished his work on the UI so the process can proceed with the next step

For services that the process layer calls, this information is available in Web Services Description Language (WSDL) format. By default, WSDL is provided for service calls. It's SAP's recommended approach to use Web services only. However, GPs also allow you to plug in other background steps, such as an ABAP function module called via RFC or a simple Java class. In these cases, the interface information is not based on WSDL.

You don't need any additional programming to make collaboration with the process layer possible. You should also be aware of two important characteristics of a composite application (relating to its back-end connection):

- Stateless service calls of back-end services: You don't keep any state information in the back-end system between service calls.
- Non-invasiveness: You do *not* modify back-end systems. Composite applications use external functionality as-is and add necessary functionality outside the back-end systems.

By following this layered structure, you get an application that is flexible to adaptation and allows accelerated innovation on an existing landscape.

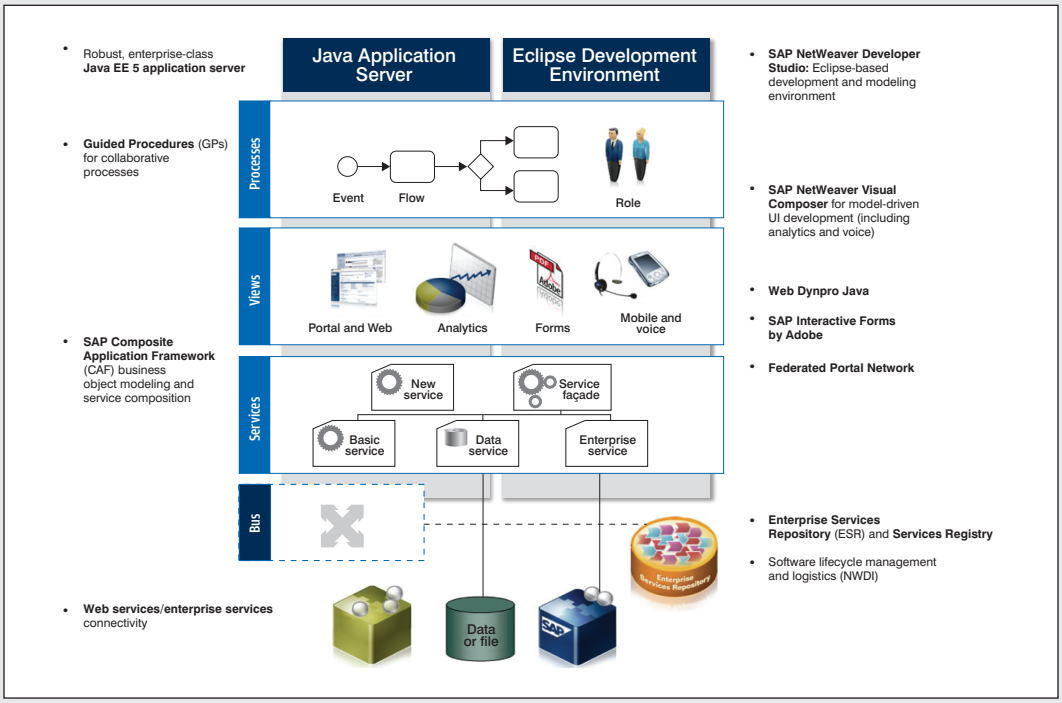


Figure 2 SAP NetWeaver CE architecture

However, you have to implement such an application at some time and the question is: Is there an available tool that can help you translate your process idea into an executable solution? It's SAP NetWeaver CE.

SAP NetWeaver Composition Environment

Developer efficiency and a short time-to-market are essential to achieve the benefits that composite applications promise. That's why the idea of model-driven development has influenced tool design so heavily. Model-driven development means that you can generate executable code using models. The benefits includee higher developer productivity, fewer and less serious errors, and a stronger concentration on business-related logic only. The generated code performs all the cumbersome, time-consuming, and error-prone coding for authorization, authentication, logging, database-locking, and transaction-handling — the boilerplate code.

New developers come up to speed much sooner in such an environment, than if they have to learn and master a complex programming language. The architecture of a composite application (**Figure 1**) uses model-driven development on all layers; it supports modeling for the processes, the UIs, and the business objects. Programming skills are needed only if you need to implement composite-specific business logic and provide it as a service.

How does SAP NetWeaver CE implement these requirements? The first thing you'll recognize (as shown in **Figure 2**) is the Java EE 5-compliant application server on which SAP NetWeaver CE is based. It represents the runtime environment and supports all relevant Java EE 5 standards, such as Enterprise Java Beans (EJB) 3.0, JavaServer Faces (JSF) 1.2, JavaServer Pages (JSP) 2.1, Java Persistence API (JPA), Java API for XML Web Services (JAX-WS) 2.0, etc. Another part of SAP NetWeaver CE is the development and modeling environment SAP NetWeaver Developer Studio (NWDS), based on Eclipse 3.3.

Next, there is a common infrastructure for Web services, the lifecycle management for all the development artifacts that make up a composite application, as well as an SAP Enterprise Services Repository (ESR) and a Services Registry. The ESR contains, among other things, the service interfaces of all the enterprise services that SAP's business applications are shipping. You don't have to install all the SAP solutions to get this list. It's just a catalog of the enterprise services available, and you can use it while you're designing a composite application to determine which services exist and what gaps your own development has to fill. The ESR is not a "closed shop." You can add your enterprise services to it as well.

The Universal Description Discovery and Integration (UDDI)-based Services Registry, on the other hand, contains all the physically available services running in your landscape. You can actually call the services shown in the Services Registry, while the services described in the ESR may or may not be in your landscape. The bus is an enterprise services bus (ESB), a component coming in one of the next releases of SAP NetWeaver CE. The ESB is good for more complex scenarios with many connections to external systems. In those cases, you probably will want to have one location where you maintain routing, mapping, data format transformations, and connectivity, etc.

Since composite applications consist of different architectural layers, dedicated modeling tools support your modeling efforts in the following ways:

- On the bottom layer is CAF, which makes modeling your domain business objects easier. It is the central tool if you need to consume external services or compose new ones. CAF in SAP NetWeaver CE follows the same principles as SAP NetWeaver 7.0 (formerly 2004s), but it was completely rewritten on the business object side because of the availability of JPA for business objects. The APIs for the application services changed so it isn't possible to migrate SAP NetWeaver 7.0-based application services to SAP NetWeaver CE.
- On the UI layer you have choices to make, depending on your particular needs. Visual Composer is the tool of choice for "lightweight" UIs that contain little or no complicated logic. Visual Composer has been completely rewritten with a new architecture and a new concept. Therefore, it currently has less functionality than the SAP NetWeaver 7.0 version, especially when it comes to SAP NetWeaver BI. If you need more complex input verifications or tailored UI components that don't come out of the box with Visual Composer, you may benefit from using SAP's Web Dynpro. It offers you an appropriate development environment for highly sophisticated UIs.

  In addition to these two online technologies, SAP NetWeaver CE provides a solution for handling offline scenarios: SAP Interactive Forms by Adobe. It enables you to handle offline data when you don't have an online connection to your company (e.g., while traveling to a customer location or working offsite, perhaps as a service technician).

  SAP NetWeaver CE now provides support for the voice applications, with which you may already be familiar. You can model your own dialog flows in a graphical environment that also integrates with Visual Composer.

- On the process layer GPs help you to stitch together your UIs and services into new, innovative, collaborative business processes. GPs come with a browser-based development environment and a robust runtime engine to take care of process execution and the notification of participating users.

One final benefit of composite applications is the collaboration it fosters between IT and business. By working with graphical tools or tables, IT and business can work collaboratively on solutions for the first time, creating a completely new development experience for both parties. This approach ensures that IT will deliver what the business wants.

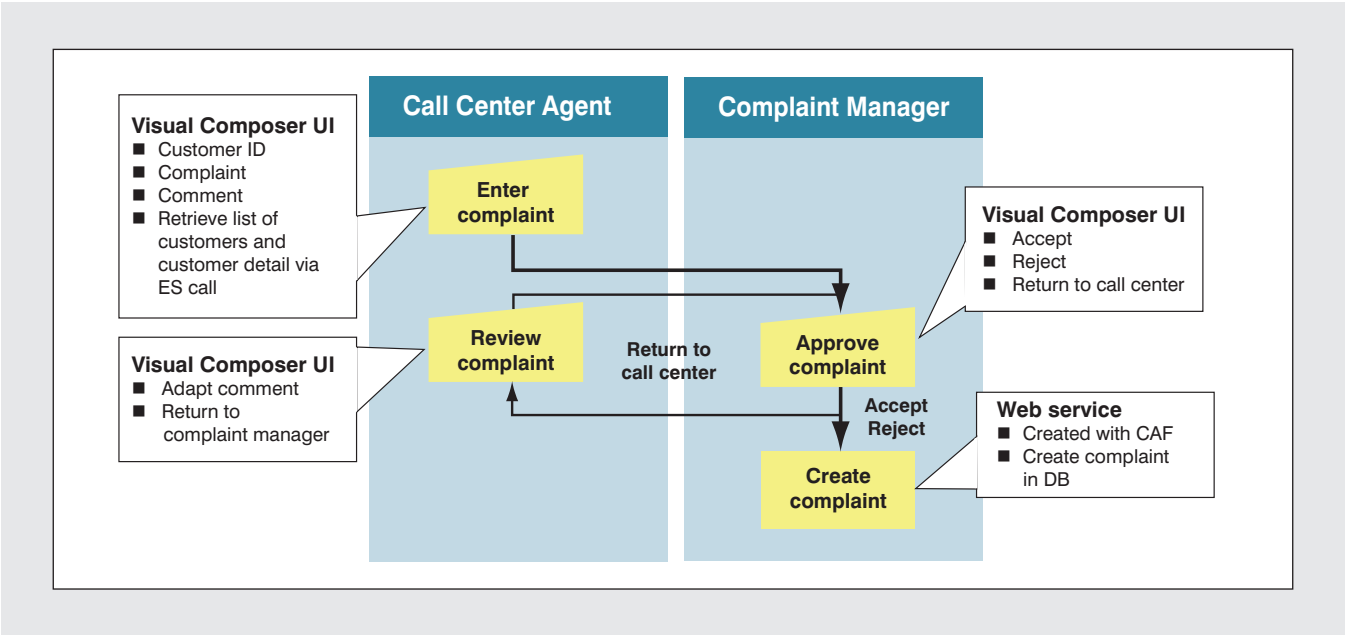


Figure 3 Process flow of the complaint management scenario

Complaint management scenario

Now, let's develop a composite application from scratch. Among the three parts of this article, I'll cover the complete development cycle of a composite application: from planning via development, deployment, and configuration to execution and testing. I won't go into all the details of the tools you need to build the composite. You can acquire the details of the tools themselves by reading the tool's documentation on SDN. Rather, it's more important that you understand the development methodology.

Developing composite applications is different from developing conventional applications. The distributed service-enabled world requires a complete sea change in thinking that is comparable to the one that occurred between procedural and object-oriented programming.

An innovative, new business process should always be the starting point for a new composite application. Remember, you're looking for collaborative processes in which different roles participate. So, let's look at the complaint management process to be implemented, as shown in **Figure 3**.

The idea is to increase customer satisfaction by shortening the response time required for complaints coming in through the company's call center. You also want to be able to analyze the collected data about complaints and use it to improve the company's quality process — and thereby the products themselves. This example process involves only two roles: the call center agent and the complaint manager.

The call center agent receives a call from a customer who has a problem with a shipment (e.g., the previous shipment was somehow damaged). As customers often don't have their customer IDs at hand when they call, the call center agent uses a search screen allowing him or her to search for customers by their names with wild cards. From the hit list of customers returned from the search, the call center agent selects the correct customer. Behind the scenes, another service reads the customer's detailed data. Finally, the call center agent adds the complaint itself and a comment to that data.

Submitting the data initiates the workflow, and the complaint manager receives notification that a complaint needing attention has been entered. The application lets the complaint manager navigate to

the appropriate approval screen, which displays the complaint and comment data just entered. The complaint manager has to choose to accept the complaint, reject the complaint, or hand it back to the call center agent (e.g., in case important information is missing). If the complaint manager accepts the complaint, the application continues via a background call to create a database entry in the composite's database (note that the icon for the background step Create complaint is a rectangle, while the icons for enter, review, and approve complaints use trapezoids). Otherwise, the call center agent receives a notification to review the complaint and add the missing information. Then, the call center agent notifies the complaint manager again.

Based on this process description, you can now begin to plan your composite application development project.

Planning your development project

Once you've defined the process from a business perspective, as above, you need to figure out what pieces should go into your composite application. The key questions you need to answer to find them are:

- Which services do you need?
- On which data does a composite application work?
- Which UIs do you need?
- On which business objects does the composite application work?
- What does the process flow look like and which roles does it use?

Let's answer these questions with the following step-by-step approach:

Step 1: Search for services in the ES Workplace

Step 2: Simplify the service interfaces with CAF

Step 3: Create UIs with Visual Composer

Step 4: Model business objects with CAF

Step 5: Model collaborative processes with GPs

Step 3 will be covered in Part 2 of this series. Steps 4 and 5 will be covered in Part 3.

Step 1: Search for services in the ES Workplace

Since services are at the center of an enterprise SOA, let's begin with them. Which services do you need? Which ones already exist? What should the interfaces for the required services look like? In the complaint scenario (**Figure 3**), there are three services:

- Find the customer by name
- Read the customer's detailed data
- Create a complaint business object

One key concept behind composite applications is to reuse as much existing functionality as possible. That's why you should look for existing services that can help to shorten the development process. Remember, the ESR contains all the services that are currently available in SAP's business solutions. It is a useful listing, not only for SAP customers, but also as a reference for non-SAP customers. The ESR can help you figure out what the interfaces look like for services that supply common business functionality.

Even if your company doesn't have the ESR installed, you can benefit from the service descriptions by checking its contents. These descriptions are available publicly in the ES Workplace on SDN. From there you can navigate to the service descriptions of all the services that SAP provides. The ES Workplace consists of Web pages explaining enterprise services in detail, a services registry containing the endpoint information for services, and several back-end systems (SAP ERP 6.0, SAP SRM 5.0, and SAP SCM 5.0) where the services actually run. (For more information about the ES Workplace, read "How to use the ES Workplace" in the How-to Guides section of the "Explore Enterprise Services" home page[2] on SDN and browse the enterprise SOA Web pages.)

[2] https://www.sdn.sap.com/irj/sdn/explore-es

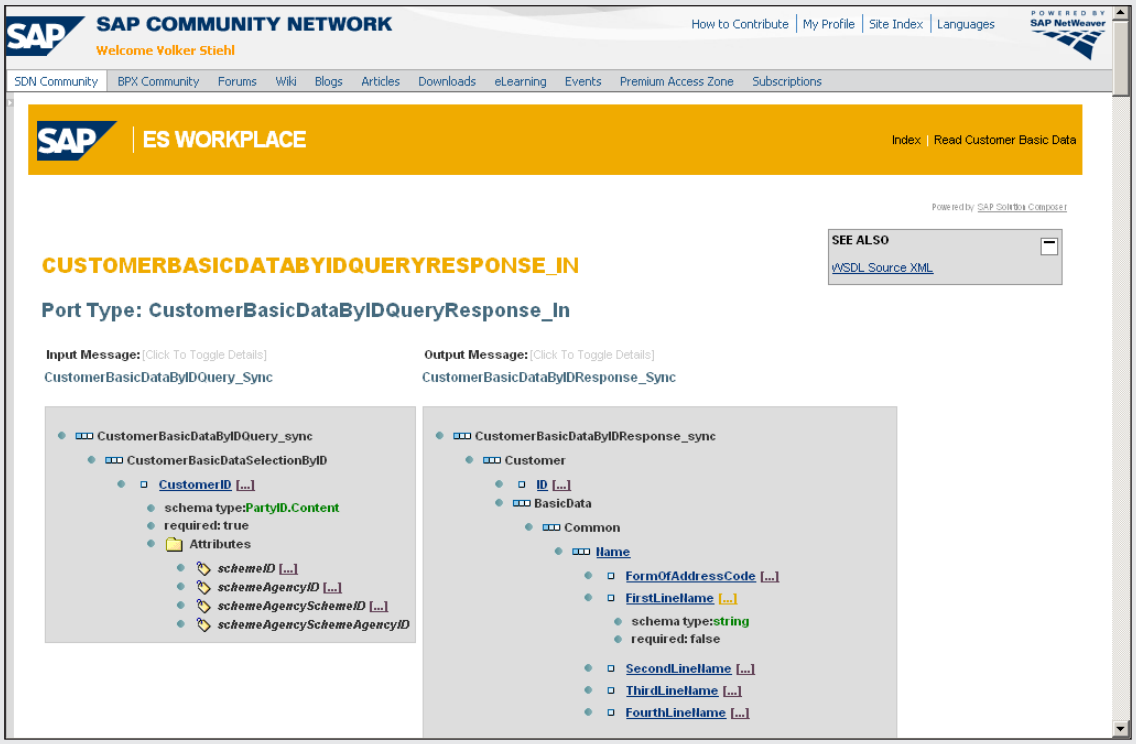


Figure 4 Parameters for the Read Customer Basic Data service

For the complaint process, you need customer-related services. Therefore, drill down from the ES Workplace home page to the customer business object. The click path looks like this: Enterprise Services Index → Process Components in ESM ERP 603 (Enterprise Services Model of ERP 6.0 Enhancement Package 3) → Business Partner Data Management → Business Objects → Customer. See the sidebar that starts on page 12 for information on an alternative way to use the ES Workplace to search for business objects and their related enterprise services. In any case, you end up on the customer business object's home page.

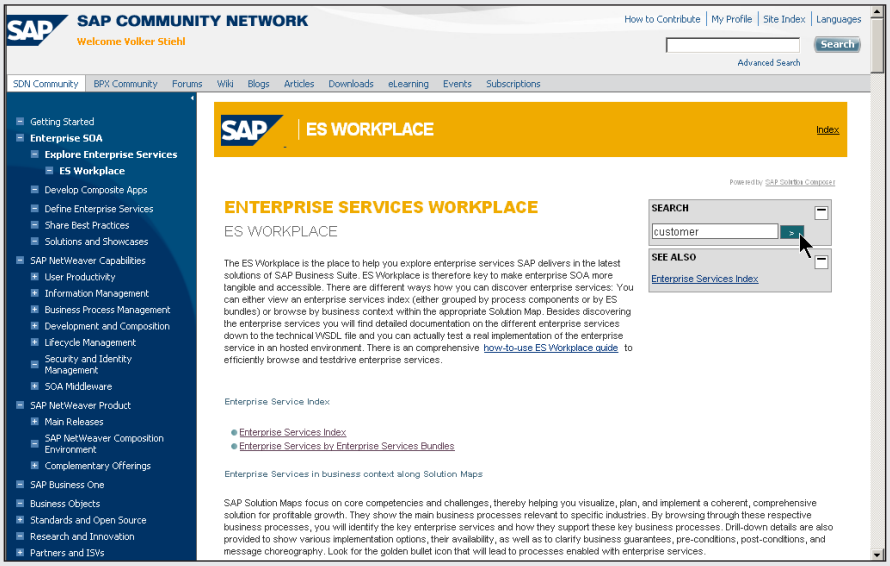
In the Related Items section of the business object's home page, you'll find all the enterprise services belonging to the customer business object. In this case, you need two of them: Because the call center agent can't expect the caller to know his or her own customer ID number, you need a service that searches for the customer by name. The Find Customer by Name and Address service is the right choice. Click on it to drill down into the details of the service.

Once the search returns a list of names and addresses, you'll need more detailed information about the customer. To obtain this, another enterprise service is needed: the Read Customer Basic Data service. Click on it on the business object's home page to retrieve more details about it.

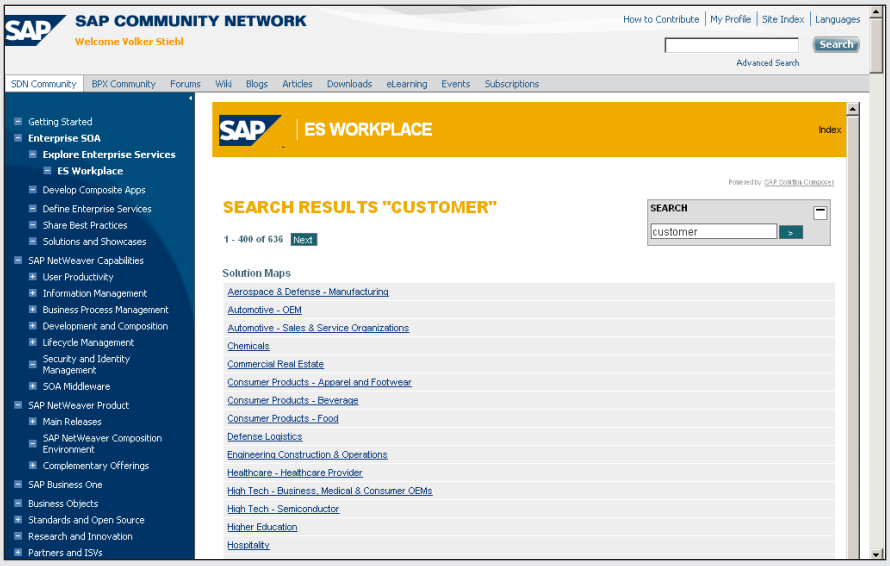
Between the two, these services provide the functionality you need. You can even drill down from the lists to retrieve further details about the service's interface. For example, if you want to know exactly what the interface looks like, click on the service's name and from the resulting page, click on Detailed field description. As result, you get an overview of the service's input and output parameters (e.g., query and response, respectively) as shown in **Figure 4**.

Using the ES Workplace to search for business objects and their related enterprise services

Developers of the ES Workplace simplified the search for enterprise services significantly. They added a "search" field, which allows you to enter the term you are looking for (e.g., customer), as shown in the screenshot below.

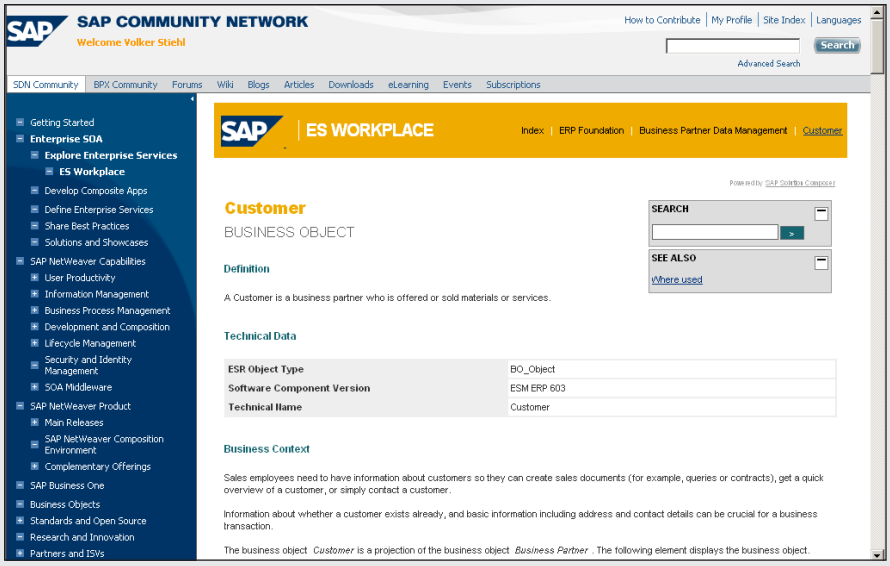


After you've clicked on the Search button, the results of the search are listed, organized in different sections (e.g., Solution Maps, Business Objects, Data Types, Process Components, and Process Component Interactions), as you can see in the screenshot below.



Scroll down, until the Business Objects section appears (see the next screenshot). Click on the Customer link of the ERP system (there is another Customer business object for the CRM system, but we are interested in ERP functionality).

Next you reach the business object's home page, as shown in the bottom screenshot.



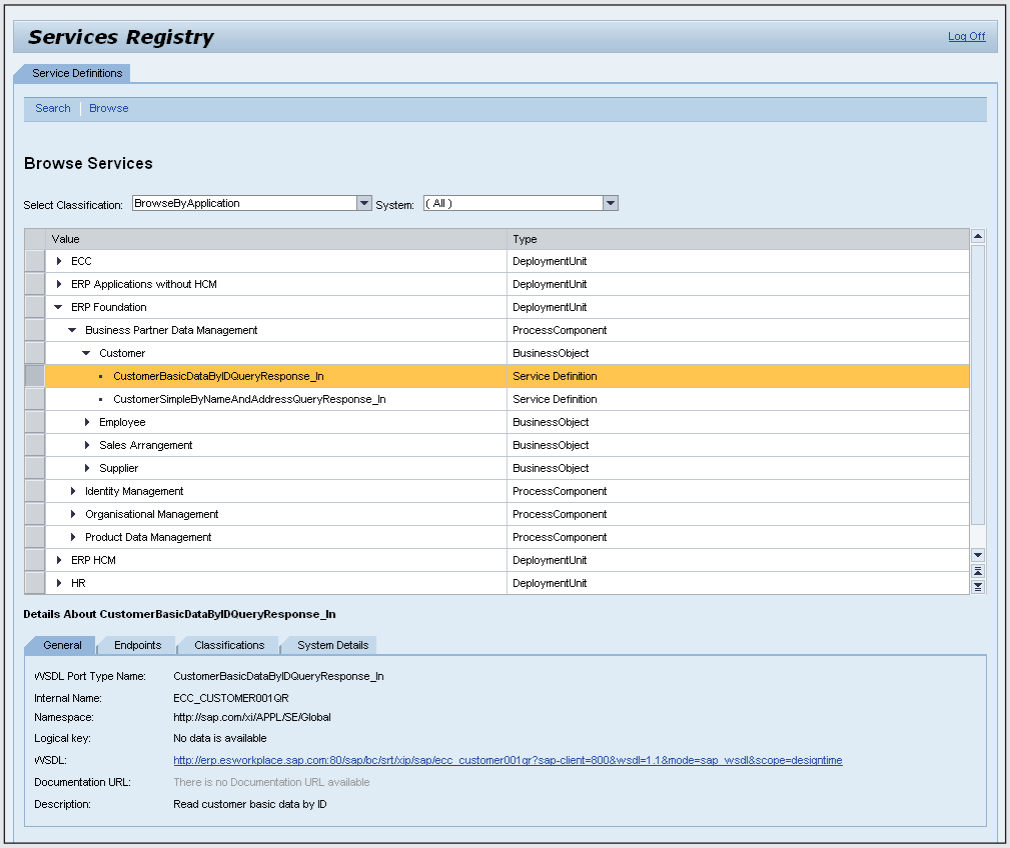


Figure 5 Services in the Services Registry

Now that you know what the service looks like, how do you determine how these services behave? Through the ES Workplace! SAP not only opens all services with service descriptions, but also hosts systems on which you can actually try the services. This is particularly useful if you don't have a service-enabled SAP system in use yet (SAP ERP 6.0 or higher) or if you have an older release. But how do you determine on which servers the services run? The Services Registry is responsible for this. And yes, SAP provides a Services Registry for you as well.

In summary SAP is hosting a complete infrastructure for you to discover and test services:

- The ES Workplace for retrieving detailed interface information for enterprise services
- A Services Registry that contains all end-point information for those enterprise services (i.e., where a particular service is running and how you call it)
- Back-end systems on which the enterprise services actually run

To access the Services Registry, proceed to the Explore Enterprise Services home page. (For details about the ES Workplace's Services Registry, I recommend you read the guide "How to use Services Registry for ES Workplace" also downloadable from the How-to Guides link on SDN.)

Remember, you not only want to browse the services in the registry, but you also want to try them

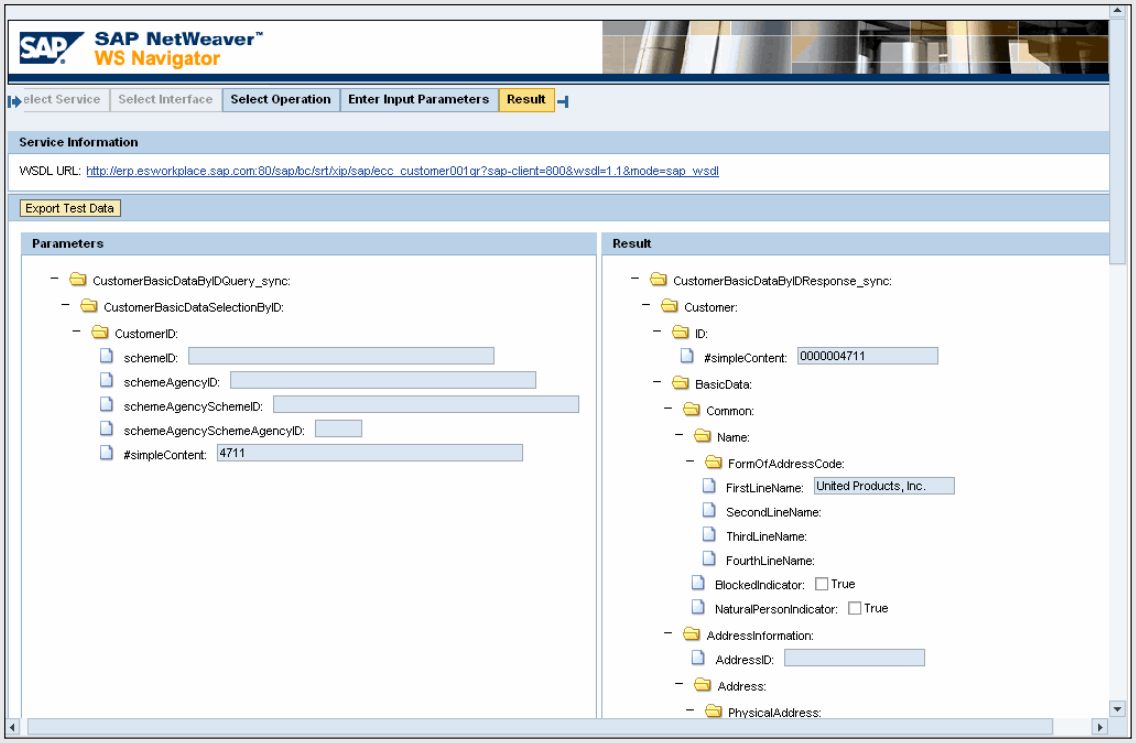


Figure 6 Result of the service call for CustomerBasicDataByIDQueryResponse_In

While you can browse the Services Registry with a publicly known user (sruser with password eswork123), the actual services run on SAP back-end systems for which you must acquire an individual user ID. Just follow the Get your user for free link on the home page to request your personal account at https://www.sdn.sap.com/irj/sdn/esareg.

To browse the Services Registry, click on Services Registry in the box of the same name on the right side of the Explore Enterprise Services screen. Use the following click sequence to reach the screen shown in **Figure 5**: Browse → ERP Foundation → Business Partner Data Management → Customer → CustomerBasicDataByIDQueryResponse_In. Details about the highlighted service appear in the table there.

To test the service, click on the Endpoints tab. Then, click on the Test button to retrieve the WSDL description for that particular service from the back-end system on which the service is actually running. You will be asked for the logon credentials of that back-end system; that's why you had to register for it previously.

After you provide the credentials for the back-end system, you will find yourself in the Web Services Navigator (WS Navigator), which enables you to test the Web service. Just select the operation you'd like to test. In theory, a Web service can consist of several operations. For enterprise services, each operation is wrapped within one Web service. In this case it's the CustomerBasicDataByIDQueryResponse_In operation. Select the row in the Operations table (even though there's only one entry, you have to select the row explicitly), and you are automatically guided to a dialog that lets you enter sample data for the service. Here, you only have to fill in the CustomerID node's #simpleContent field. An appropriate entry is 4711. Click on the execute button and the final result should look like the screen in **Figure 6**.

Now that you know how to find services and how to test them, why is this exercise important? It's crucial to identify which services exist and which ones are missing. Because SAP service-enables its business functionality (starting with SAP ERP 6.0), it recommends that you use the SAP services as a blueprint, even if you don't have an SAP system landscape in place. To provide your own implementation of a certain enterprise service, you should use the exact same interface that SAP used. That way, you're always in a position to switch to SAP's implementation without affecting other parts of your composite application. One implementation might be calling a BAPI to fulfill the business functionality if you don't have the latest SAP release installed. By following this approach, you immediately benefit from the new interface while you are prepared for a later release change.

Perhaps you want to start modeling your UI or your process, but the service isn't available yet or your programmers have just started to implement it. Your UI developers and process experts shouldn't wait. Based on the service interface taken from the ES Workplace, you can provide a dummy implementation for a particular service to support parallel development on all layers (process, UI, and business logic). CAF will help you here. The next section explains in detail how this works.

Step 2: Simplify service interfaces with CAF

Now you know the available services and how they work. As you saw in Step 1, interfaces for enterprise services can become complex. You shouldn't expose them as-is to consumers who need a particular business functionality, such as the UI or process layer. Composite applications should concentrate on the data that *they* need. For example, if you have a composite application that works on orders, the composite doesn't necessarily need every order field that's available on the SAP system. Instead, you should provide the service with a simpler interface that strips down the complexity to only those fields that are absolutely necessary. As you tailor this service to the needs of your composite application, it ceases to be an enterprise service and becomes, in the nomenclature of composite applications, an *application service*.

It's a good habit to wrap enterprise services in the business objects and services layer, not to consume them directly from the higher layers. It's recommended as well. This decouples the consumers from the enterprise services, making it easier and more flexible to replace those services later. So, you need to:

- Simplify a complex interface to the needs of your composite. This then becomes the interface of the application service.
- Provide a dummy implementation of the simplified interface so that development on other layers can continue in parallel.
- Wrap the complex external service in the simplified interface, and replace your dummy implementation.

This step focuses on the first two bullet points and uses the Read Customer Basic Data service as an example. I will discuss the third bullet point when the composite application is running with a dummy implementation. Also, please keep in mind that the service you're going to implement doesn't belong to the composite itself. It's only an alternative to the real enterprise service, and its intention is to show you how to develop your own implementations of enterprise services based on real ones.

The tool of choice for developing services is CAF, a plug-in for NWDS. (I assume you have basic knowledge about CAF so you can begin development of the simplified service. If you need an introduction to CAF, I recommend the online help for SAP NetWeaver CE. For the CAF part, I recommend the tutorial on SAP Help Portal at http://help.sap.com; in the tree on the left side of the screen, click on SAP NetWeaver → SAP

NetWeaver CE → SAP NetWeaver Composition Environment Library → Developer's Guide → Developing and Composing Applications → Composing Services with CAF.)

Consider which data your composite application really needs. In the example, you can assume that you're working with the following Customer fields:

- ID
- Name
- Region
- District
- City
- Street
- House number

Next, compare the fields that the composite needs with those that the enterprise service just returned. Compare the fields from the service's response message (**Figure 4**) with the list. You have perfect matches, as shown in **Figure 7**.

So far, you've filtered the key fields that the composite application needs out of the overwhelming list of fields available in the enterprise service. But you need to know what types of data exist in those fields before you start modeling. SAP recommends that you *always* use the same data types as the original enterprise service uses. Using the following rule of thumb, you can model a simplified service:

1. Open NWDS and start the Composite Application Explorer plug-in (Window → Show View → Other, Composite Application Framework → Composite Application Explorer).

2. Create a new CAF project (File → New → Project). Follow the menu path to start the wizard. It requests information for correctly creating a project. Choose Development Infrastructure → Development Component (first wizard step) → Composite Application (second wizard step); enter the project name (e.g., spj_provider1).

3. If you plan to reuse existing data types to define your application service's interface, you must first import the original enterprise service's interface. Before that, however, you have to make sure that NWDS is correctly connected to the Services Registry of the ES Workplace (from which you want to get the interface). Set the connection to Services Registry in the Preferences dialog of NWDS (Window → Preferences). For a correct configuration, see **Figure 8** on the next page.

   When the connection parameters are correctly set, you can continue to download the interface. Right-click on your project's external node in

Composite field to enterprise service field	Path to field in returned enterprise service
ID of the customer to ID	Customer → ID
Name to FirstLineName	Customer → Basic Data → Common → Name → FirstLineName
Region to RegionCode	Customer → Basic Data → AddressInformation → Address → PhysicalAddress → RegionCode
District to DistrictName	Customer → Basic Data → AddressInformation → Address → PhysicalAddress → DistrictName
City to CityName	Customer → Basic Data → AddressInformation → Address → PhysicalAddress → CityName
Street to StreetName	Customer → Basic Data → AddressInformation → Address → PhysicalAddress → StreetName
House number to HouseID	Customer → Basic Data → AddressInformation → Address → PhysicalAddress → HouseID

Figure 7 Mapping the Customer fields to the fields of the enterprise service

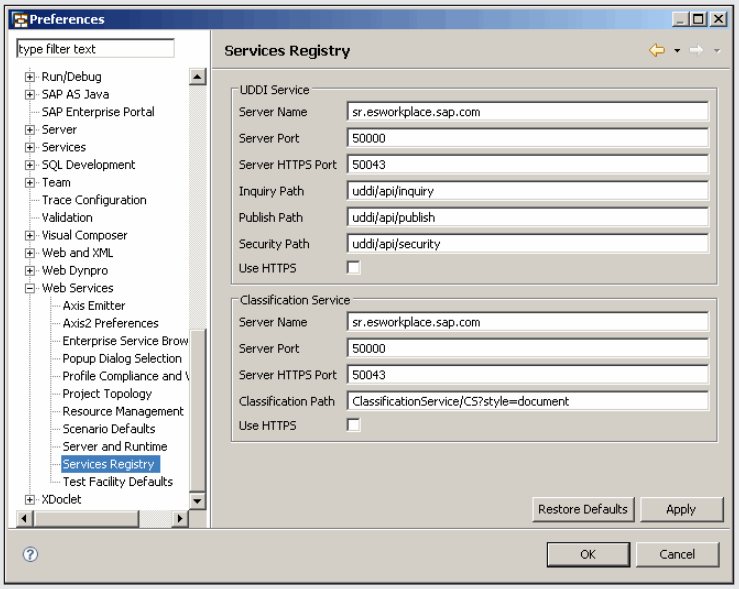


Figure 8 Connecting NWDS to the ES Workplace

the Composite Application Explorer, and select the Import Web Service entry from the context menu. The dialog that pops up offers you three options from which to fetch the enterprise service: Enterprise Services Repository, Remote Location/File System, and Services Registry. In this case, choose Services Registry, and click on Next.

Then, a logon dialog pops up asking you for the credentials to connect to the Services Registry. Enter the publicly known user (sruser with password eswork123). On the next screen, a wizard for searching the registry, you can simply enter a search criteria or add more advanced search parameters by following a link called Show Advanced to restrict the search further. When you click on Show Advanced, you get a list of all of the registry's classifications: Business Object, Deployment Unit, Lifecycle Status, Process Component, Service Interface, Service Operation, and Software Component Version, just to name a few. These classifications let you structure and group services; you typically place services that belong together in a classification so you can find them more easily. If you imagine thousands of enterprise services, you'll appreciate classifications to help you find them faster. For simplicity, expand the Business Object node and check the checkbox of the Customer node. This restricts your search, leading to shorter response times.

As an alternative, you can also find the Customer node by following the path: BrowseByApplication → DeploymentUnit → ERP Foundation → ProcessComponent → Business Partner Data Management → BusinessObject → Customer. This approach indicates that you want to see only those services associated with the Customer business objects. Click on Next to see a list of Found Service Definitions. See **Figure 9**. Pick the "read" service from the list (see Description field), and then select the appropriate end point (which is retrieved after you choose the service definition).

You can also get the WSDL description for a service from the file system. So, you can finish this exercise even if you don't have a connection to the Services Registry. You can get the WSDL file you need from *SAP Professional Journal*'s download page for May/June 2008, save it on your

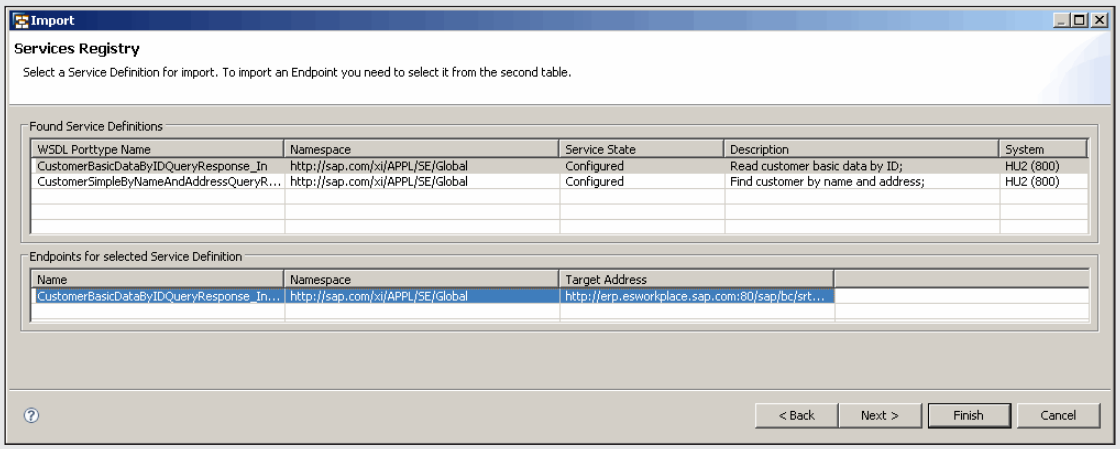


Figure 9 Choosing the end point for the read service's definition

local file system, choose the Remote Location/File System radio button, and click Next.

Click on Finish. After you enter the credentials for the back-end system from which the WSDL description is retrieved, you'll find all the data types that make up the enterprise service in your project's "external" node, as shown in **Figure 10** on the next page.

4. Next you create a new structure. To do this, right-click on the node labeled "modeled" and choose New Structure from the context menu. In the dialog that pops up, enter the name for this new structure (e.g., CustomerBasicData) and click on Finish. A new tab opens, as shown in **Figure 11** on the next page. On the left side, you see all the available data types from which you can choose to build your structure (even those imported from the enterprise service). On the right side, you see a list of the fields in your new structure. Initially, it is empty. Your task is to add fields to this structure.

   To do this, select the appropriate data type from the list on the left side and then click on the button with the right-facing arrow (the first one at the top of the column of buttons). The new field appears in your structure, and you can give it a new name.

   Repeat these steps until your structure looks like the one shown in **Figure 11**.

   In summary, your new structure now contains those fields your service call should return to the caller. It is much simpler than the corresponding enterprise service's return structure (its output parameters). These are the fields that contain the information from the enterprise service that you want to use in your composite application.

   So, you've chosen the data types from the original enterprise service, but how do you know which data types you need? Use the Composite Application Explorer and browse the output parameters for the imported enterprise service. Let's assume you want to find out what type applies to the street name, which is in the PhysicalAddress node. The field's type name is StreetName, as shown in **Figure 12** on page 21. That's the type you should use in your return structure above as well.

5. Create a new application service with the name ReadCustomerBasicData (right-click on the node called "modeled," and choose New Application Service from the context menu).

6. Create a new readCustomerBasicData operation

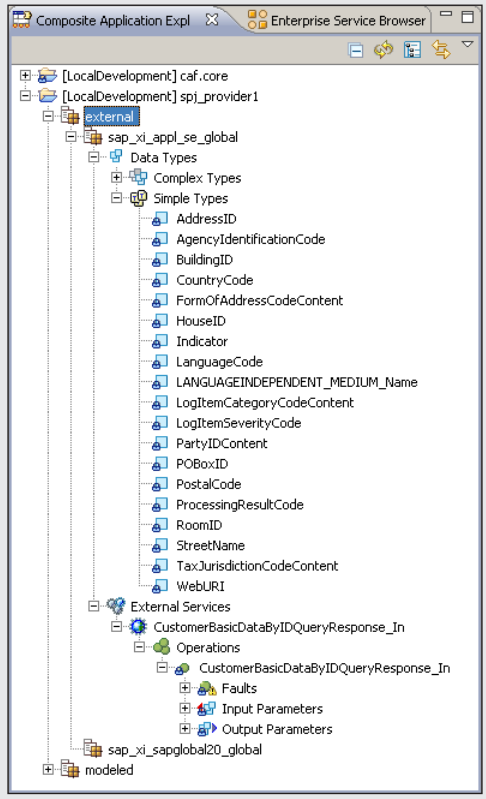


Figure 10 The enterprise service CustomerBasicDataByIDQueryResponse_In's data types

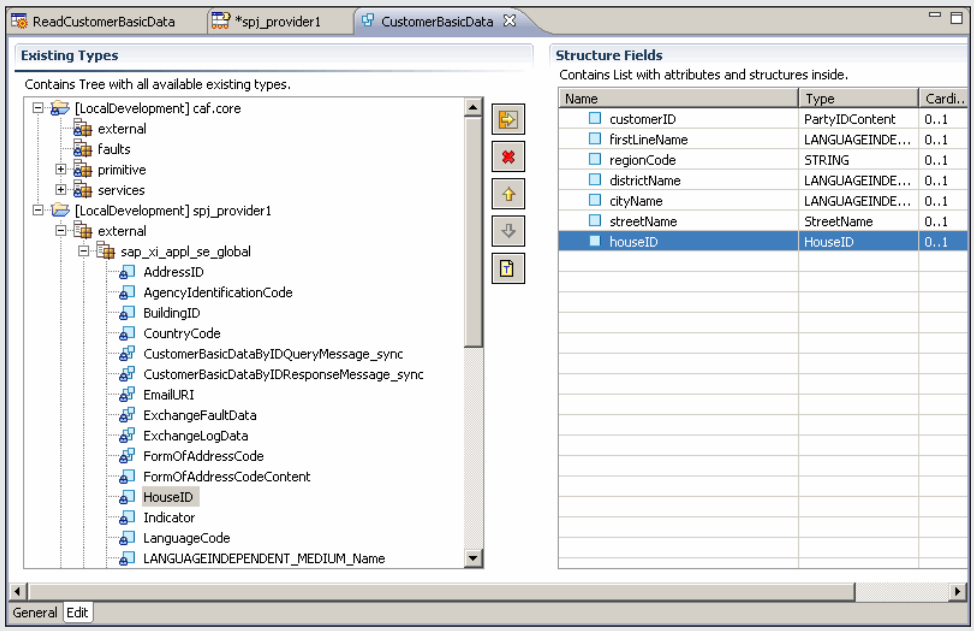


Figure 11 New structure (on the right) containing the composite application's relevant fields

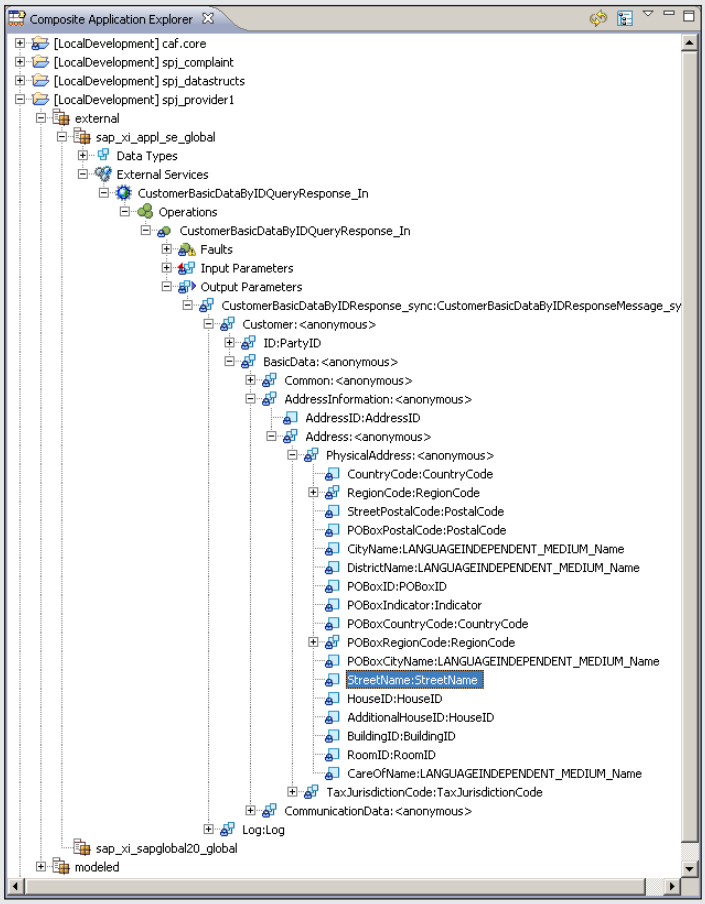


Figure 12 Structure of the enterprise service's output parameters

for the application service. Switch to the Operations tab of your application service and click on the Add button. You need the customer ID (type PartyIDContent) as an input parameter, and as an output parameter you want your new structure CustomerBasicData (see **Figure 13** on the next page). Make sure that the Implemented checkbox near the top of the screen is checked. It indicates that this method is not mapped against an external service. Instead, it contains its own logic, its own implementation. This marker means, during design time, you add your own coding.

7. Now you add the Java code to the application service. For this, select the Implementation tab (at the bottom of the screen) and click on the link to the Java class. The Java Editor will open, and you should add the code shown in **Figure 14** on the next page.

   As you can see, the implementation of the method returns static data independent of the customer ID that you transferred to the service. This is OK for this scenario because you replace the implementation with a real enterprise service in the end without affecting the UI modeling.

8. Because you want to consume the service on the UI layer, you must make the functionality

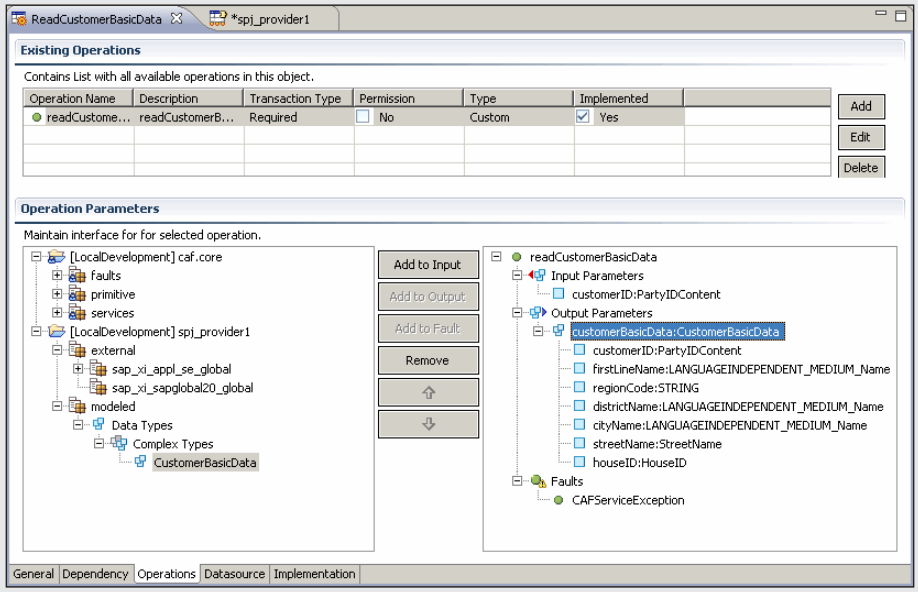


Figure 13 Operation of the application service

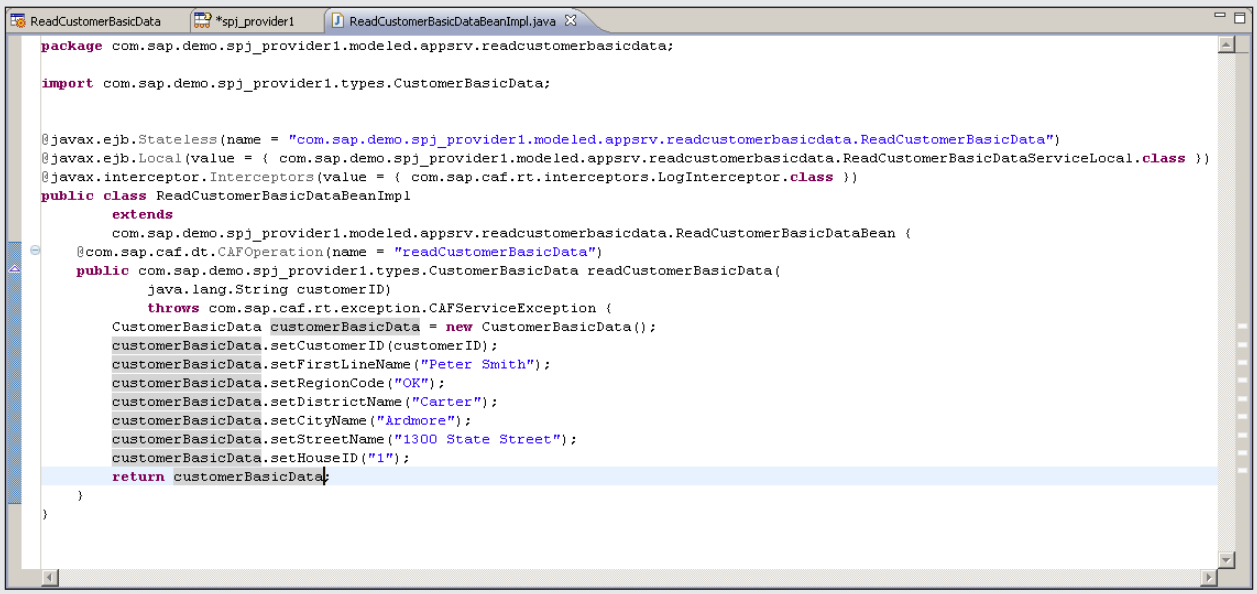


Figure 14 Implementing the example enterprise service

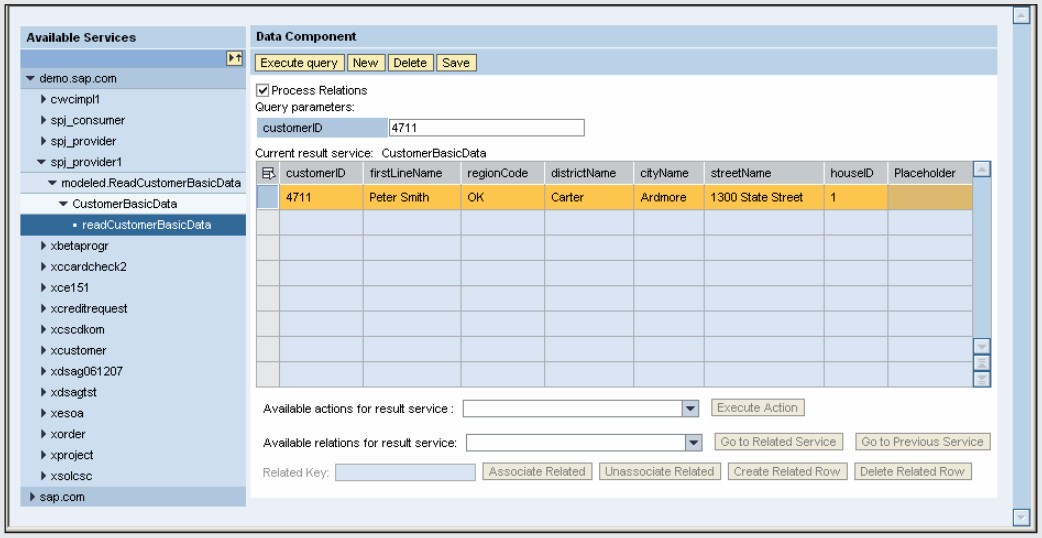


Figure 15 Testing the application service in the Service Browser

available as a Web service. Right-click on your application service in the Composite Application Explorer pane and choose Expose service as Web Service from the context menu. (The application service is really a session EJB. The EJB interface is mapped to a WSDL representation conforming to the EJB specs.) In the dialog that pops up, enter ReadCustomerBasicDataService. SAP recommends that the name of the Web service consists of the application service name plus the suffix "Service" (e.g., if the application service's name is "xyz," the Web service's name should be "xyzService"). That's how CAF works if it derives the names of the Web services from the WSDL descriptions.

When you want to replace your dummy service with a real implementation, you derive a new application service from the WSDL file of the dummy implementation. A smooth replacement is possible only if the Web services names are identical. That's why you have to pay attention when you're naming the Web service.

9. Save, generate, build, and deploy your application. You will find the commands as entries in the context menu of your project (right-click on your project in the Composite Application Explorer).

10. Test your application in the service browser that comes with CAF (right-click on your application service, and choose Test Service from the context menu). The result of your service execution should look like the output in **Figure 15**.

11. Test your application service in the WS Navigator as well to ensure that the Web service generation was successful. Open your WS Navigator via URL http://<*host*>:<*port*>/wsnavigator. Select your Web service from the list of available Web services, click on the readCustomerBasicData operation, and provide a customer ID. If you can't identify your Web service in the WS Navigator immediately, type the project name in the filter field, and press Return; your Web service then appears as the only entry in the table. Since the

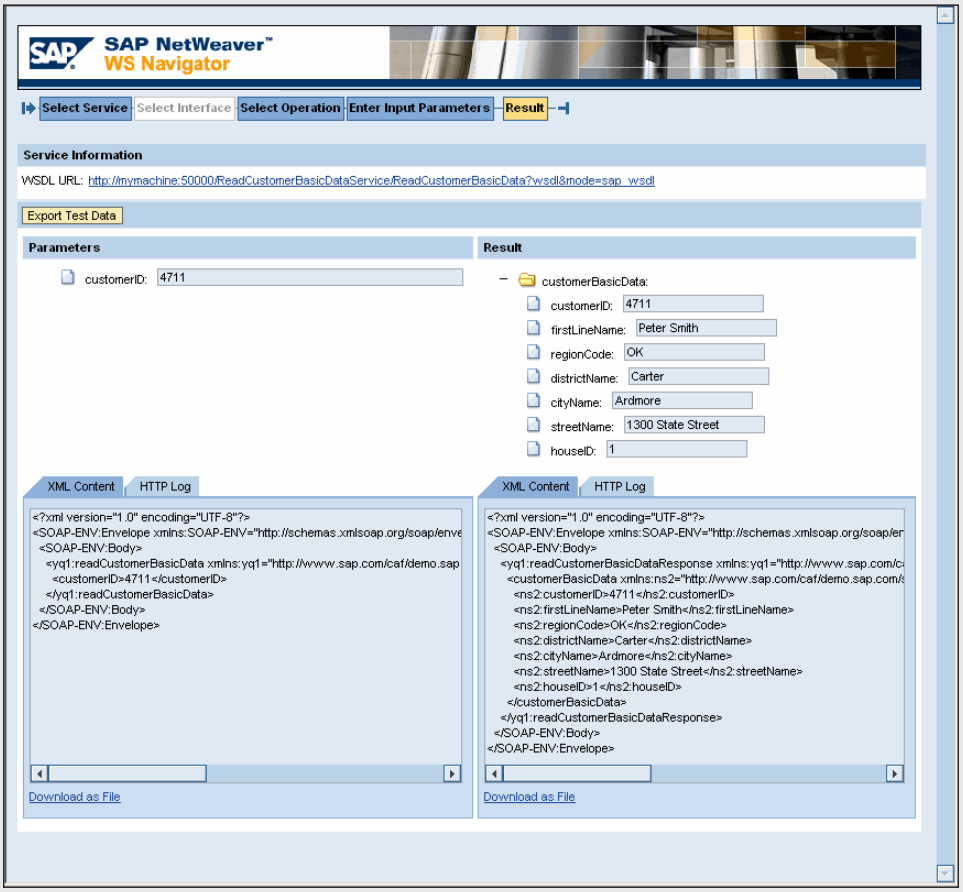


Figure 16 Testing the application service in the WS Navigator

dummy service doesn't evaluate this ID, you can enter whatever you want. As result, you should see something similar to that shown in **Figure 16**.

Now that you have implemented the service functionality and it's available as a Web service, you can easily consume it (e.g., in the UI or process layer), but don't do this directly. Remember, doing this would make your Web service dependent on the system on which it runs, and you want it to be as independent as possible. That's why you create a logical destination for your service, decoupling the service consumer from the service provider. It is similar to the concept of SAP Java Connector (JCo) connections in Java Web Dynpro that decouple the ABAP connections by maintaining JCo connections in the System Landscape Directory (SLD).

Creating a logical destination is part of SAP NetWeaver Administrator (NWA), which comes with SAP NetWeaver CE. Basically, you provide a name for your Web service and the service consumer works only with this name, not the actual service. This enables you to replace a service implementation later. To create a logical destination, you need to do the following:

1. Open NWA via URL http://<*host*>:<*port*>/nwa.
2. Click on SOA Management → Technical Configuration.

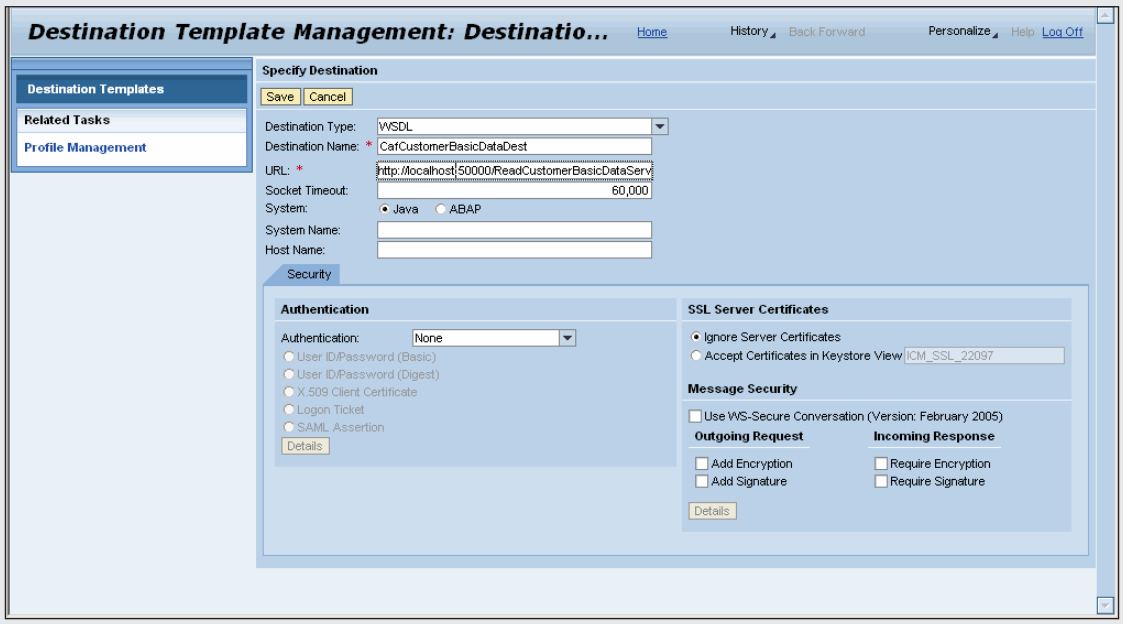


Figure 17 Creating a logical destination for your dummy service

3. Click on Destination Template Management. Here, you reach the editor for logical destinations. A table lists all previously defined destinations.
4. Click on the Create Destination button for a new destination.
5. Fill the fields as follows:
   a. Destination Type: WSDL
   b. Destination Name: CafCustomerBasicDataDest
   c. URL: <URL for WSDL file taken from the WS Navigator>
   d. Authentication: None (choose from drop-down list if not set)
6. Save your entry and see the final result, as shown in **Figure 17**.

That was the last step on your journey into service adaptation. You created a dummy service for a real-life enterprise service with a simplified interface based on the original data types of the enterprise service. You published the service as a Web service, tested it, and made it publicly available for service consumption via a logical destination. This takes you to consuming services on the UI layer, which is covered in Step 3 in the second article in this series.

One final remark about this dummy approach: You can apply this technique for the second enterprise service in your example, too. The benefit is that you can build the whole application without accessing any external services. So, if you run into problems with external service consumption, use this technique to proceed.

Conclusion

Composite applications play an important role in SAP's enterprise SOA strategy and with that the SAP NetWeaver CE becomes a cornerstone in SAP's product portfolio. Enterprise SOA changes the way

software will be developed in the future. SAP NetWeaver CE provides an environment that supports you in your efforts to implement innovative collaborative processes on top of existing IT landscapes. The importance of composite applications will only increase over time.

Developing composite applications isn't necessarily easy, but you can divide it into the following five manageable steps:

Step 1: Search for services in the ES Workplace

Step 2: Simplify the service interfaces with CAF

Step 3: Create UIs with Visual Composer

Step 4: Model business objects with CAF

Step 5: Model collaborative processes with GPs

This article discussed the first two steps, explored SAP NetWeaver CE, and gave you some best practices and recommendations to support your own development efforts. The second article talks about creating UIs with Visual Composer. The third article in this series will discuss the remaining two steps: modeling business objects with CAF and modeling the collaborative process with GPs.

