June 2006

Building Workflow enabled End-to-End Applications using Microsoft .NET Framework 3.0 and SAP Enterprise Services over Web Services

White Paper

Authors:

Lei Liu, Spell GmbH, lei.liu@spell-gmbh.com
Tilo Böttcher, Microsoft Corp, tiloboet@microsoft.com
Jürgen Daiberl, Microsoft Corp, jdaiberl@microsoft.com

Summary:

This whitepaper reviews the next generation of Windows managed programming APIs - .NET Framework 3.0 (formerly codenamed “WinFX”) and demonstrates how the core parts of the .NET Framework 3.0 technologies, namely Windows Communication Foundation (WCF) and Windows Workflow Foundation (WF), can be used in combination with SAP’s Enterprise Services via Web Services to build workflow enabled end-to-end enterprise applications. Furthermore, this whitepaper explains how Office applications can be used as the front-end UI for such business workflow by means of Visual Studio Tools for Office.

Applies to:

- Microsoft .NET Framework 3.0 Runtime Components - Beta 2
- Microsoft® Windows® Software Development Kit (SDK) for Windows Vista and .NET Framework 3.0 Runtime Components - Beta 2
- Microsoft Visual Studio 2005 Team Suite
- Microsoft Visual Studio 2005 Tools for Office (VSTO)
- Microsoft Visual Studio Development Tools for .NET Framework 3.0 Codename “Orcas”
- Microsoft Visual Studio 2005 Extensions for Windows Workflow Foundation Beta 2
- Microsoft Office 2003 Professional
- SAP NetWeaver 2004s ABAP Edition

Keywords:

.NET Framework 3.0, WinFX, WCF, WF, Visual Studio, Office, SAP ECC, SAP NetWeaver, Service-oriented Architecture

Audience:

Technical consultants, Architects, Developers, IT Managers

For the latest information, please visit http://www.microsoft-sap.com

Contact

This document is provided to you by the Collaboration Technology Support Center Microsoft. For feedback or questions, you can contact the CTSC at ctsc@microsoft.com.

Executive Summary

As two leading companies in the new era of service-oriented development, Microsoft and SAP make incessantly efforts to embed their architectural best practices for building highly adaptable software and services in their products, which can help customers to design, develop and manage their service-oriented architecture using Web Services as their key technology. In this whitepaper, we introduce Microsoft's next generation managed programming model - .NET Framework 3.0 (formerly codenamed "WinFX") - and outline how it can be applied to build reliable, secure, robust and effective end-to-end solutions based upon SAP Web services.

Using the existing .NET Framework 2.0 components at the core, .NET Framework 3.0 introduces a set of developer-focused innovative technologies in terms of Windows Presentation Foundation (WPF, formerly codenamed "Avalon"), Windows Communication Foundation (WCF, formerly codenamed "Indigo"), Windows Workflow Foundation (WF), , and the new Windows CardSpace (WCS, formerly known as "InfoCard") for workling with and managing digital identities. For building enterprise end-to-end applications with workflow support, WCF and WF are the fundamental parts for developers to facilitate the development process. WCF provides secure, reliable, and transacted interoperability through the build-in support for WS-* specifications, which dramatically reduces the overhead to archive the heterogeneous interoperability. Furthermore, WF delivers the common framework for building workflows into client and server side Windows applications that coordinate interactions among software or/and people. Utilizing WCF and WF together with SAP's Enterprise Service exposed through Web services, every business can build processes today directly from within their Windows applications to interact with customers, partners and suppliers, both within and beyond the walls of the organizations, in spite of the platforms they use.
In this whitepaper, we explain how a business can utilize these products to facilitate their interoperable IT infrastructure that is resilient to inevitable change and is easy to manage over time by means of the common flight-booking scenario from SAP's basis training. Driven by the user interaction through smart clients in Microsoft Word 2003 and Microsoft Outlook 2003 as well, even casual users can search for flight connections to various destinations and make a booking request directly from Office UI. Managers can deny or approve the booking request from their Office UI as the case may be. Furthermore, we will outline the impact of the new programming model in terms of security and transactional behavior of activities in the business process. With many practical step-by-step instructions for building Web service-based applications as well as configuring SAP web services, one can use this whitepaper as a hands-on paper to develop WinFX-based Smart Client applications using Web services from SAP backend ECC system.

Table of Contents

Introduction

In today's world of globalization, enterprises are always looking for competitive advantages to deliver agile IT solutions to support business. With the help with Web services, enterprise can build dynamic and effective computing infrastructure within and beyond the boundaries of the organizations. A flexible and reliable framework for building service-oriented and workflow-enabled applications is crucial for the success of enterprise computing. The forthcoming Windows .NET Framework 3.0, formerly called **WinFX**, is comprised of .NET managed APIs based on the core runtime components of .NET Framework 2.0 for a set of new technologies, among other things, the **Windows Workflow Foundation** (WF) and the **Windows Communication Foundation** (WCF, former code-named "Indigo") together with the **Windows Presentation Foundation** (WPF, former code-named "Avalon") and **Windows CardSpace** (WCS, formerly known under the codename "InfoCard") – the next generate solution for working with and managing diverse digital identities. Together with Visual Studio 2005, the integrated development environment from Microsoft, .NET Framework 3.0 provides the robust foundation for building agile network enabled end-to-end server and client applications.

As two leading companies in the technology revolution towards service-oriented enterprise computing, Microsoft and SAP work closely to help customer to design, develop, deploy and use highly adaptable software using the advantage of Web services. Based upon the Web services interoperability profiles defined by the open industry organization WS-I[1], both companies provide all the necessary technical platforms to transport business data from as well as into SAP. Since SAP Web Application Server 6.20, SAP began to open to the outside world. For the ABAP business functionalities, which were only accessible via various connector products (Java Connector, .NET Connector, etc) based on the SAP's proprietary ABAP-based communication protocols (RFC, IDoc), customers have the possibility now to expose both built-in and customized RFC-enabled function module as Web services.

In this whitepaper, we will introduce the capabilities of the forward-looking technologies from Microsoft and SAP based on a sample scenario and outlines how the technologies can be combined with one another to realize the maximal value out of them. The sample scenario reproduces the common flight-booking process and demonstrates how to build an inter-organizational workflow upon the .NET Framework 3.0 technologies and SAP Enterprise services.

[1] *Web Services Interoperability (WS-I) Organization*, http://www.ws-i.org

Intended Audience

This whitepaper aims at the technical decision makers and architects. With detailed but concise technical overviews, this whitepaper helps them to understand the technical concept, the functionalities and the use of the aforementioned technologies and products. With detailed step-by-step development instructions for the flight-booking scenario, this whitepaper is also useful for developers and technical consultants who work with IDES or SAP NetWeaver AS Sneak Preview editions. Because the solutions are developed in Visual C#, it is assumed that the readers have fundamental understanding about Visual C# programming with .NET framework. Furthermore, the readers should have basic understanding about the concept of Web services and service-oriented enterprise computing.

The Ideal World of Service-oriented Computing

The reference model for SOA from OASIS[2] defines the service-oriented architecture as a paradigm for organizing and utilizing distributed capabilities that may be under the control of different ownership domains. An SOA provides the necessary capabilities to integrate, publish, discover and manage services. The central concept in an SOA-based enterprise computing is *services*, which are means that meet the needs of service consumers with the capabilities brought by service providers. A service in an SOA can be any possible functionality that exposes its capability using a prescribed interface in compliance with the SOA standards and acts consistently with the constraints and policies defined in the service contract. The actual implementation of the respective service is therefore not the concern of SOA. This “black box” approach allows changing the service’s implementation details on the provider side without impact upon the service consumer, as long as the prescribed interface being exposed remains unchanged. Furthermore, it allows the integration of legacy applications that are either not network-enabled or not standard compatible with SOA.

The SOA-based computing infrastructure is a completely XML-driven architecture. The Web service framework defined by W3C [3] builds on top of three core specifications: *Web Services Description Language* (WSDL[4]) for service description, *Universal Description, Discovery, and Integration* (UDDI[5]) for service discovery and *Simple Object Access Protocol* (SOAP[6]) for message transmission. This basic Web service architecture establishes the foundation for creating loosely coupled

[2] OASIS, *Reference Model for Service Oriented Architecture*, http://www.oasis-open.org/committees/download.php/16587/wd-soa-rm-cd1ED.pdf
[3] W3C, *Web Services Architecture*, 2004, http://www.w3.org/TR/ws-arch/
[4] W3C, *Web Services Description Language*, http://www.w3.org/TR/wsdl
[5] OASIS, *Universal Description, Discovery, and Integration*, http://www.uddi.org/specification.html
[6] W3C, *Simple Object Access Protocol*, http://www.w3.org/TR/soap/

Web services that encapsulate isolated business functionality. Based on these core specifications, businesses can build service-oriented applications within or beyond the boundaries of organizations. However, these specifications are not sufficient for building applications in the real world, because they do not address most of the problem domains that distributed systems have to face, such as reliable messaging, security, context, and transaction based on the stateless connections between services. The goal to empower the service-oriented architecture to meet the real world's requirements drive the Web services community to extend the capabilities of the Web services architecture based on the W3C Web service framework. In the following, some of the major emerging Web services specifications are listed:

- **Messaging**: a challenge for distributed computing platforms is reliable messaging between both communication partners. To make Web services capable of enterprise level applications, BEA, IBM, Microsoft and TIBCO have jointly published the *WS-ReliableMessaging* specification[7] to allow messages to be delivered between distributed applications even in presence of software, system or network failures.
- **Transaction**: the initial set of Web services specifications lacks support for maintaining context across several loosely coupled Web services because the Web services work independently and stateless from each other. To enable distributed transactions across several Web services, further Web services specifications are proposed, such as *WS-BusinessActivity*, *WS-AtomicTransaction* and *WS-Coordination* that are currently hosted by OASIS Web Services Transaction TC[8].
- **Security**: To well-established service-oriented enterprise applications belongs a well-secured communication framework. Diverse specifications have been proposed by industry and standardization organizations. The foundations for the Web service security framework are *XML Signature*, *XML Encryption* from W3C and *WS-Security* from OASIS[9]. They establish the security measures along the message transport way and protect the SOAP messages from unauthorized actions.

[7] BEA, IBM, Microsoft and TIBCO, *WS-ReliableMessaging Specification*: http://msdn.microsoft.com/webservices/webservices/understanding/specs/default.aspx?pull=/library/en-us/dnglobspec/html/wsrmspecindex.asp

[8] Microsoft, *Web Services Transaction Specifications Index Page*: http://msdn.microsoft.com/webservices/webservices/understanding/specs/default.aspx?pull=/library/en-us/dnglobspec/html/wsatspecindex.asp

[9] Microsoft, *Web Services Security Specifications Index Page*: http://msdn.microsoft.com/webservices/webservices/understanding/specs/default.aspx?pull=/library/en-us/dnglobspec/html/wssecurspecindex.asp

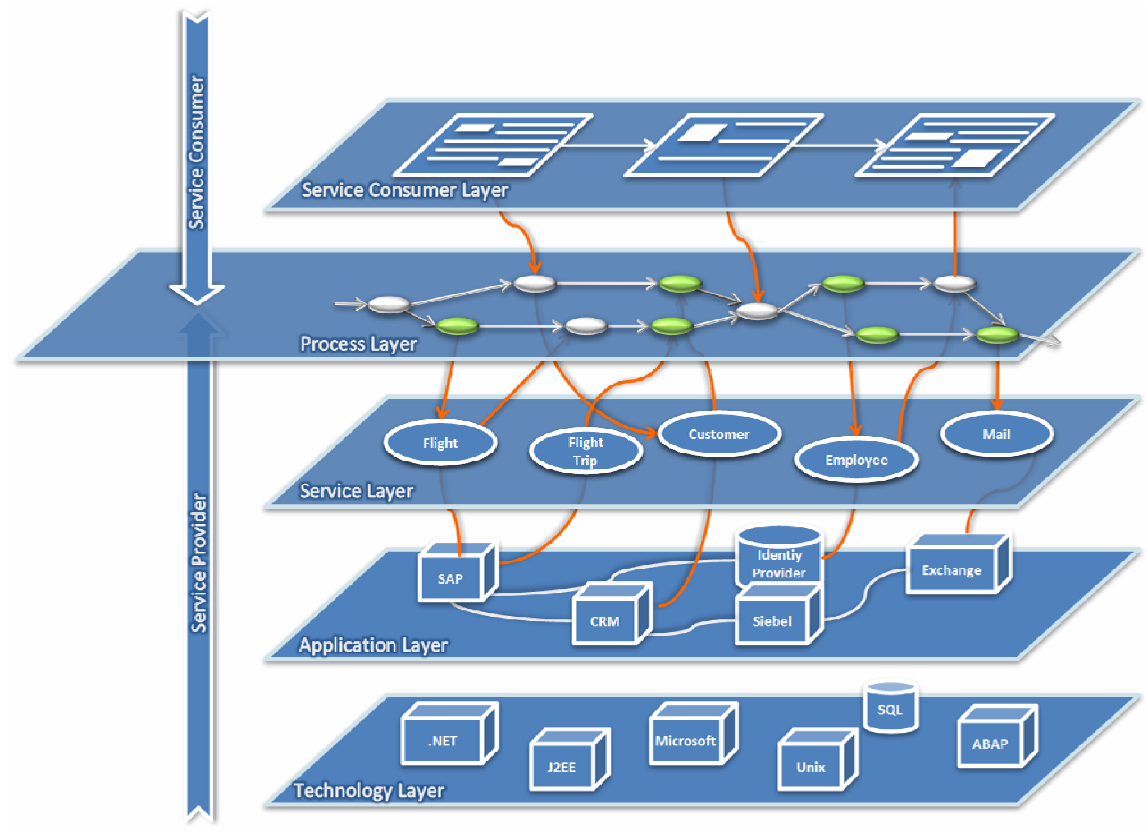


Figure 1: The ideal World of Service-oriented Computing Infrastructure

Figure 1 shows an ideal world of enterprise computing based on service-oriented architecture. The lowest layer, the *Technology* layer, contains the enabling technologies for the enterprise computing and builds so that the foundation for the applications in the *Application* layer. The enterprise applications, including the legacy systems, expose their unit of business functionalities as services in the *Service* layer. The business processes in the *Process* layer compose the services into processes with composite capabilities. The *Process* layer separates the service provider from the service consumer, where the client applications consume the services as well as the business processes provided by the server applications. As shown in Figure 1, an ideal service-oriented enterprise-computing infrastructure maximizes the reusability of the software adopted in the infrastructure by encapsulating the software capabilities into standard-based services. It enables the consumption of services across the organizational boundaries and facilitates cross-organizational business processes. The clear separation between the layers in the service-oriented computing infrastructure simplifies the development for enterprise-level applications and provides the foundation for agile development of such applications.

MSDN provides a set of good articles at all levels for understanding Web services and service-oriented architecture. You can visit the

MSDN's Web services column http://msdn.microsoft.com/webservices/ for more information.

The Computing Infrastructure in the Pre-WinFX Era

In the last section, we have reviewed the concept of the service-oriented computing and benefits of adopting service-orientation into the enterprise-computing infrastructure. However, it depicts only an ideal world with the perfect encapsulation of business functionalities as Web services. In this section, let us simply go back to the real world and check up the reality in the enterprise IT.

Figure 2 illustrates the heterogeneous computing infrastructure with the end-to-end applications in the today's IT. At first glance, the situation differs strongly from the ideal SOA-world in the roles of the service provider and the service consumer. The service provider provides its services not only in the *Service*- and the *Process* layer, but also directly in the *Application* layer. Analogically, the service consumer has to work even till down to the *Application* layer to get the capability that it needs to fulfill its functional requirement. This unclearly defined concept between applications, services and processes causes enterprise applications that operate throughout nearly the whole technology stack in the enterprise. Such enterprise applications are vulnerable to changes in the technology stack and require high maintenance efforts in case of changes.

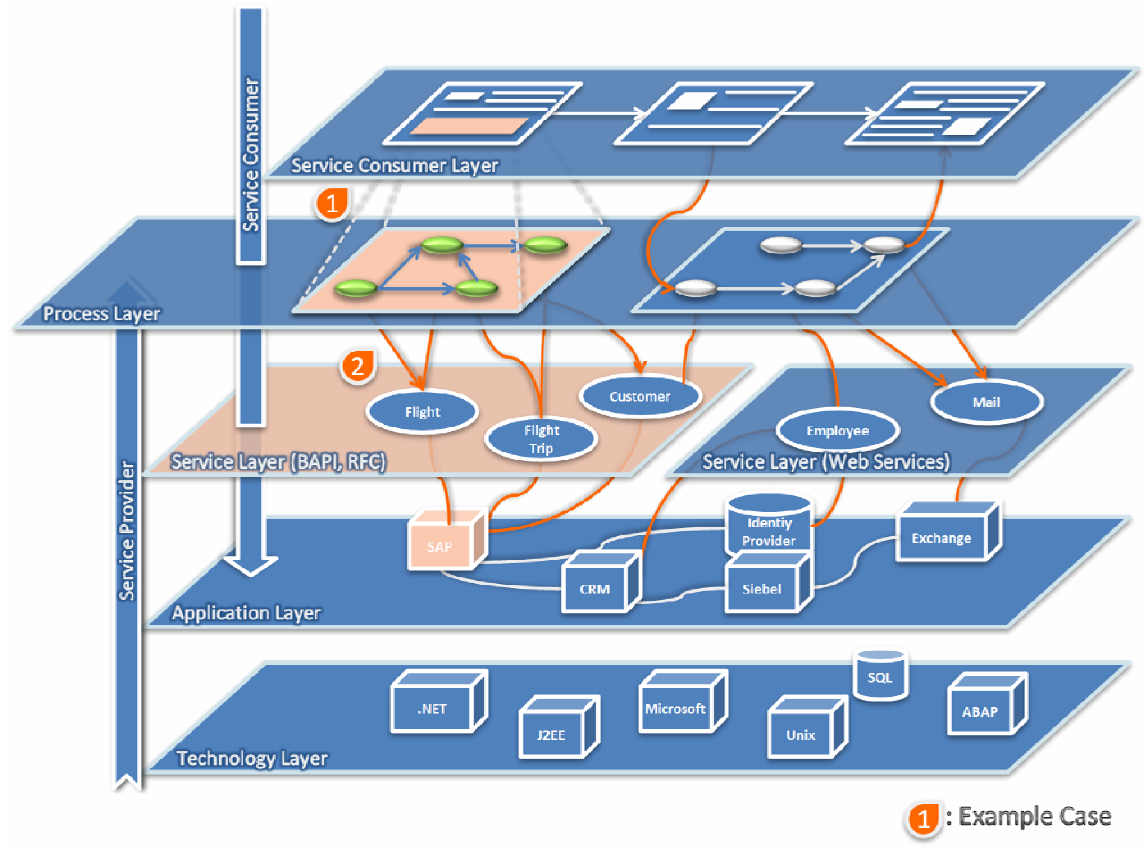


Figure 2: Heterogeneous Computing Infrastructure with SAP

Among other things, there are the following two challenges to build a workflow-enabled end-to-end application for enterprise:

- **Building Workflows to reflect the business processes**: in an enterprise application, it is normally a functional requirement to build workflows into the application. There are several common workflows in daily business, for instance, workflows within line-of-business (LOB) applications, such as SAP; document-centric workflow, where the workflow is driven by some document(s); human workflow, where several roles in the enterprise are involved in the workflow. Without the adoption of external business process management applications, such as Microsoft BizTalk Server or SAP XI, the application has to model workflows by itself-such as the example case 1 in Figure 2-where the workflow is built directly hard-coded into the application. The design and implementation of the underlying workflow framework for modeling and hosting workflows is quite a challenge for the software developers. Furthermore, the necessary utility functions for workflow runtime management, such as state tracking, persistency of state information for long-running processes, etc, complicate the application development process additionally. What's more, non-functional requirements on the workflow framework, such as flexibility in case of changes, scalability regarding number of the activities in the workflow and security in the workflow makes the development even more challenging for developers.
- **Accessing Functionalities in a unified manner**: An enterprise application needs to access business capabilities from various applications and backend systems. Just like the example case 2 in Figure 2, to access the functionalities exposed by the SAP system, the application has to use an external framework, such as the SAP Connector for Microsoft .NET, to call the BAPIs. The execution is carried out through the SAP's proprietary RFC protocol. Therefore, an enterprise application must have the necessary components to communicate with backend systems through different communication protocols, which makes the development process more complex and complicates the administration of such enterprise applications.

The Service-oriented Computing Infrastructure with .NET Framework 3.0

With the introduction of .NET Framework 3.0, developing workflow-enabled enterprise applications enters a new generation. With WCF and WF as two of the core technologies of .NET Framework 3.0, building service-oriented applications on the Windows platforms gets more simplified than ever. Among other things, the new technologies in .NET Framework 3.0 bring the following improvement for the software development:

- .NET Framework 3.0 introduces the Workflow foundation, which is a framework for building workflows into Windows applications. The developers can use the WF APIs to build both human and system workflow scenarios on the .NET platform, and mostly on the server side. For application developers, WF provides the necessary abstractions of the artifacts of a workflow, which make it easy to describe the real world workflows using the integrated development tools in Visual Studio 2005. All workflows, including the long-running ones, can be hosted directly in the WF due to the throughout maintenance of workflow state in the framework at runtime. In addition, the ability for developers to override or skip the steps in the workflow at runtime gives the workflow the necessary flexibility. All these capabilities make WF to be an excellent environment for building and hosting business processes in the *Process* layer for the service-oriented enterprise computing.
- Another upcoming improvement with .NET Framework 3.0 is the Windows Communication Foundation, which is a unification of today's distributed technology stack on machine, cross machine, and cross Internet. As the Microsoft's next-generation programming platform for building, configuring and deploying over the network distributed services, WCF enables developers to build enterprise services with reliable communications that support sessions and transaction flow without deep system-level background knowledge about the distributed technologies being adopted, such as ASMX, WSE, MSMQ, .NET Remoting etc. Therefore, WCF makes developers more productive, because they only have to master a single programming model to build distributed application and services. Furthermore, with the support of the WS-* standards that Microsoft has developed jointly with its industry partners, WCF also ensures the broad interoperability with other technology platforms. Together with Web Services that are exposed by SAP NetWeaver Application Server, WCF provides all the necessary functionalities to build the *Service* layer in service-oriented enterprise computing.

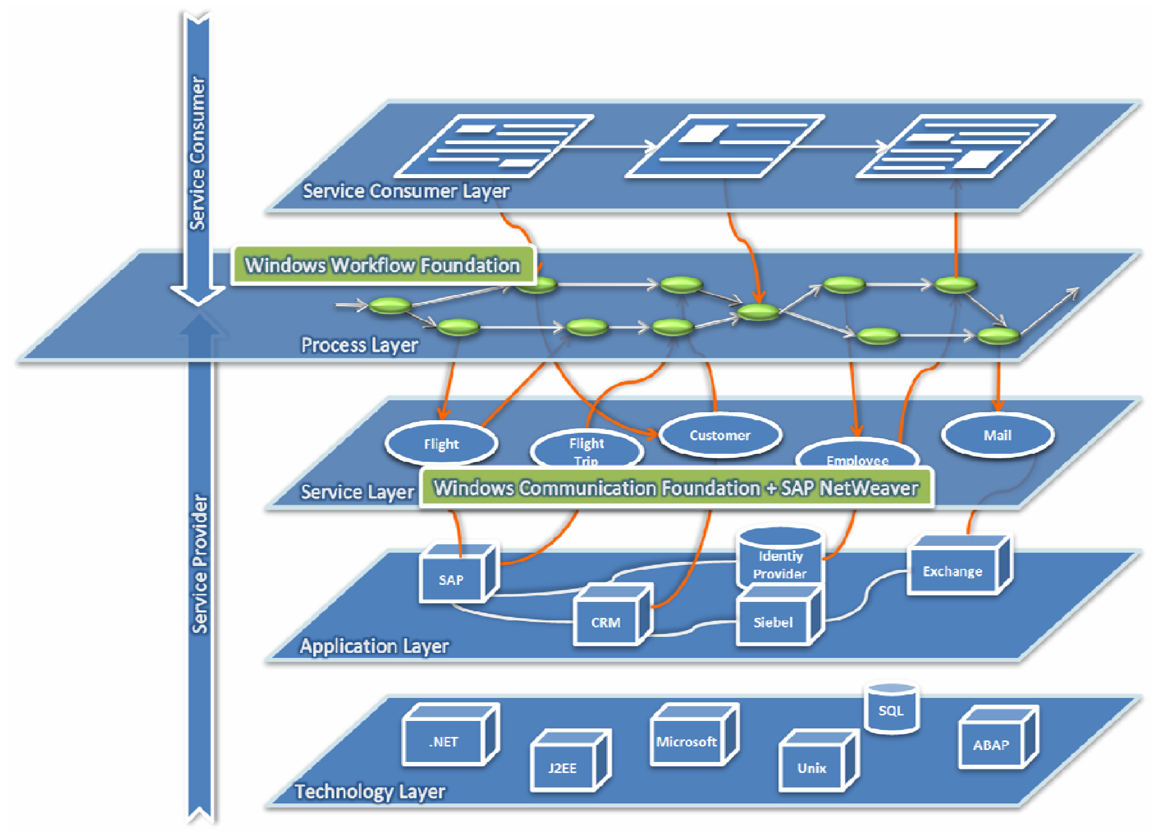


Figure 3: The Service-oriented Enterprise Computing with WinFX

Comparing to the heterogeneous computing infrastructure as shown in Figure 2, .NET Framework 3.0 provides a unified platform for building workflow-enabled distributed applications in compliance to service-oriented enterprise computing. As depicted in Figure 3, the WF provides the foundation for integrating workflows into client applications and workflow-enabled server applications on the Windows platforms. And the WCF together with SAP NetWeaver forms a unified service layer based on the WS-* standards in the service-oriented enterprise computing.

The Flight-booking Scenario

To demonstrate the capabilities of .NET Framework 3.0 and the new Web services platform of SAP NetWeaver, we use WCF, WF, and the integrated development tools to implement the flight-booking scenario from the SAP basic training. The flight-booking scenario itself is simple, as depicted in Figure 4. In the scenario, an employee can look for flight connections to some given arrival airport and submit booking request for the selected flight trip to his manager. A manager will be notified by emails if there is any booking request ready for review. He can approve or deny such booking requests, as the case may be. And in case that the submitted booking request is approved by a manager, the corresponding flight booking will be created in the travel agency's SAP system. The functionalities dealing with the flight-related operations, such as searching flight connections or creating flight trips, are available in SAP ECC as BAPI functions. And the most important business objects in SAP ECC are flight connections (SFLCONN) and flight trips (SFLTRIP).

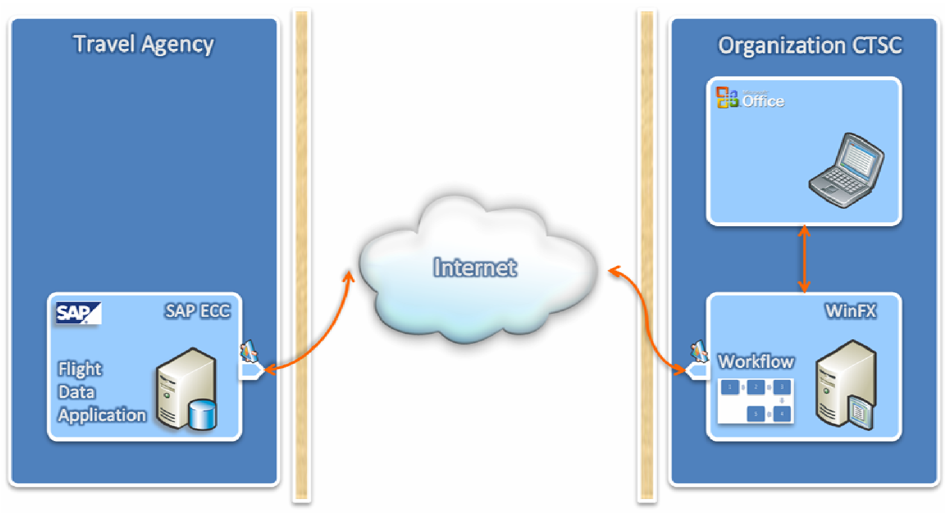


Figure 4: The Flight-booking Scenario

The implementation of the flight-booking scenario aims at demonstrating the following technologies or tools:

- Use Windows Workflow Foundation to model the booking-request process: the booking-request workflow contains activities from submission to approval/denial of booking requests will be modeled as a WF workflow. In this sample case, we demonstrate how to build an event-driven workflow with human intervention. We will also introduce how to extend the capabilities of WF by designing and implementing customized activities for the workflow. We demonstrate the way to combine external business functionalities into the workflow by consuming Web services from SAP. The workflow developed will be exposed as a Web service, too.
- Use Windows Communication Foundation to provide enterprise service: the booking-request workflow will be offered as enterprise service for demonstrating the capabilities of WCF to provide interoperable enterprise service. This demo case outlines the concept of WCF and shows how to combine WCF and WF together to build workflow-enabled enterprise services.
- Use Visual Studio Tools for Office to develop Office-based applications: Comparing to the VSTO 2003, the new VSTO 2005 provides further possibility to develop customized solutions with Windows Form controls for InfoPath and Outlook in addition to the improved programming APIs. In the flight-booking scenario, we use VSTO to implement *Smart Document* with Office Word 2003, which should demonstrates how to build customized ActionPane for Office Word and how to manipulate the content of a Word document from the custom ActionPane. Furthermore, we use VSTO to implement an Outlook-Plug-in and show how to bind external

functionalities as plug-in into Outlook and how the interaction between the external plug-in and Outlook works.

- Use Visual Studio 2005 to consume SAP Web services: all the flight-related operations provided by SAP ECC are accessible over Web services. In this flight-booking scenario, we demonstrate how to use Visual Studio 2005 to bind SAP Web services into the Visual Studio solutions and how to call these Web services from inside of the solutions, for instance, from a Workflow or from a Smart Document.

Prerequisites

In this whitepaper, we use the most recent software that are available at the moment as this whitepaper being written to develop the applications. As we already mentioned before, the .NET Framework 3.0 runtime environment is used in the demo scenario to host part of the implementation. In the following sections, we describe the software prerequisites for developing and running the demo scenario, so that you can easily reproduce the same environment as we have. The installation contains mainly two parts: the first part is the installation of development environment for .NET Framework 3.0 and Office applications; the second part is the installation of the SAP ECC 5.0 test system with a SAP NetWeaver 2004s ABAP Edition. It is recommended that you build the first part of the demo environment in one Windows environment and the SAP ECC 5.0 in another one, since we intend to construct a distributed flight-booking scenario over the Internet. You can use hardware virtualization software, such as Microsoft Virtual Server 2005 R2, to build the demo environment. You can download the software from http://www.microsoft.com/windowsserversystem/virtualserver/default.mspx free of charge. The implementation of the flight-booking scenario described in the remainder of this whitepaper is based on the two aforementioned Windows environments. In the following text, the Windows environment with the .NET Framework 3.0 & Office development environment is referred to as the "**WinFX**" environment and the other environment with the SAP installation is referred to as the "**SAP ECC**" environment.

Before we go on setting up the development environments, please note that, some of software being used are still under beta status by the time we wrote the whitepaper, and may cause occasionally application crash. Therefore, do not install these builds on the machine you operational depend on. However, the application crashes does not interfere with your development experience to get to know the functionality of the software being adopted.

Installation of the Development Environment for WinFX & Office

For developing .NET Framework 3.0 applications, you need the following components:

- **Development Environment**: you can use the Visual Studio 2005 full retail version as your development environment. In this whitepaper, the whole demo scenario is built using Visual Studio 2005 Team Suite, which is available to MSDN subscribers. As an alternative development environment, you can use the Visual Studio 2005 Express Editions with limitation, too. The free Visual Studio 2005 Express Editions can be downloaded from http://msdn.microsoft.com/vstudio/express/default.aspx. The Express Edition supports the development of WCF- and WPF-based applications. However, it does **NOT** support in developing Windows Workflow Foundation (WF) applications, since the current Visual Studio 2005 Extensions for Workflow Foundation beta is not compatible with the Visual Studio 2005 Express Editions.
- **.NET Framework 3.0 Runtime Environment**: for running .NET Framework 3.0 applications you developed, you need the **.NET Framework 3.0** runtime components. It is currently a pre-beta release and is available as Beta 2 under http://www.microsoft.com/downloads/details.aspx?FamilyId=4A96661C-05FD-430C-BB52-2BA86F02F595&displaylang=en.
- **Windows SDK**: The compatible Windows Software Development Kit (SDK) for the .NET Framework 3.0 runtime components Beta 2 is required for developing .NET Framework 3.0 applications. In addition to the APIs for the .NET Framework 3.0 technologies, the Windows SDK contains helpful documentation and code samples that you can consult during development. The compatible Windows SDK version is available under http://www.microsoft.com/downloads/details.aspx?FamilyId=13F8E273-F5EA-4B7B-B022-97755838DB94&displaylang=en
- .NET Framework 3.0 **Development Tools**: for integrated development of .NET Framework 3.0 applications in Visual Studio 2005, you need the Visual Studio Development Tools for .NET Framework 3.0, which install the design tools, the project templates and the .NET Framework 3.0 SDK documents into the final release of Visual Studio 2005. The .NET Framework 3.0 Development Tools that are compatible with the aforementioned runtime components can be downloaded from http://www.microsoft.com/downloads/details.aspx?FamilyId=31F9F15D-00E0-4241-8014-2F12679119AA&displaylang=en
- **Development Tools for Workflow Foundation**: the aforementioned .NET Framework 3.0 Development Tools do not contains development extensions for WF. To develop workflow-enabled applications using WF, you need a separate Visual Studio 2005 Extensions for Windows Workflow Foundation, which are available as Release Candidate 2 (RC2) under http://www.microsoft.com/downloads/details.aspx?

FamilyId=63A80A4B-BD27-4124-A2A5-61786ADB626E&displaylang=en

- **SQL Server 2005 Express Edition**: the Windows WF needs storage for saving tracking and state information at runtime. There are several options that one can use to serve as runtime storage for WF. In our flight-booking scenario, we will use the free available SQL Server 2005 Express Edition to save the runtime information of WF. The installation kit for SQL Server 2005 Express can be downloaded from http://msdn.microsoft.com/vstudio/express/sql/download/
- **Visual Studio Tools for Office 2005**: for developing managed solutions hosted in Office applications, such as Word, Excel, Outlook and InfoPath, Microsoft provides the Visual Studio Tools for Office (VSTO) 2005. Comparing to its predecessor, the VSTO 2005 provides improved developer tools and programming models. Furthermore, it enables usage of rich office UI and intuitive windows controls directly in the office applications. MSDN provides a good article about the improvements of VSTO 2005 in comparison to VSTO 2003. Visit http://msdn.microsoft.com/office/understanding/vsto/default.aspx?pull=/library/en-us/odc_vsto2005_ta/html/officewhatsnewinvsto2005.asp for more information.

After the steps listed above, you should have installed and configured the complete development environment for building .NET Framework 3.0 applications as well as Office-based applications. From now on, you can begin to build applications using WCF, WPF and WF technologies. In the following sections, we outline the installation of the backend SAP system and the necessary configuration to be made in the SAP system for the flight-booking scenario.

Installing SAP NetWeaver 04s (ABAP) Sneak Preview

For our flight-booking scenario, we use the SAP NetWeaver 2004s ABAP Edition with flight data application to serve as the backend SAP ECC system. The flight data application available in each SAP NetWeaver ABAP installation is intended for use in training and demos to demonstrate the SAP's integration technologies, especially the interface concept BAPI and IDoc. SAP provides the NetWeaver 2004s ABAP Sneak Preview Edition on the SDN for free download. You can find the installation kit on the SDN under https://www.sdn.sap.com/irj/sdn/downloaditem?rid=/library/uuid/cfc19866-0401-0010-35b2-dc8158247fb6. The installation of the NetWeaver 04s ABAP Edition is somewhat straightforward. SAP provides a detailed guide for downloading and installing the NetWeaver 04s ABAP Edition, which you can find on the same download page for the installation kit, too. After walking through the steps outlined in the guide, you should have a new SAP system with the ID "NSP" on your local PC installed. To access the SAP system you need the SAPGUI installed on your local PC. You can download the software from https://www.sdn.sap.com/irj/sdn/softwaredownload?download=ftp://ftp.sap.com/pub/sdn/devkits/netweaver/abap/50072743_4.zip&df=0.

In order to start the SAP system, use the "**SAP Management Console**" that you can find either directly on your desktop or from the program list in the "**Start**"-menu of your Windows. Select in the management console the SAP system – in our case the "NSP" System, right click it and select "**Start**" from the context menu shown. Follow instructions given in the installation guide to log on to the SAP system using the SAPGUI.

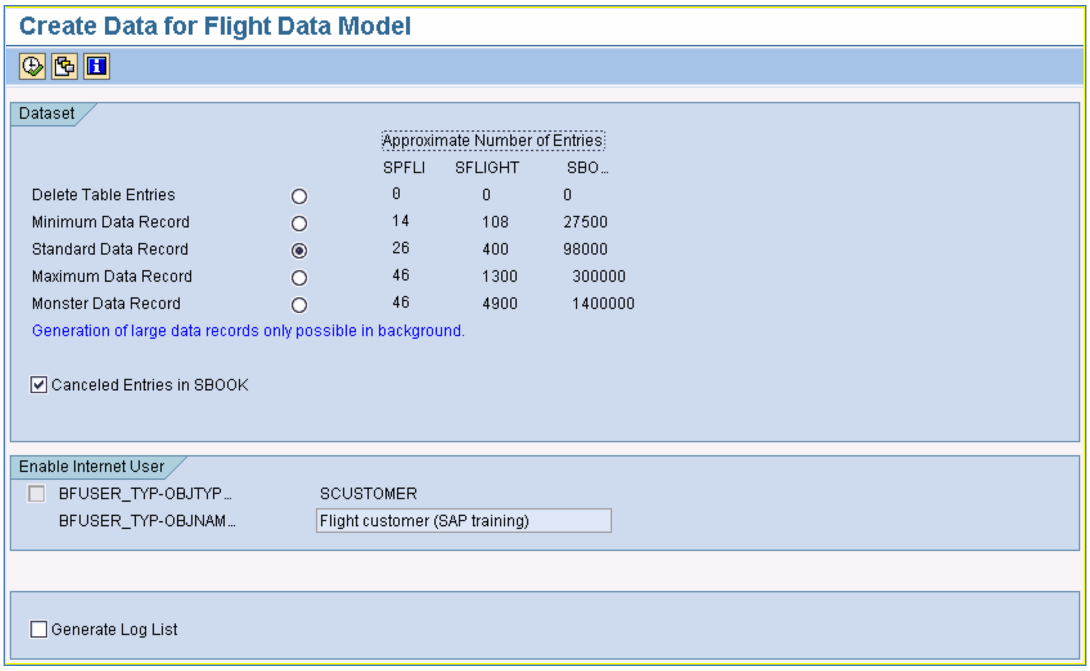
Another issue that has to be figured out is the validity of the license key in the SAP system being installed. The standard installation includes only a 30-days license key for evaluation. However, you can request another 90-days license key for your SAP NetWeaver 2004s ABAP installation from SAP. SAP provides a license key guide that documents the procedure you have to make to get a further license key on the SAP web site. You can download the guide from https://www.sdn.sap.com/irj/servlet/prt/portal/prtroot/docs/library/uuid/bb493f34-0801-0010-a3bc-ce2821492490.

Configuring SAP NetWeaver 04s for the flight-booking scenario

Each SAP NetWeaver 04s ABAP Edition contains the flight data application to demonstrate the integration technologies from SAP, especially the interface concept of BAPI and IDoc. The example flight data application supports several business processes around the artifacts of flight booking, such as flights, flight trips, flight connections and flight customers. The standard installation of SAP NetWeaver 04s contains all the business objects and the BAPI interfaces that you need to deal with the various flight-booking artifacts. Simply start the transaction "**BAPI**" in the SAPGUI and navigate in the standard hierarchical view to the node "**Basic Components**" -> "**ABAP Workbench, Java IDE and Infrastructure**". There you can find all the business objects and BAPIs for the flight data application.

However, the initial installation of SAP NetWeaver 04s does not generate the flight data that are compatible with the corresponding business objects and the BAPIs. To make sure that the necessary flight data for the scenario is available in your system, you can use the **Data Browser** (Transaction **SE16**) in the SAP system to check if the following tables are filled with test data: SCARR (airlines), SPFLI (flight schedules), SFLIGHT (flights), SBOOK (flight bookings), SCUSTOM (flight customers), SFLCONN (flight connections), SFLCONNPOS (route segments of flight connections), SFLTRIP (flight trips), SFLTRIPPOS (flight trip passengers), and SFLTRIPBOK (booking numbers for flight trips). If any data is missing in the table(s), you can set up the demo flight data application as follows:

1. Start the application **ABAP Editor** (Transaction **SE38**) in your SAP system
2. Execute the program **SAPBC_DATA_GENERATOR** in the editor
3. In the following dialog windows, select "**Standard Data Record**" from the configuration list and select "**Generate Log List**", if you want to see the log information during the generation process, as shown in the following figure. After that, start the generation process.
4. After the generation process has completed, you can check the data generated using the **Data Browser** again.



Create Data for Flight Data Model

Dataset

		Approximate Number of Entries		
		SPFLI	SFLIGHT	SBO...
Delete Table Entries	○	0	0	0
Minimum Data Record	○	14	108	27500
Standard Data Record	●	26	400	98000
Maximum Data Record	○	46	1300	300000
Monster Data Record	○	46	4900	1400000

Generation of large data records only possible in background.

☑ Canceled Entries in SBOOK

Enable Internet User

☐ BFUSER_TYP-OBJTYP... SCUSTOMER

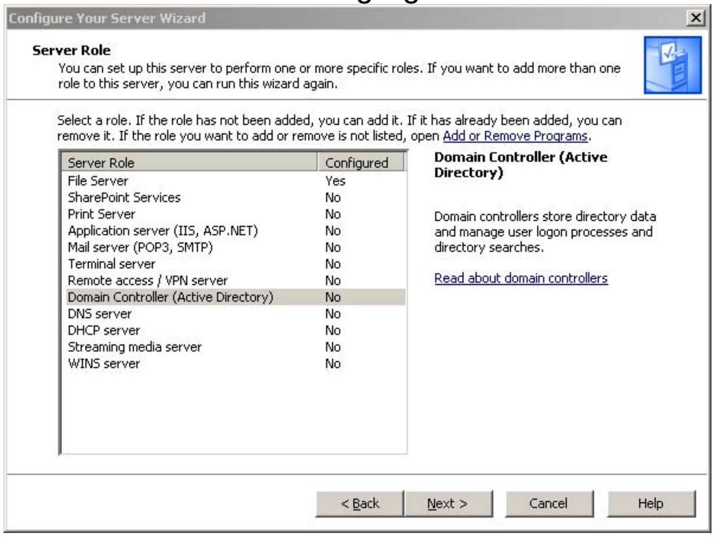
BFUSER_TYP-OBJNAM... Flight customer (SAP training)

☐ Generate Log List

Till now, you have set up the development environment for the flight-booking scenario. The following configuration is optional and is of interest for those, who want to reproduce the exact demo environment as we have for this whitepaper. To support the business processes in the flight-booking scenario, we need an Email-server to support the email interactions between the employee and his manager. Furthermore, we need two active users in our scenario, one as the employee and the other one as his manager. In the following section, we will configure the "SAP ECC" Windows environment to support the demo scenario:

1. Set up a **Domain Controller**: use the server configuration wizard to add a domain controller to the "**SAP ECC**" environment.
   a. In the "**SAP ECC**" environment, go to "**Start**" -> "**All Programs**" -> "**Administrative Tools**" and click "**Configure Your Server Wizard**" to start the server configuration wizard.

b. In the "**Server Role**" Dialog window, select the server role "**Domain Controller (Active Directory)**" from the list and click "**Next**", as shown in the following figure.


Configure Your Server Wizard

Server Role

You can set up this server to perform one or more specific roles. If you want to add more than one role to this server, you can run this wizard again.

Select a role. If the role has not been added, you can add it. If it has already been added, you can remove it. If the role you want to add or remove is not listed, open Add or Remove Programs.

Server Role	Configured
File Server	Yes
SharePoint Services	No
Print Server	No
Application server (IIS, ASP.NET)	No
Mail server (POP3, SMTP)	No
Terminal server	No
Remote access / VPN server	No
Domain Controller (Active Directory)	No
DNS server	No
DHCP server	No
Streaming media server	No
WINS server	No

Domain Controller (Active Directory)

Domain controllers store directory data and manage user logon processes and directory searches.

Read about domain controllers

c. In the following dialog window shown, specify the name of the new domain as "**CTSC.org**" and click "**Next**" to continue the configuration. In the following dialogs, always use the standard configuration as given in the dialog and click "**Next**".

d. Restart the "**SAP ECC**" environment after the installation has completed.

2. Set up the active users in the Active Directory

a. Set up the manager with the name "**Bill Hillman**" in the Active Directory

i. Start the "**Active Directory Users and Computers**" management console (navigate in "**SAP ECC**" environment to "**Start**" –> "**All Programs**" -> "**Administrative Tools**" and click there "**Active Directory Users and Computers**")

ii. Navigate in the menu "**Action**" –> "**New**" -> "**User**" to add a new user

iii. In the "**New Object – User**" dialog, fill the text boxes for "**Bill Hillman**". Specify the logon name as "**hillman**" and click "**Next**" to continue.

iv. Specify a password for Bill Hillman and select "**Password never expires**". Click "**Next**" to create the new user object in Active Directory.

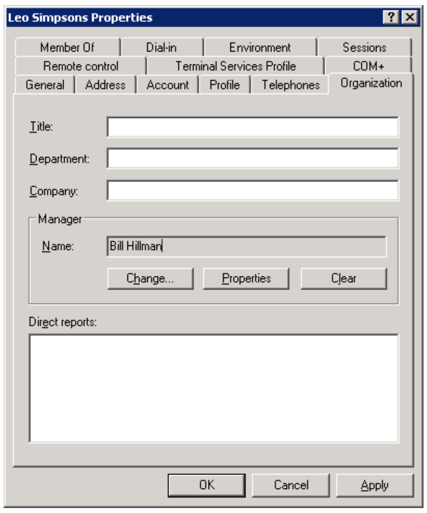
b. Set up the employee with the name "**Leo Simpsons**" in the Active Directory

i. Repeat the steps i ~ iv as given in the last point **a** and create a new user "**Leo Simpsons**" with the logon name "**simpsons**" in the Active Directory.

ii. Right-click the user object "**Leo Simpsons**" in the console and select "**Properties**" in the context menu to show the user properties.

iii. Switch to the register tab "**Organization**" in the properties window.

iv. Change the value of the property "**Manager**" to "**Bill Hillman**" by click the "**Change**" button and specify the value "**Bill Hillman**" in the popup dialog window, as shown in the following figure



Title:

Department:

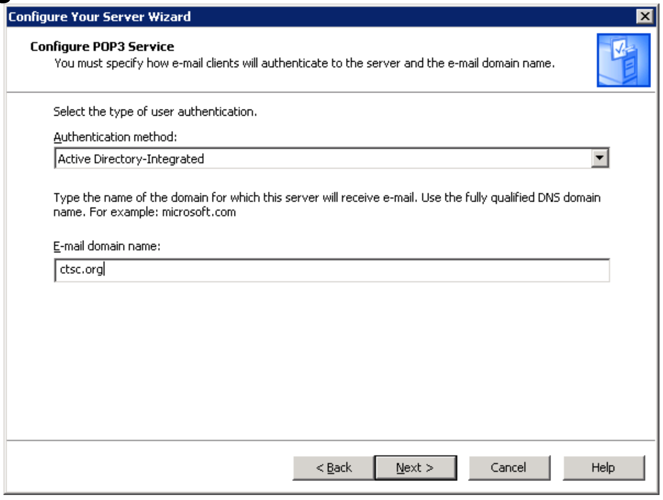
Company:

Manager

Name: Bill Hillman

Direct reports:

3. Set up the mail servers: use the server configuration wizard to add a domain controller to the "**SAP ECC**" environment.
   a. In the "**SAP ECC**" environment, go to "**Start**" -> "**All Program**" -> "**Administrative Tools**" and click "**Configure Your Server Wizard**" to start the server configuration wizard.
   b. In the "**Server Role**" Dialog window, select the server role "**Mail Server (POP3, SMTP)**" from the list and click "**Next**".
   c. Select in the following dialog "**Active Directory-Integrated**" as the **Authentication method** and set the **Email domain name** as "**ctsc.org**".



Configure POP3 Service

You must specify how e-mail clients will authenticate to the server and the e-mail domain name.

Select the type of user authentication.

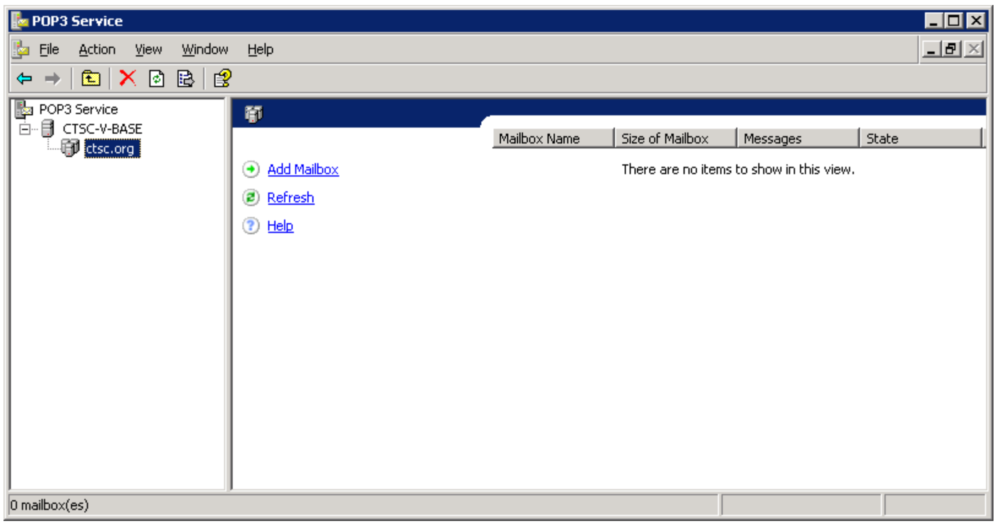
Authentication method: Active Directory-Integrated

Type the name of the domain for which this server will receive e-mail. Use the fully qualified DNS domain name. For example: microsoft.com

E-mail domain name: ctsc.org

   d. In the following dialogs, always use the standard configuration as given and click "**Next**" to complete the wizard.
4. Set up the mailbox for the demo users

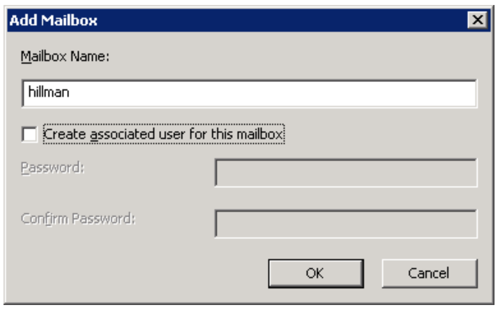
a. In the “**SAP ECC**” environment, go to “**Start**” -> “**All Programs**” -> “**Administrative Tools**” and click “**POP3 Service**” to start the server configuration wizard.
b. In the console, select the mail domain that you have created before. In our case, it is the “CTSC.org”.



There are no items to show in this view.

0 mailbox(es)

c. In the console menu, select “**Action**” -> “**New**” -> “**Mailbox...**” to create a new mailbox for Bill Hillman
d. In the popup dialog, enter “**hillman**” as **Mailbox Name**, and unselect the check-box “**Create associated user for this mailbox**”, see the following figure. If the mailbox has been created successfully, a message box with the new account name and the corresponding mail server will be shown. Repeat this step to create a mailbox for Leo Simpsons with the mailbox name “**simpsons**”.

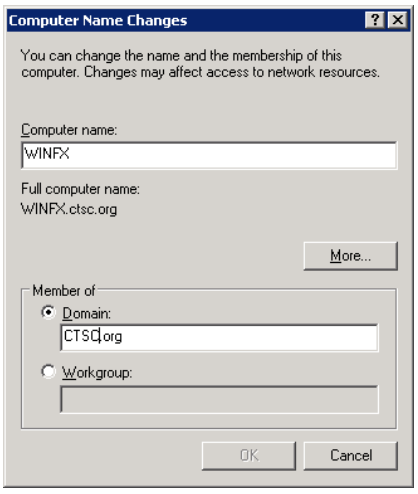


Mailbox Name: hillman

☐ Create associated user for this mailbox

5. Add the “**WinFX**” Environment to the Domain Controller “**CTSC.org**”
a. In the “**WinFX**” environment, go to “**Start**” -> “**Control Panel**” and click “**System**” to open the **System Properties** dialog.
b. Change to the register tab “**Computer Name**”
c. Click the Button “**Change**” to open the “**Computer Name Changes**” dialog

d. In the dialog, change the value of the text box “**Domain**” to “**CTSC.org**”, so that the “**WinFX**” Environment will be a domain member in “**CTSC.org**” in the future, see the following figure.



You can change the name and the membership of this computer. Changes may affect access to network resources.

Computer name: WINFX

Full computer name: WINFX.ctsc.org

Member of
(● Domain:) CTSC.org
○ Workgroup:

e. Click “OK” to close the dialog
f. You may want to restart the “**WinFX**” environment after having changed the system properties.

Until now, you have completely installed and configured the demo environments for the flight-booking scenario. In the following sections, we will introduce the implementation of the flight-booking scenario bottom-up systematically. We begin with the backend SAP ECC system and describe how to expose BAPI/RFC as Web services for external access. Afterwards, we look at the service as well as process layer, and discuss how to use WCF and WF to build workflow-enabled middleware. At last, we work on the client applications with Office 2003 and show, how to integrate Office client applications into the workflow to drive human-centric business processes.

Accessing SAP BAPI/RFC using Web Services

Since the release of SAP Web Application Server 6.20, SAP has made instant efforts to offer its business functionalities via the standard-based Web services interface. SAP Web Application Server allows customers to expose their business capabilities via Web services to integrate enterprise applications and to drive collaborative business processes across organizational boundaries. In the following, we review the SAP's Web service technology stack with SAP NetWeaver and demonstrate how to expose SAP business functionalities as Web services.

Overview Web Services on SAP NetWeaver

SAP NetWeaver Application Server provides both a development and runtime environment for Web services, as depicted in Figure 5.

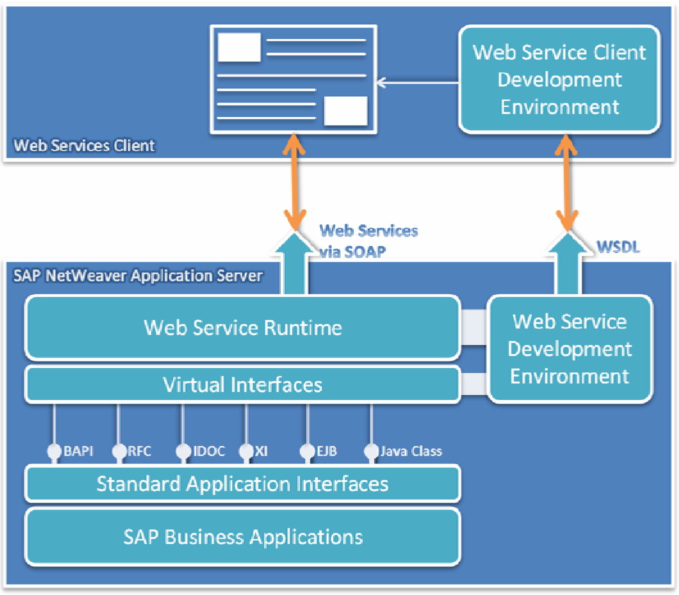

Web Service Client Development Environment
Web Services Client
Web Services via SOAP
WSDL
SAP NetWeaver Application Server
Web Service Runtime
Virtual Interfaces
Web Service Development Environment
BAPI
RFC
IDOC
XI
EJB
Java Class
Standard Application Interfaces
SAP Business Applications

Figure 5: Web Services on SAP NetWeaver Application Server[10]

The implementation of business capabilities for Web services is taken place in the normal SAP business applications with a corresponding development environment, such as ABAP Workbench or Java Developer Studio. It is the same approach as developing normal business applications and no Web service specific implementation or configuration is required at this point, apart from a well-defined interface. The business capabilities being built in the business applications are exposed for external access via the standard application interfaces, among other things, the BAPI, RFC, IDoc interfaces.

[10] Karl Kessler, *Web Service Technology for SAP NetWeaver*, https://www.sdn.sap.com/irj/servlet/prt/portal/prtroot/docs/library/uuid/f65ecf90-0201-0010-94b0-c9983be54c67

The encapsulation from normal business capabilities to Web services takes place in the Web service development environment, which is normally the same development environment as for business applications, namely ABAP Workbench for ABAP Web services or Java Developer Studio for Java Web services. Each SAP Web service always has the following three functional parts:

- *Virtual Interface*: each virtual interface defines the selected business capabilities that the Web service encapsulates and the customization of the interface parameters
- *Web Service Definition*: a Web service definition defines a Web service and the mapping between the Web service and the virtual interface. Furthermore, a Web service definition defines a set of Web service-related features, such as authentication, session management, etc.
- *Web Service Configuration*: the Web service configuration controls the behavior of the SAP Web services, e.g., the transport way or the security profile of the Web services

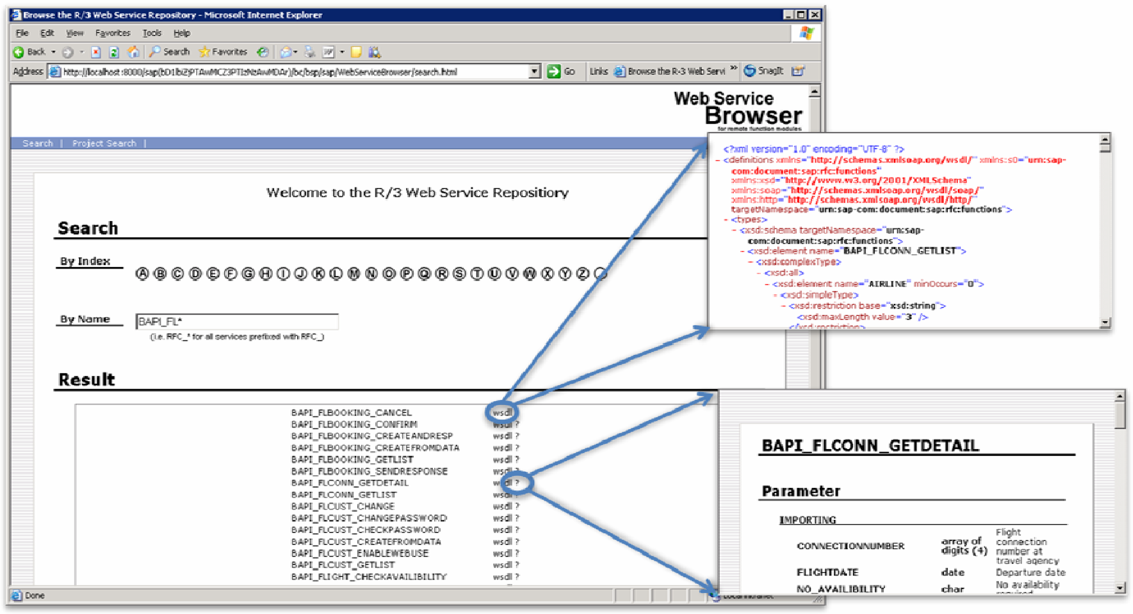
The *Development Environment* maintains all the functional parts of the Web service. Furthermore, the *Development Environment* is responsible for the generation of the WSDL documents, which are shared with the Web service clients to retrieve detailed information about the target Web service. The *Web Service Runtime* in SAP NetWeaver Application Server manages the SAP Web services by reading configuration information from the *Web Service Configuration* and provides the runtime environment that complies to the configuration settings. At runtime, the *Web Service Runtime* receives SOAP request and forwards the request via the standard application interface to the desired business capability. After the request has been processed by the business functionality, it gets the result of the execution returned by the business functionality and forwards the result back to the Web service client.

Currently, SAP provides native support for both Java-based and ABAP-based Web services. For more information about developing Web services on the top of SAP NetWeaver as well as the other topics about SAP Web services, for instance, security or interoperability with other Web services platforms, you can visit the Web service column at SAP Developer Network under:
https://www.sdn.sap.com/irj/sdn/developerareas/esa/webservices

In the following sections, we demonstrate two different ways to expose SAP business functionalities via Web services.

Accessing Web Services from SOAP Processor in SAP NetWeaver 6.20

Since the release of 6.20, SAP NetWeaver Application Server provides a set of out-of-box Web services via the SOAP processor. However, the SOAP processor is nothing else as a SOAP message translator and forwarder. Besides the standard Web services protocols, it does not provide further support for additional Web services standards for e.g. security or transport. In the 6.40 release, this interface remains unchanged and exposes all the RFC-enables function modules (BAPIs) as Web services. To get a list of Web services for the BAPIs, navigate to the following web address in a browser:
http://[domain]:[port]/sap/bc/bsp/sap/WebServiceBrowser/search.html
In the web page shown, you can search for the Web services by using the BAPI name and get the list of Web services that matches the name you entered. It may look like Figure 6. With the links “wsdl” and “?” shown next to the Web services found in the result, you can get either the WSDL document or the detailed documentation for the corresponding Web services.


Browse the R/3 Web Service Repository - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites
Address http://localhost:8000/sap(bD1lbiZPTAwMCZ3PTIzNzAwMDAr)/bc/bsp/sap/WebServiceBrowser/search.html
Go Links Browse the R-3 Web Servi SnagIt
Web Service Browser for remote function modules
Search | Project Search |
Welcome to the R/3 Web Service Repository
Search
By Index
Ⓐ Ⓑ Ⓒ Ⓓ Ⓔ Ⓕ Ⓖ Ⓗ Ⓘ Ⓙ Ⓚ Ⓛ Ⓜ Ⓝ Ⓞ Ⓟ Ⓠ Ⓡ Ⓢ Ⓣ Ⓤ Ⓥ Ⓦ Ⓧ Ⓨ Ⓩ
By Name
(i.e. RFC_* for all services prefixed with RFC_)
Result
Done
Local intranet

By Name: BAPI_FL*

BAPI_FLBOOKING_CANCEL	wsdl ?
BAPI_FLBOOKING_CONFIRM	wsdl ?
BAPI_FLBOOKING_CREATEANDRESP	wsdl ?
BAPI_FLBOOKING_CREATEFROMDATA	wsdl ?
BAPI_FLBOOKING_GETLIST	wsdl ?
BAPI_FLBOOKING_SENDRESPONSE	wsdl ?
BAPI_FLCONN_GETDETAIL	wsdl ?
BAPI_FLCONN_GETLIST	wsdl ?
BAPI_FLCUST_CHANGE	wsdl ?
BAPI_FLCUST_CHANGEPASSWORD	wsdl ?
BAPI_FLCUST_CHECKPASSWORD	wsdl ?
BAPI_FLCUST_CREATEFROMDATA	wsdl ?
BAPI_FLCUST_ENABLEWEBUSE	wsdl ?
BAPI_FLCUST_GETLIST	wsdl ?
BAPI_FLIGHT_CHECKAVAILIBILITY	wsdl ?

```
<?xml version="1.0" encoding="UTF-8" ?>
- <definitions xmlns="http://schemas.xmlsoap.org/wsdl/" xmlns:s0="urn:sap-
    com:document:sap:rfc:functions"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
    xmlns:http="http://schemas.xmlsoap.org/wsdl/http/"
    targetNamespace="urn:sap-com:document:sap:rfc:functions">
  - <types>
    - <xsd:schema targetNamespace="urn:sap-
        com:document:sap:rfc:functions">
      - <xsd:element name="BAPI_FLCONN_GETLIST">
        - <xsd:complexType>
          - <xsd:all>
            - <xsd:element name="AIRLINE" minOccurs="0">
              - <xsd:simpleType>
                - <xsd:restriction base="xsd:string">
                    <xsd:maxLength value="3" />
```

BAPI_FLCONN_GETDETAIL

Parameter

IMPORTING

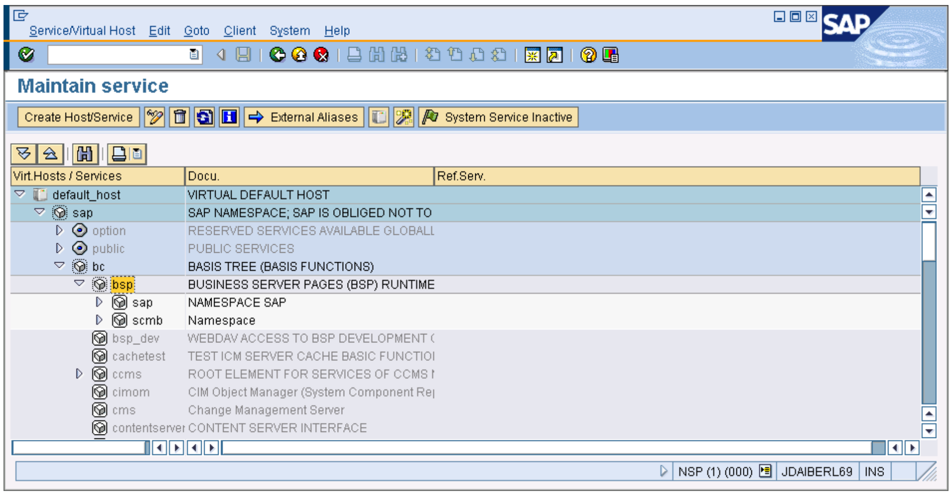
CONNECTIONNUMBER	array of digits (4)	Flight connection number at travel agency
FLIGHTDATE	date	Departure date
NO_AVAILIBILITY	char	No availability

Figure 6: Web Service Repository for BAPIs (since SAP WAS 6.20)

To access BAPI functions via Web services in this way, both the *Business Server Pages Runtime* and the *ICF Service for SOAP Handler* must be activated. If you cannot access the list of out-of-box Web services under the URL listed above, you have to check up, if all the services necessary are enabled in ICF. To do it, follow the following instructions:

1. Start the application “**Maintain Service**” (Transaction **SICF**) in your SAP system
2. Navigate to the service “**default_host->sap->bc->bsp**” and check up if the service *Business Server Pages Runtime* is enabled.

If not, right click the node "**bsp**" and select "**Activate Service**" from the context menu, as shown in the following figure.



Maintain service

Virt.Hosts / Services	Docu.	Ref.Serv.
default_host	VIRTUAL DEFAULT HOST	
sap	SAP NAMESPACE; SAP IS OBLIGED NOT TO	
option	RESERVED SERVICES AVAILABLE GLOBALL	
public	PUBLIC SERVICES	
bc	BASIS TREE (BASIS FUNCTIONS)	
bsp	BUSINESS SERVER PAGES (BSP) RUNTIME	
sap	NAMESPACE SAP	
scmb	Namespace	
bsp_dev	WEBDAV ACCESS TO BSP DEVELOPMENT (	
cachetest	TEST ICM SERVER CACHE BASIC FUNCTIOI	
ccms	ROOT ELEMENT FOR SERVICES OF CCMS I	
cimom	CIM Object Manager (System Component Re|	
cms	Change Management Server	
contentserver	CONTENT SERVER INTERFACE	

NSP (1) (000) JDAIBERL69 INS

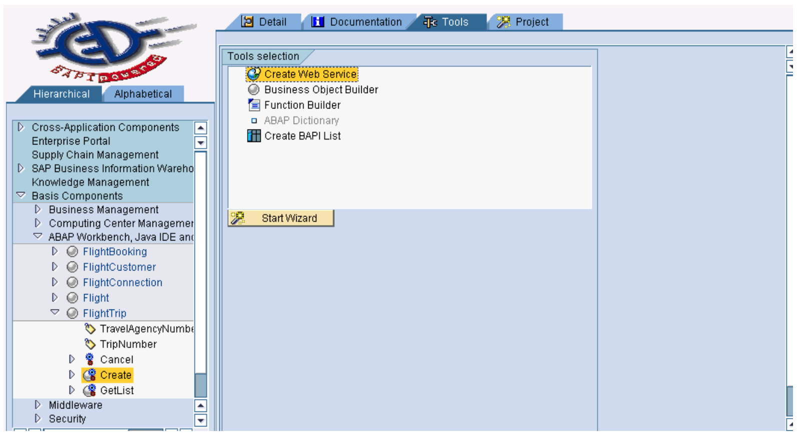
3. Repeat the last step to check, if the **service for SOAP HTTP Handler** (**default_host->sap->bc->soap**) is enabled. If not, activate it, too.
4. Now you should be able to access the Web service repository via browser.

In this way, you can access all the BAPI functions via Web services calls. It is convenient for people to call SAP business functionalities without using the proprietary RFC protocols. However, this way is only valid for built-in BAPI functions in the SAP system. Customized BAPI functions cannot be accessed in this way. Furthermore, this approach does only support the basic Web services standards with basic authentication. In other words, the client can only authenticate itself against the SAP Web services using a username/password credential. Other advanced client authentication mechanisms like X.509 certificates are not supported. If there is no security support from the transport layer in the network stack, the SOAP messages transferred between the client and the Web service are in plain text, which is not the best way to transfer sensible data across the Internet. Although one can use transport layer encryption, such as Secure Sockets Layer (SSL), to secure the message throughout the transport way, this method is still limited, because we need a secure end-to-end connection between the client and the Web service and the transport layer encryption provides only a point-to-point connection. Moreover, this approach does not support customization of the Web service interface or the parameters. This limitation is solved with the new approach for exposing Web services in the SAP NetWeaver Application Server 6.40, which we introduce in the next section.

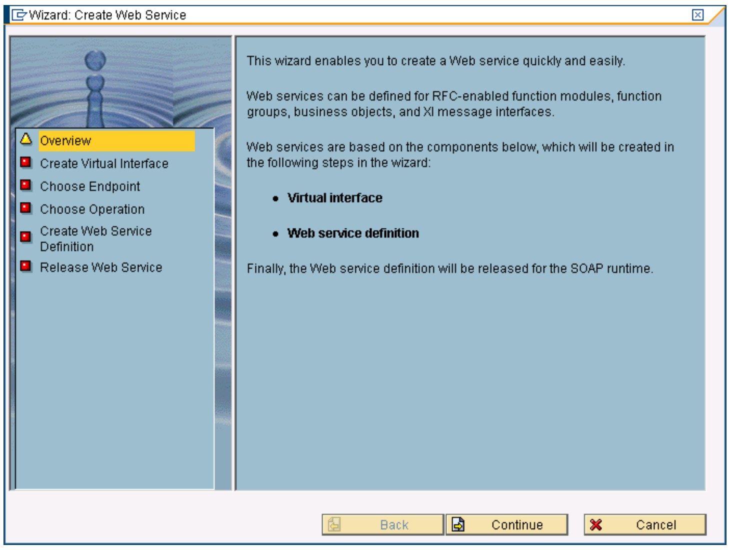
Accessing Web Services from SOAP Runtime in SAP NetWeaver 6.40

The new Web service approach of the SAP NetWeaver Application Server 6.40 provides more possibilities to create customized Web service from each BAPI functions that are available in the SAP ECC. The Web services created with the new approach contain exactly the three parts that we discussed in the introduction section: *Virtual Interface, Web Service Definition, and Web Service Configuration*. In the following, we demonstrate this new Web service concept by creating a Web service that expose the BAPI function BAPI_FLTRIP_CREATE for creating flight trip as Web services. To facilitate the task of exposing functionalities as Web services, SAP provides a Web service creation wizard for use. There are several ways to initialize the wizard. Since we have determined the BAPI function to expose, we will start the wizard through the **BAPI Explorer**. It is e.g. also possible to start the wizard through the application **Object Navigator** or the transaction **WS_WZD_START**.

1. Start the application **BAPI Explorer** (transaction **BAPI**) in your SAP system
2. Navigate in the *hierarchical* view to **Basic Components** -> **ABAP Workbench, Java IDE and Infrastructure** -> **Flight Trip** and select the method "**Create**"
3. In the workbench, switch to the tab "**Tools**", as shown in the following figure.



4. Select the tool "**Create Web Service**" and click "**Start Wizard**" to continue. The Web service creation wizard will be shown in a new window, as illustrated in the following figure. Click "**Continue**" to start the wizard.



This wizard enables you to create a Web service quickly and easily.

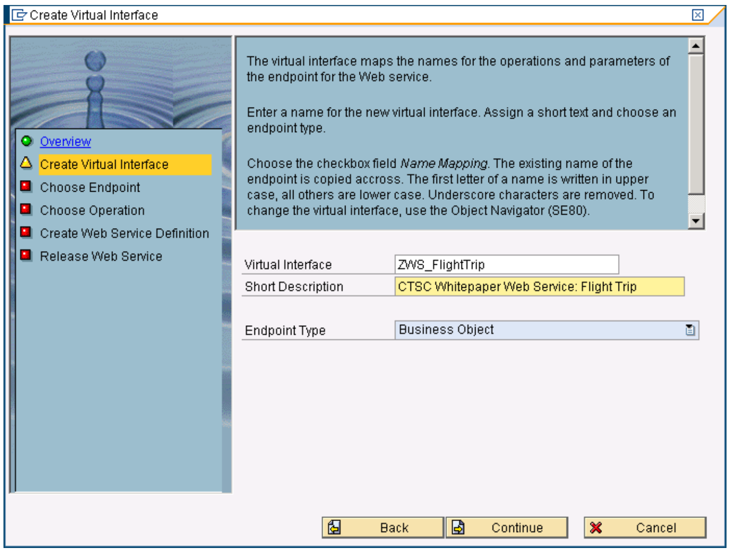
Web services can be defined for RFC-enabled function modules, function groups, business objects, and XI message interfaces.

Web services are based on the components below, which will be created in the following steps in the wizard:

- **Virtual interface**
- **Web service definition**

Finally, the Web service definition will be released for the SOAP runtime.

5. In the following dialog, you are asked to create a new Virtual Interface for the Web service. Enter a meaningful name for the virtual interface, e.g. "**ZWS_FlightTrip**" and write some short declarative description for the interface. Leave the "**Business Object**" unchanged as the **Endpoint Type**.



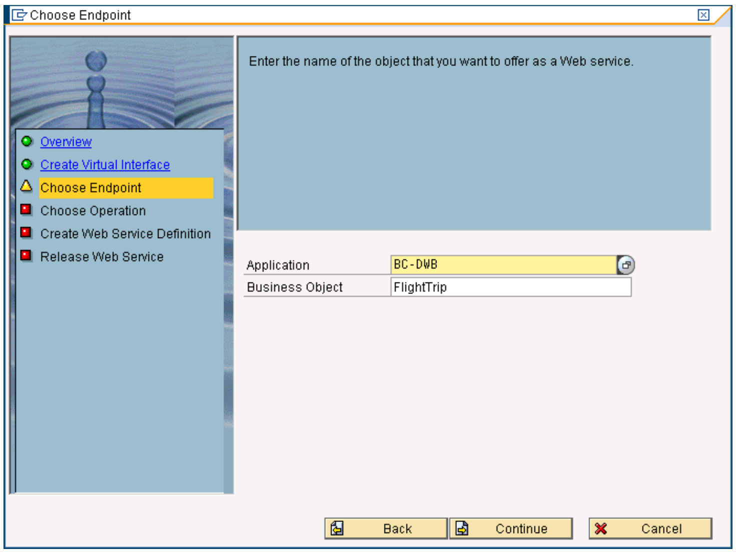
The virtual interface maps the names for the operations and parameters of the endpoint for the Web service.

Enter a name for the new virtual interface. Assign a short text and choose an endpoint type.

Choose the checkbox field *Name Mapping*. The existing name of the endpoint is copied accross. The first letter of a name is written in upper case, all others are lower case. Underscore characters are removed. To change the virtual interface, use the Object Navigator (SE80).

Virtual Interface	ZWS_FlightTrip
Short Description	CTSC Whitepaper Web Service: Flight Trip
Endpoint Type	Business Object

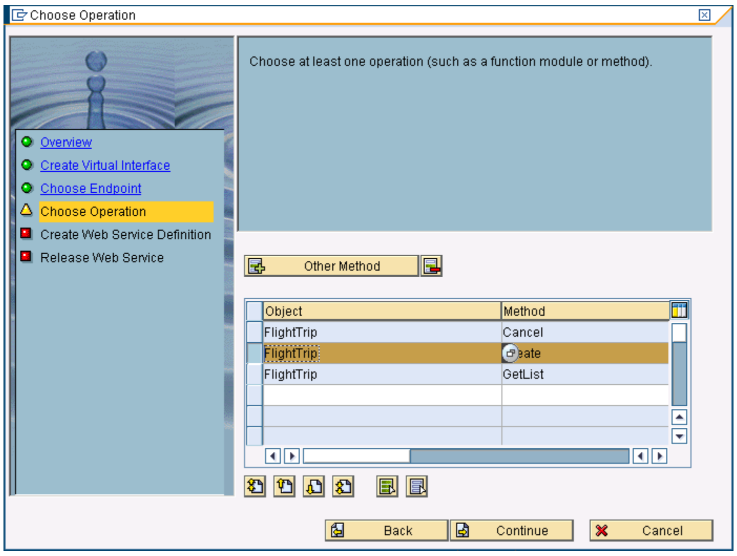
6. In the next screen, the endpoint for the new Web services will be specified. This information will be used later by the Web service runtime to determine which BAPI should be called to process the request. In the screen, enter the application "**BC-DWB**" as the **Application**, which represents the ABAP application "ABAP Workbench, Java IDE and Infrastructure". Click "Continue" to switch to next screen.


Choose Endpoint

Enter the name of the object that you want to offer as a Web service.

Application	BC-DWB
Business Object	FlightTrip

7. Now you are asked to choose the Operations, which are available later as Web methods in the new Web service. In our case, we want to make the operation “**Create**” later via Web service available. Select the Method “**Create**” from the list and click “**Continue**” to go on. If you want to make some other BAPI functions available via the same Web service, you can add the corresponding method into the list in this step. To do it, click “**Other Method**” and add the corresponding business object and its method to the list.

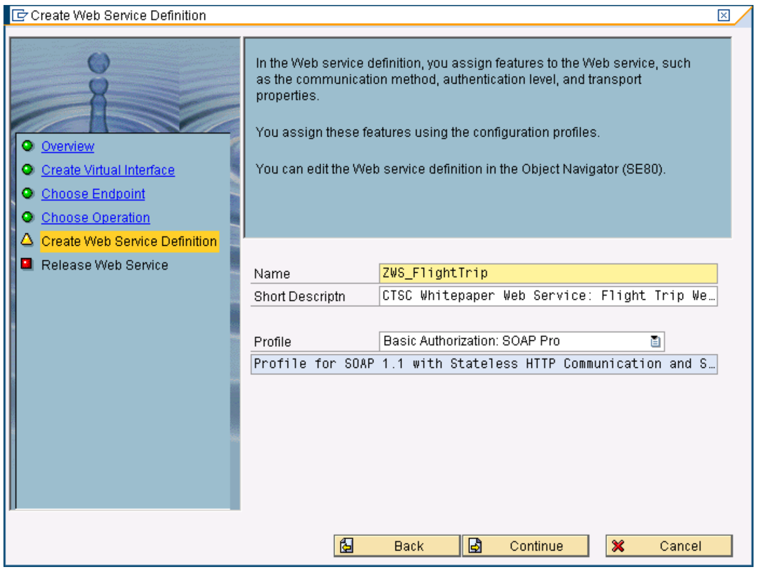

Choose Operation

Choose at least one operation (such as a function module or method).

Object	Method
FlightTrip	Cancel
FlightTrip	Create
FlightTrip	GetList

8. In this step, the wizard creates the corresponding **Web Service Definition** for the new Web service. Enter a name and a declarative description for the new Web Service Definition. You can also specify the security profile for the new Web service in this step. You can either choose the basic authorization profile “**Basic Authorization: SOAP Pro**” with authentication through username and password or choose the advanced authorization profile “**Secure SOAP Profile**” with authentication through client

certificate and secured transport using Secure Socket Layer (SSL) protocol. In our demo, we only use the basic authorization profile to keep the scenario simple.


Create Web Service Definition

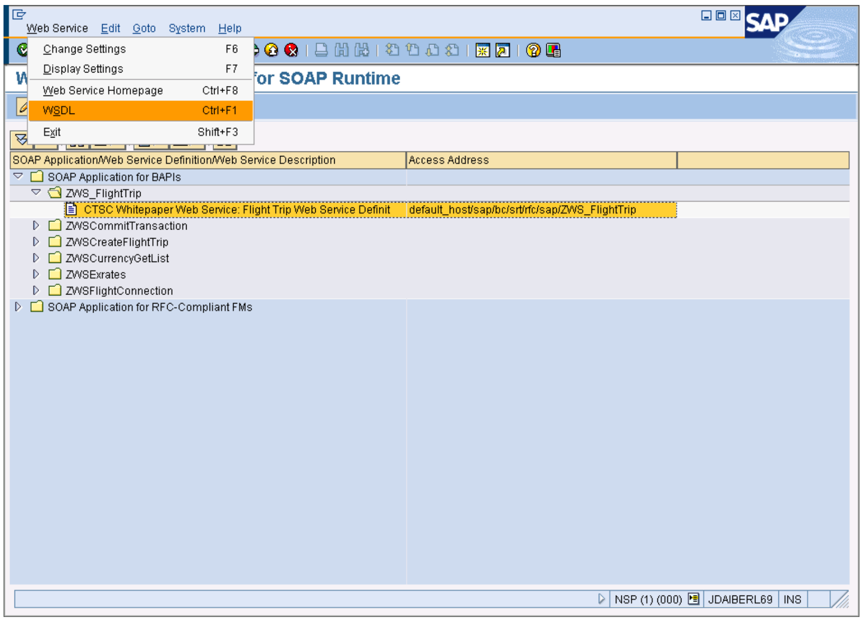
In the Web service definition, you assign features to the Web service, such as the communication method, authentication level, and transport properties.

You assign these features using the configuration profiles.

You can edit the Web service definition in the Object Navigator (SE80).

Name	ZWS_FlightTrip
Short Descriptn	CTSC Whitepaper Web Service: Flight Trip We…
Profile	Basic Authorization: SOAP Pro
	Profile for SOAP 1.1 with Stateless HTTP Communication and S…

9. In the last step, click "**Complete**" to release the new Web service.
10. To verify that the Web service has been successfully created and released, you can check the new Web service in the administration tool for SOAP runtime. Start the application "**Web Service Administration for SOAP Runtime**" (transaction **WSADMIN**) in your SAP system
11. Expand the node "SOAP Application for BAPIs" in the administration tool
12. You should be able to see the new Web service "**ZWS_FlightTrip**" in the list. Expand the node and you find there the Web Service Definition for "**ZWS_FlightTrip**" together with the access address for the new Web service.
13. To get the WSDL document for the new Web service, go to the menu "**Web Service**" -> "**WSDL**" and click it. The WSDL document is displayed in a new window of your default Web browser.



SOAP Application/Web Service Definition/Web Service Description	Access Address
SOAP Application for BAPIs	
ZWS_FlightTrip	
CTSC Whitepaper Web Service: Flight Trip Web Service Definit	default_host/sap/bc/srt/rfc/sap/ZWS_FlightTrip
ZWSCommitTransaction	
ZWSCreateFlightTrip	
ZWSCurrencyGetList	
ZWSExrates	
ZWSFlightConnection	
SOAP Application for RFC-Compliant FMs	

Now you have finished creating a new Web service from the BAPI for flight trip. Besides the administration tool introduced above, you can use a set of other tools for maintaining as well as modifying your Web services. For example, you can check all the Web service definitions and virtual interfaces directly in the Object Navigator (transaction SE80). You can specify there other features for your Web services. Because this is out-of-scope of this whitepaper, please refer to the SAP documentation http://help.sap.com/saphelp_erp2004/helpdata/en/e5/a68d10f4eb894087fc9c1c3f9ae433/frameset.htm for more information.

With the Web services exposed by SAP NetWeaver, we can realize a set of business functionalities. However, to support an active business process in the enterprise, one needs more than the business functionalities in the backend system. One needs something that can integrate all the business functionalities to build collaborative business processes. In the next section, we check out how to create collaborative business processes based on the SAP Web services and .NET Framework 3.0.

Building Business Processes using WF®

In the keynote at this year's Microsoft CEO Summit, Microsoft's chairman Bill Gates has outlined the current problem for enterprise information workers as a twofold problem: information overload/underload. Cited from his Email after the keynote, "Faced with the endless deluge of data that is generated every second of every day, how can we hope to keep up? And in the struggle to keep up, how can we stay focused on the tasks that are most important and deliver the greatest value?"[11]. The way out of this situation is more than just better search tools. It requires a comprehensive approach to "enterprise information management that spans information creation, collection and use". One of the biggest challenges for information workers is the fact that the enterprise data is often distributed in the different backend systems. In order to complete a single task, an information worker has to access several data sources in the enterprise manually using different client applications as UI. With the new programming model .NET Framework 3.0, Microsoft tries to deliver the basis for building enterprise information access solutions that give the information workers to access the information in a unified way and increase so that the productivity of them. As two of the new technologies that Microsoft intends to roll out in the next few months to solve the problem of enterprise information access, Windows Communication Foundation and Windows Workflow Foundation provide the basis for unified information access across the enterprise and across the partners along the business process. In this section, we introduce the concept and the characteristics of both WWF and WCF by walking through the implementation of the flight-booking scenario and explain how they can help to build enterprise applications with different backend systems for application workers.

Windows Workflow Foundation

Windows Workflow Foundation (WF) provides a rich programming model for building the semantics of processes in terms of activities into an application. The set of *Activities* that coordinate people and software, for instance, the activities *GetFlightList* or *BookFlightTrip* in our flight-booking scenario, are organized into a *Workflow*. WF provides a way for organizing activities into a workflow with additional support for program control flow, transactions, synchronization, exception handling and interactions with other applications. Furthermore, it provides a set of services for advanced workflow management, including workflow persistence, compensating transaction, activity tracking, runtime tracing etc. All these services are controlled by the

[11] You can read the full text of this email at Microsoft's web site: http://www.microsoft.com/mscorp/execmail/

workflow runtime, which can be hosted in any CLR application domains and so that can be embedded into any .NET applications.

In this section, we address the following topics:

- The basic concept of WF: the architecture and the components of WF
- The Extensibility of WF: how to extend WF by developing customized workflow activity
- Interaction with external application/person: how a workflow can be controlled by external application/person
- Hosting workflows in .NET application: how to build workflows into your applications

We only covered the key part of the WF in this whitepaper. Meanwhile, there are several great information sources for getting started with the WF technology. Besides the .NET Framework 3.0 developer center for Workflow Foundation on MSDN (available under http://msdn.microsoft.com/winfx/technologies/workflow/default.aspx), Paul Andrew et al., who are the key members of the Microsoft Team responsible for developing WF, have written a book "*Presenting Windows Workflow Foundation*" that covers WF at an introductory level for .NET developers. Furthermore, you can download all the sample codes of the book from its web site, which are a good start point for getting familiar with the concepts of WF. Among other things, the following sites may also be of interest to you:

- http://www.windowsworkflow.net
- http://msdn.microsoft.com/workflow

The Concept of Windows Workflow Foundation

The key concept in the WF is the workflows, which are a set of activities. The activities are the building blocks of workflows and enable reuse of the unit of functionalities. The activities can be composed to *Composite Activities*, which can be built into a workflow as a unit. Figure 7 illustrates the architecture and the components of WF. The workflows run in a hosting environment at runtime, which can be any .NET applications. The basis for operating workflows at runtime is the workflow foundation that consists of the following components:

- *Runtime Service* provides the necessary foundational services for the workflows at runtime, such as persistence service.
- *Runtime Engine* is responsible for workflow execution and state management at runtime. It consumes the *Runtime Service* to save serialized workflow instance to the disk or restore saved workflow instance from disk. In addition, the *Runtime Engine* schedules the execution of the activities based on event being caught or sequence defined in the workflow.

- *Basic Activity Library* contains a set of out-of-box basic activities and the base for the custom activities. The basic activities are the steps within a workflow.

Based on the *Basic Activity Library*, developers can build their own *Custom Activity Library* to extend the WF for matching domain specific purpose requirements. For building workflows from activities in the *Base/Custom Activity Libraries*, WF provides also a *Visual Designer* that is integrated in Visual Studio for designing and debugging workflows.

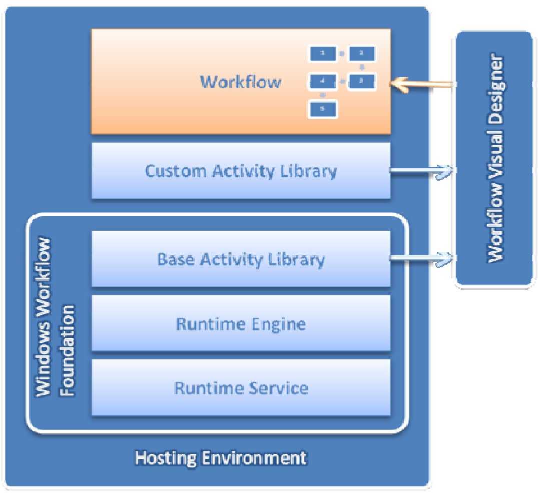
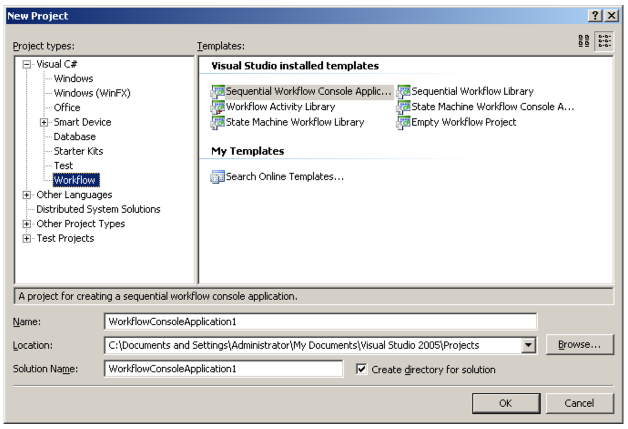


Figure 7: Architecture and Components of Windows Workflow Foundation

After having installed the Visual Studio Extension for Workflow Foundation, the Visual Designer and a set of project templates are added to Visual Studio 2005. To create a WF project in Visual Studio, navigate to "**File**"-> "**New**"-> "**Project...**" to start the "**New Project**" dialog. In the dialog, select "**Workflow**" in the Project types to get all the project templates available for workflow projects, as shown in the following figure:


New Project

Project types:
- Visual C#
  - Windows
  - Windows (WinFX)
  - Office
  - Smart Device
  - Database
  - Starter Kits
  - Test
  - Workflow
- Other Languages
- Distributed System Solutions
- Other Project Types
- Test Projects

Templates:

Visual Studio installed templates

Sequential Workflow Console Applic...	Sequential Workflow Library
Workflow Activity Library	State Machine Workflow Console A...
State Machine Workflow Library	Empty Workflow Project

My Templates

Search Online Templates...

A project for creating a sequential workflow console application.

Name: WorkflowConsoleApplication1

Location: C:\Documents and Settings\Administrator\My Documents\Visual Studio 2005\Projects

Solution Name: WorkflowConsoleApplication1

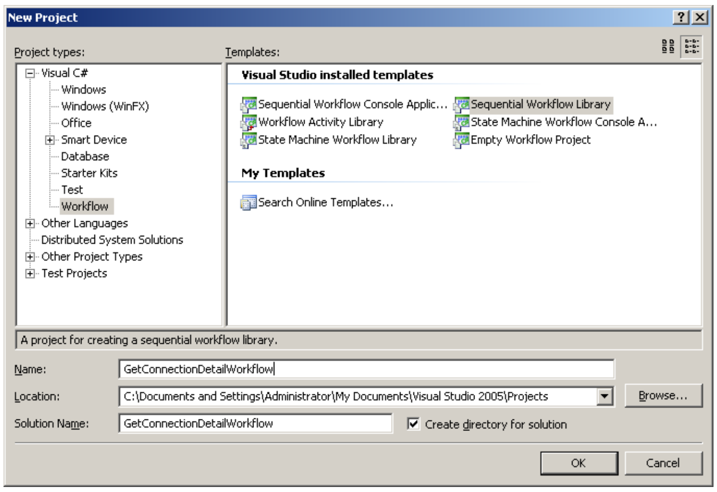
[✓] Create directory for solution

In the dialog, you can create various workflow projects from the templates:

- *Sequential Workflow Console Application*: this template creates a workflow project that contains a default sequential workflow and a test console application for hosting the workflow
- *Sequential Workflow Library*: this template creates a project for building sequential workflow in a library that can be referenced by other applications
- *Workflow Activity Library*: this template creates a project that creates a library of activities for re-use later in workflow projects
- *State Machine Workflow Console Application*: this templates creates a workflow project that contains a default state machine workflow and a test console application for hosting the workflow
- *State Machine Workflow Library*: this template creates a project for building state machine workflow in a library that can be referenced by other applications
- *Empty Workflow Project*: this template creates an empty project that can contain workflow and hosting applications later at design time.

There are two workflows types to create workflow projects through the templates. The *sequential* workflow is a set of activities that can be expressed using a process pipeline and the execution of the workflow acts in accordance with the sequence in the process. The order of the execution can be affected by external events. The *state machine* workflow is totally event-driven. It contains a set of states, transitions between states and actions. Starting with an initial state, a transition can be made to another state based on an event being caught. And the end of the workflow is defined by a final state.

For our flight-booking scenario, we need a workflow that returns a set of flight connections to the desired airport. The result is not only a list of flight connections. It contains also the details of the flight connections. To achieve this functionality, we implement a sequential workflow that consumes BAPI functions from SAP to create the flight connections. Therefore, we need to create a Visual Studio project by using the "Sequential Workflow Library" template and call the new Visual Studio project "GetConnectionDetailWorkflow", as shown in the following figure:


New Project
Project types:
Visual C#
Windows
Windows (WinFX)
Office
Smart Device
Database
Starter Kits
Test
Workflow
Other Languages
Distributed System Solutions
Other Project Types
Test Projects
Templates:
Visual Studio installed templates
Sequential Workflow Console Applic...
Sequential Workflow Library
Workflow Activity Library
State Machine Workflow Console A...
State Machine Workflow Library
Empty Workflow Project
My Templates
Search Online Templates...
Browse...
OK
Cancel

A project for creating a sequential workflow library.

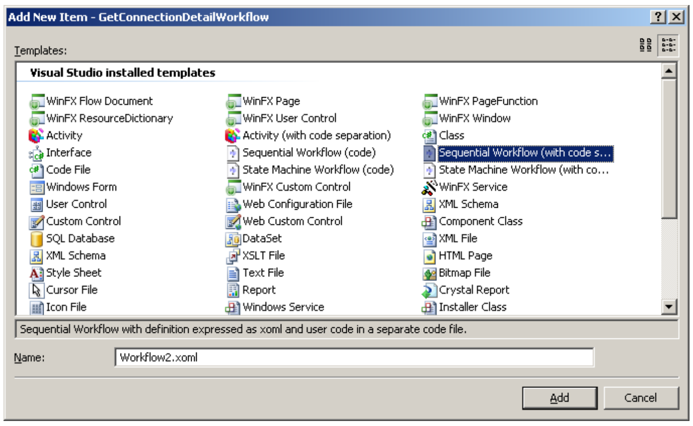
Name: GetConnectionDetailWorkflow

Location: C:\Documents and Settings\Administrator\My Documents\Visual Studio 2005\Projects

Solution Name: GetConnectionDetailWorkflow

☑ Create directory for solution

The project being created contains initially two files: **workflow1.cs** and **workflow1.designer.cs**. The workflow1.cs is the code file that you can use to build in your own business logic. The other file, workflow1.designer.cs, is generated by the WF Visual Designer and contains the description of the workflow. As an alternative, you can also use a XML-based declarative Extensible Application Markup Language (XAML) to describe the workflow. To add a XML-based workflow file, simply add a new item to the workflow project we created before. In the dialog, you can see a file template "**Sequential Workflow (with code separation)**", as shown in the following figure. If you create a new code file based on this template, two new files are added to the project: **workflow2.xoml** and **workflow2.xoml.cs**. The first file describes the workflow model in XML and the second file is generated by Visual Designer. Both options for creating workflows in the project are equivalent to each other, once the source for the workflow, either in programming language or in markup language, is compiled into .NET assembly. In our scenario, we simply use the first option to describe a workflow.



Sequential Workflow with definition expressed as xoml and user code in a separate code file.

Name: Workflow2.xoml

To open the visual workflow designer, simply double-click the file **workflow1.cs**. The initial workflow contains only start and end activities. To add new activities to the workflow, you can simply drag-and-drop the appropriate activity from the **Toolbox** onto the design surface. In our scenario, we have to make two BAPI function calls one after another. The first call gets all the flight connections returned in a list and the second call gets the details of each flight connection in the list, as illustrated in the following flow chart.

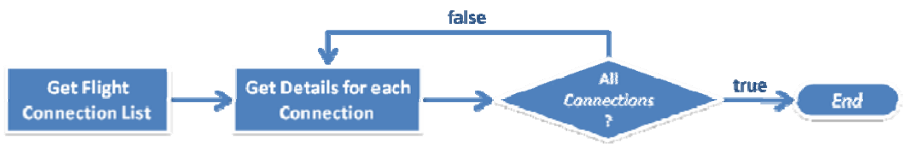
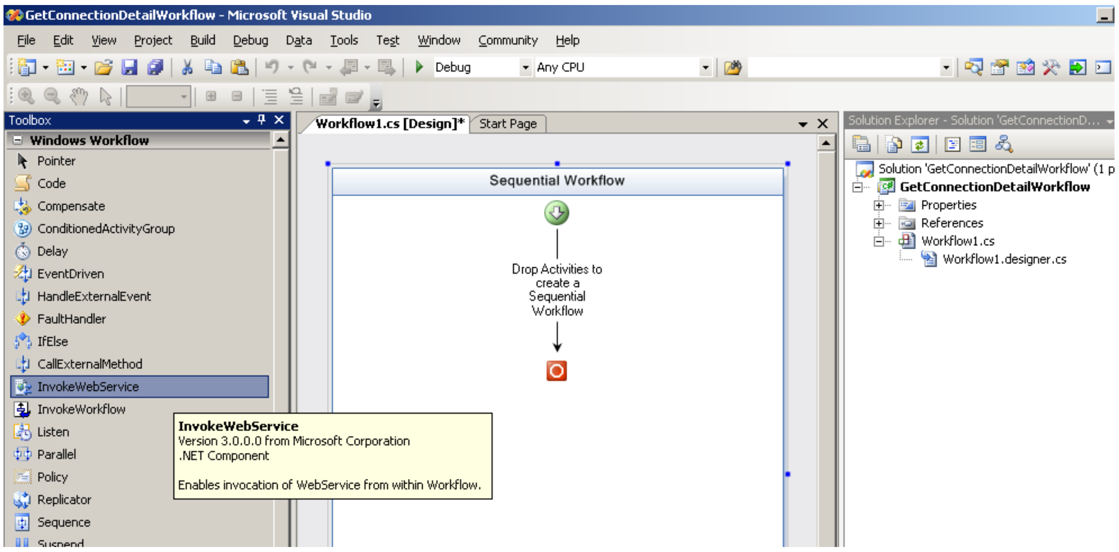


Figure 8: Illustration of the GetFlightConnectionDetails Process

To model this workflow in WF and fill the workflow with activities, please follow the instructions here:

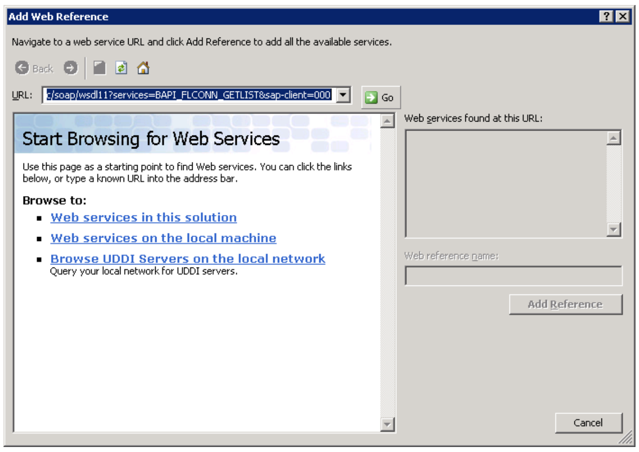
1. In the first step, we need to get a list of flight connections. This can be done by calling the BAPI function **BAPI_FLCONN_GETLIST** from SAP. Therefore, we need an activity to make a Web service call. To do it, simply drag the activity "**InvokeWebService**" from the **Toolbox** onto the design surface of the workflow to add it to the workflow, as shown in the following figure.



InvokeWebService
Version 3.0.0.0 from Microsoft Corporation
.NET Component

Enables invocation of WebService from within Workflow.

2. A new dialog is shown for adding the corresponding Web Reference to the project. In the **URL** text field, enter the URL to the WSDL document of the BAPI_FLCONN_GETLIST Web service. It may looks like http://jdaiberl69.ctsc.org:8000/sap/bc/soap/wsdl11?services=BAPI_FLCONN_GETLIST&sap-client=000 . After that, click "**Go**" to continue.


Add Web Reference
Back
Go
Browse to:
Web services in this solution
Web services on the local machine
Browse UDDI Servers on the local network
Add Reference
Cancel

Navigate to a web service URL and click Add Reference to add all the available services.

URL: c/soap/wsdl11?services=BAPI_FLCONN_GETLIST&sap-client=000

Start Browsing for Web Services

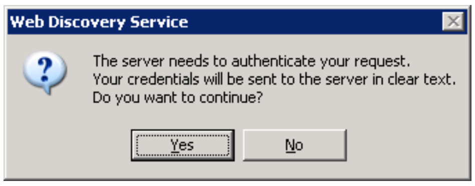
Use this page as a starting point to find Web services. You can click the links below, or type a known URL into the address bar.

Query your local network for UDDI servers.

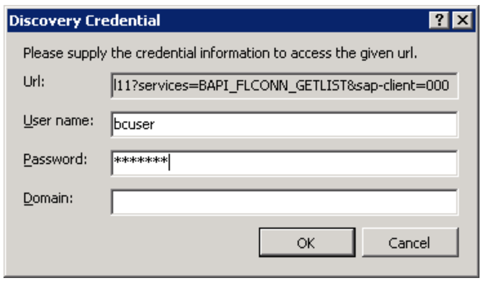
Web services found at this URL:

Web reference name:

3. You may be required to enter the authentication data for the SAP ECC system. For the SAP ECC system that we use, use the username “BCUSER” and the password “minisap” as the login data.



The server needs to authenticate your request. Your credentials will be sent to the server in clear text. Do you want to continue?



Please supply the credential information to access the given url.

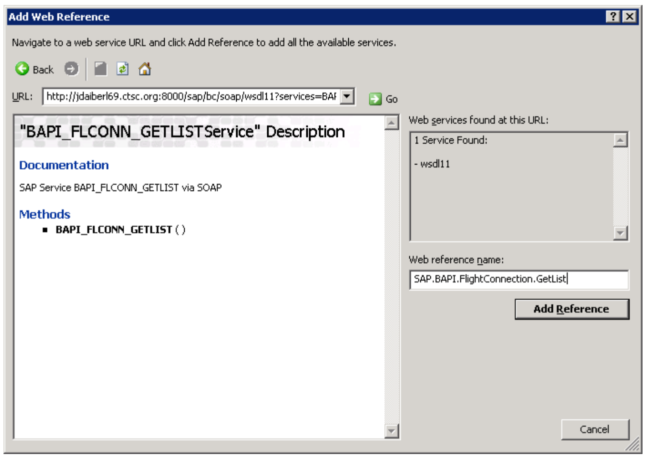
Url: l11?services=BAPI_FLCONN_GETLIST&sap-client=000

User name: bcuser

Password: *******

Domain:

4. After the successful authentication, the Web service description with the available Web methods is shown in the dialog. Give the new Web reference a meaningful name like “**SAP.BAPI.FlightConnection.GetList**” and click “**Add Reference**” to add it to the project. This adds a new Web reference node in your project, which you can find in the **Web References** folder within the **Solution Explorer**. All the programming code you need to call the BAPI Web service is generated for you.



Navigate to a web service URL and click Add Reference to add all the available services.

URL: http://jdaiberl69.ctsc.org:8000/sap/bc/soap/wsdl11?services=BAI

"BAPI_FLCONN_GETLISTService" Description

Documentation

SAP Service BAPI_FLCONN_GETLIST via SOAP

Methods

- BAPI_FLCONN_GETLIST ( )

Web services found at this URL:

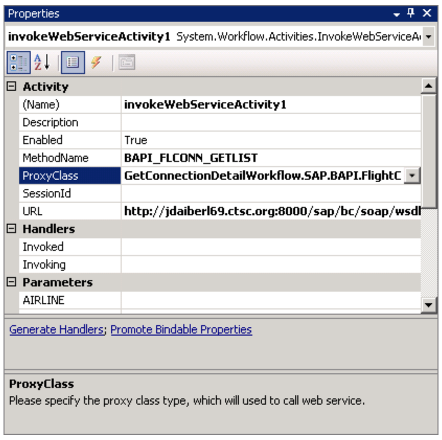
1 Service Found:

- wsdl11

Web reference name: SAP.BAPI.FlightConnection.GetList

5. In this step, we configure the **InvokeWebService** activity that we added before to call the BAPI Web service **BAPI_FLCONN_GETLIST**.

Right-click the activity and select “**Properties**” from the context menu to show the “**Properties Window**”. Then in the “**Properties Window**”, select from the drop-down menu of “**ProxyClass**” the value “**GetConnectionDetailWorkflow.SAP.BAPI.FlightConnection.GetList.BAPI_FLCONN_GETLISTService**”. Afterwards, select from the drop-down menu of “**MethodName**” the value “**BAPI_FLCONN_GETLIST**”. Now you have set up the activity so that it will invoke the BAPI function “**BAPI_FLCONN_GETLIST**” at runtime.


Properties

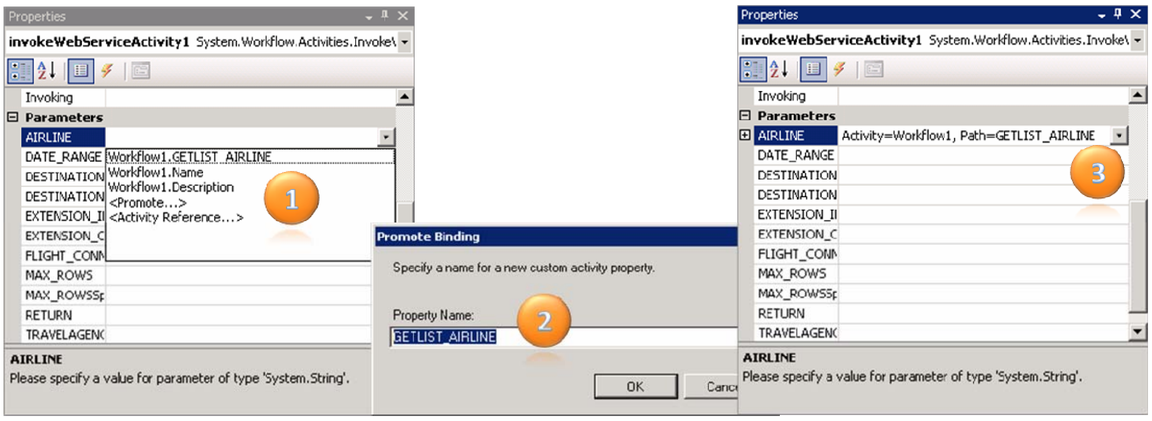
invokeWebServiceActivity1 System.Workflow.Activities.InvokeWebServiceA

Activity	
(Name)	**invokeWebServiceActivity1**
Description	
Enabled	True
MethodName	**BAPI_FLCONN_GETLIST**
ProxyClass	**GetConnectionDetailWorkflow.SAP.BAPI.FlightC**
SessionId	
URL	**http://jdaiberl69.ctsc.org:8000/sap/bc/soap/wsdl**
Handlers	
Invoked	
Invoking	
Parameters	
AIRLINE	

Generate Handlers; Promote Bindable Properties

ProxyClass
Please specify the proxy class type, which will used to call web service.

6. After the value of the property “**MethodName**” is set, the Visual Designer automatically added a set of parameters, which are required for the Web service invocation, to the properties of the activity. All these new properties have to be set, before the workflow can execute this activity at runtime. There are generally two ways to set a property for an activity. The first one is to use the parameter promotion functionality offered by the Visual Designer to automatically declare a new property of the same type in the source code and assign the property to the parameter. The other one is to do this all by you manually. We choose to use the first option with automatic promotion here. To promote a new property for the parameter, simply select “**<Promote...>**” from the drop-down menu of the parameter. In the new dialog, specify the name for the new activity property like “**GETLIST_AIRLINE**” for the parameter “**AIRLINE**” and click “**OK**” to continue. The Visual Designer declares the necessary activity property of the type “System.String” and assigns it to the parameter AIRLINE, as shown in the following figure. Repeat this step for all other parameters for this activity: DATE_RANGE, DESTINATION_FROM, DESTINATION_TO, EXTENSION_IN, EXTENSION_OUT, FLIGHT_CONNECTION_LIST, MAX_ROWS, MAX_ROWSSpecified, RETURN, and TRAVELAGENCY.


Properties

invokeWebServiceActivity1 System.Workflow.Activities.InvokeW

Invoking

Parameters

AIRLINE

DATE_RANGE

DESTINATION

DESTINATION

EXTENSION_II

EXTENSION_C

FLIGHT_CONN

MAX_ROWS

MAX_ROWSSp

RETURN

TRAVELAGENC

Workflow1.GETLIST_AIRLINE
Workflow1.Name
Workflow1.Description
<Promote...>
<Activity Reference...>

1

AIRLINE
Please specify a value for parameter of type 'System.String'.

Promote Binding

Specify a name for a new custom activity property.

Property Name: GETLIST_AIRLINE

2

OK Canc

Properties

invokeWebServiceActivity1 System.Workflow.Activities.InvokeW

AIRLINE Activity=Workflow1, Path=GETLIST_AIRLINE

3

AIRLINE
Please specify a value for parameter of type 'System.String'.

7. After the last step, the activity **invokeWebServiceActivity1** is configured to call the BAPI Web service at runtime. However, this activity needs a set of input parameters and the necessary credential to invoke the BAPI Web service in SAP ECC. The following figure illustrates how an **InvokeWebService** activity makes a Web service call at runtime. Before the activity makes the actual Web service call, it raises an "*Invoking*" event. You can catch this event by assigning an *EventHandler* to it. In the *EventHandler*, you can do some task before the actual Web service call, e.g. initializing the input parameters for the Web services or specifying the authentication credentials for the backend system. After that, the activity goes on making the Web service call. Once the server, in our case the SAP ECC, has returned the result of the Web service call, the activity raises an "*Invoked*" event again. By subscribing to this event using an *EventHandler*, you can handle the result of the invocation.

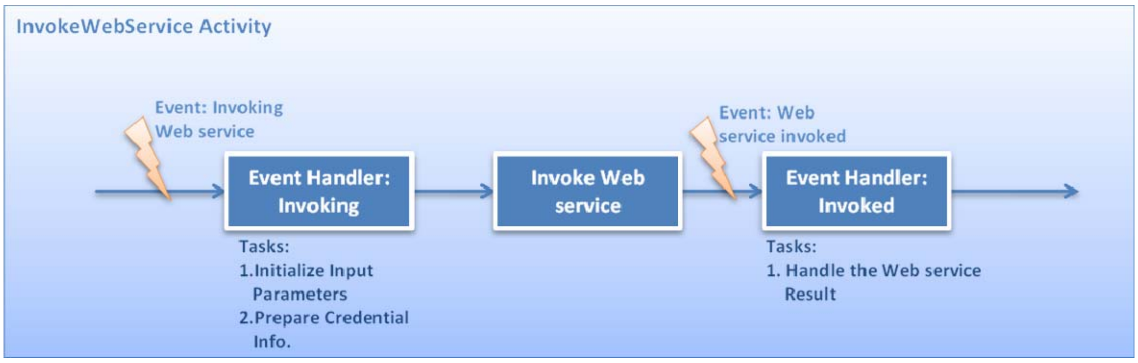
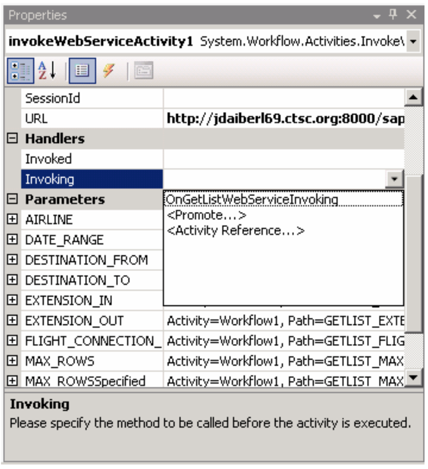

InvokeWebService Activity
Event: Invoking Web service
Event Handler: Invoking
Invoke Web service
Event: Web service invoked
Event Handler: Invoked
Tasks:
1.Initialize Input Parameters
2.Prepare Credential Info.
Tasks:
1. Handle the Web service Result

Figure 9: The Lifecycle of the InvokeWebService Activity at Runtime

Let us go back to our scenario to explain how it works. Right-click on any blank space of the design surface and select "**View Code**" to switch to the *Code*-View of the workflow. At first, we need an *EventHandler* for the ***Invoking*** event. The *EventHandler* should at first set up the network credential for the SAP BAPI Web service call and it has to initialize the input parameters for the Web service call, too. Analogously, we need an *EventHandler* for the ***Invoked*** event that handle the result returned by the Web service. The source code for these both *EventHandlers* is as follows:

```
private void OnGetListWebServiceInvoking(object sender, InvokeWebServiceEventArgs e)
{
        // set the network credential for SAP BAPI call
        System.Web.Services.Protocols.SoapHttpClientProtocol client =

(System.Web.Services.Protocols.SoapHttpClientProtocol)e.WebServiceProxy;
        System.Net.NetworkCredential credential = new System.Net.NetworkCredential();
        credential.UserName = "bcuser";
        credential.Password = "minisap";
        client.Credentials = credential;

        // initialize the parameters
        this.GETLIST_DATE_RANGE = new BAPISCODRA[] { };
        this.GETLIST_DESTINATION_FROM = new BAPISCODST();
        this.GETLIST_DESTINATION_TO = new BAPISCODST();
        this.GETLIST_EXTENSION_IN = new BAPIPAREX[] { };
        this.GETLIST_EXTENSION_OUT = new BAPIPAREX[] { };
        this.GETLIST_FLIGHT_CONNECTION_LIST = new BAPISCODAT[] { };
        this.GETLIST_RETURN = new BAPIRET2[] { };

        return;
}

private void OnGetListWebServiceInvoked(object sender, InvokeWebServiceEventArgs e)
{
        // do something with the Web service call result here.
}
```

Now we have declared two *EventHandlers*, one for the *Invoking* event, and the other for the *Invoked* event. Both these *EventHandlers* have to subscribe to the corresponding events that activity will fire at runtime. To do it, open the properties of the activity in the **Properties Window**. Find the event "***Invoking***" and "***Invoked***" in the section "**Handler**s". Then select from the drop-down list of both events the appropriate EventHandler: **OnGetListWebServiceInvoking** for the ***Invoking*** event and **OnGetListWebServiceInvoked** for the ***Invoked*** event. Now we have finished setting up the activity with all the parameters and event handlers.



invokeWebServiceActivity1 System.Workflow.Activities.InvokeW

SessionId	
URL	**http://jdaiberl69.ctsc.org:8000/sap**
Handlers	
Invoked	
Invoking	
Parameters	OnGetListWebServiceInvoking
AIRLINE	<Promote...>
DATE_RANGE	<Activity Reference...>
DESTINATION_FROM	
DESTINATION_TO	
EXTENSION_IN	
EXTENSION_OUT	Activity=Workflow1, Path=GETLIST_EXTE
FLIGHT_CONNECTION_	Activity=Workflow1, Path=GETLIST_FLIG
MAX_ROWS	Activity=Workflow1, Path=GETLIST_MAX
MAX_ROWSSpecified	Activity=Workflow1, Path=GETLIST_MAX

Invoking
Please specify the method to be called before the activity is executed.

8. To complete the workflow, we need a **While** activity to go through all the flight connections in the list and make in each loop a Web service call using the **InvokeWebService** activity. The procedure is similar to the activity that we created in the last steps. For the further implementation details, please refer to the sample Visual Studio project for the flight-booking scenario to this whitepaper.

Exposing Workflow as Web Services

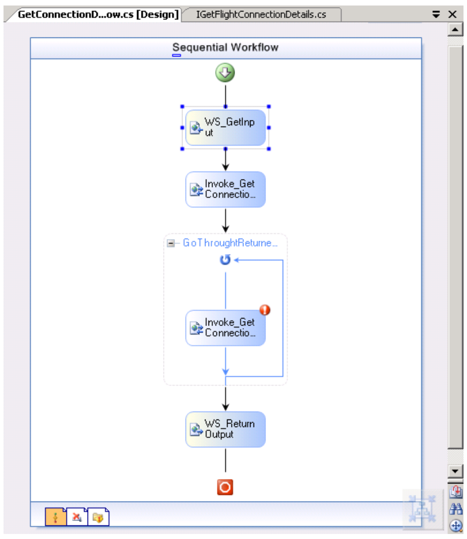
Windows Workflow Foundation provides the possibility to expose a workflow as an ASP.NET Web service to .NET client applications or other workflows. There are two prerequisites for exposing workflows as Web services: the first, there must be a declaration of an Interface that the Web service exposes; the second, the workflow must use the **WebServiceInput** activity. The idea is that the workflow has-at least-a single pair of **WebServiceInput** and **WebServiceOutput** activities that implement the methods declared in the interface. In common case, the workflow is activated by calling a web method exposed by the Web service. Then it proceeds to the end of the workflow and returns the output values via the **WebServiceOutput** activity.

For our flight-booking scenario, we expect a Web service that takes the search criteria for the flight connections, such as airline, departure airport and arrival airport, and return the detailed information of all flight connections matching the search criteria. To expose the workflow that we created in last section as a Web service, please follow the instructions here:

1. Add a new file **IGetFlightConnectionDetails.cs** to the project and double-click it to open it in the source code editor.
2. Declare the interface with the method **GetFlightConnectionDetails**(…), as follows:

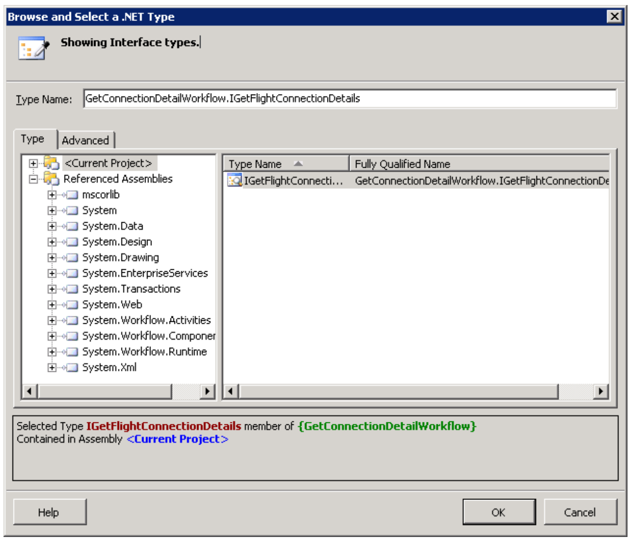
```
using System;
using System.Collections.Generic;
using System.Text;
using CTSC.Whitepaper.WinFX.Workflow.DataModel;

namespace CTSC.Whitepaper.WinFX.Workflow.Interface
{
	interface IGetFlightConnectionDetails
	{
	/// <summary>
	/// this method defines the single web method exported by the workflow later
	/// </summary>
	/// <param name="Agency">the travel agency number</param>
	/// <param name="Airline">the airline, which will be used later to restrict the search
result</param>
	/// <param name="From">the departure airport</param>
	/// <param name="To">the arrival airport</param>
	/// <returns></returns>
	FlightConnectionDetails GetFlightConnectionDetails(
	    string Agency,
	    string Airline,
	    string From,
	    string To);
       }
}
```

3. Add a **WebServiceInput** activity as the initial activity to the workflow created in the last section. And add a **WebServiceOutput** activity as the last activity to the workflow. Afterwards, the workflow should look like this:



4. In the following steps, we will assign the interface to the **WebServiceInput/WebServiceOutput** activities in the workflow. At first, right-click the activity **WS_GetInput** and select “**Properties**” from the context menu to display the activity properties in the **Properties Window**. Then select the property **InterfaceType** and click the button ... to open the dialog for selecting the corresponding interface. In the dialog “**Browse and Select a .NET Type**”, select the

type “**IGetFlightConnectionDetails**” from the list and click “**OK**” to confirm.


Browse and Select a .NET Type

Showing Interface types.

Type Name: GetConnectionDetailWorkflow.IGetFlightConnectionDetails

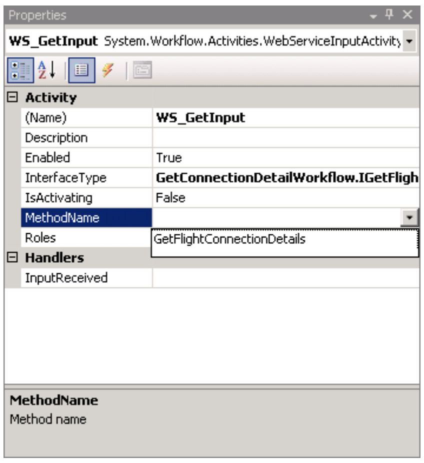
Type | Advanced

```
<Current Project>
Referenced Assemblies
    mscorlib
    System
    System.Data
    System.Design
    System.Drawing
    System.EnterpriseServices
    System.Transactions
    System.Web
    System.Workflow.Activities
    System.Workflow.Componer
    System.Workflow.Runtime
    System.Xml
```

Type Name	Fully Qualified Name
IGetFlightConnecti...	GetConnectionDetailWorkflow.IGetFlightConnectionDe

Selected Type **IGetFlightConnectionDetails** member of **{GetConnectionDetailWorkflow}**
Contained in Assembly **<Current Project>**

5. Switch back to the **Properties Window** of the activity **WS_GetInput**, select the property “**MethodName**” and choose “**GetFlightConnectionDetails**” from the list (as shown in the figure below). Now you have assigned the method to the new Web service. In other words, the new Web service provides a Web method that complies with the signature as defined in the interface. The Visual Designer reads the method signature from the interface and adds automatically all the input parameters as the activity parameters for the activity **WS_GetInput** to the properties list. In the next step, we have to set up the parameters.


Properties

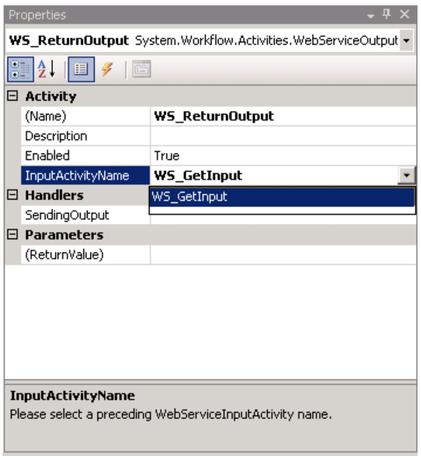
WS_GetInput System.Workflow.Activities.WebServiceInputActivity

Activity	
(Name)	**WS_GetInput**
Description	
Enabled	True
InterfaceType	**GetConnectionDetailWorkflow.IGetFligh**
IsActivating	False
MethodName	
Roles	GetFlightConnectionDetails
Handlers	
InputReceived	

MethodName
Method name

6. To set up the parameters, please consult the instruction from the last section. Please note that you have to change the value of the property “**IsActivating**” from “**false**” to “**true**”. This ensures that this activity **WS_GetInput** will activate the corresponding workflow instance at runtime.

7. Now we have configured the activity **WS_GetInput** to get Web service request from client at runtime. To teach the WF that the activity **WS_GetInput** correlates with the activity **WS_ReturnOutput** for returning output values, we have to get the activity **WS_ReturnOutput** know that it should returns output for the input activity **WS_GetInput**. To do it, change the activity property "**InputActivityName**" of the activity **WS_ReturnOutput** to "**WS_GetInput**", as follows.



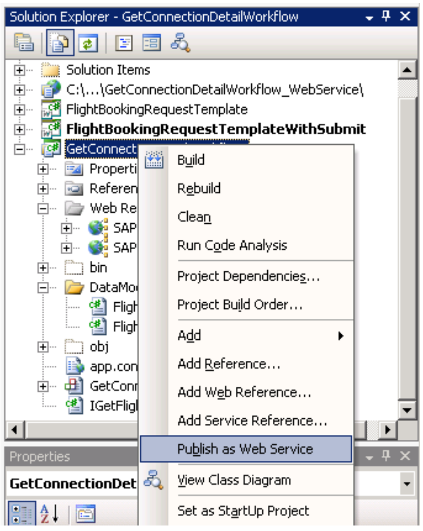
Properties

WS_ReturnOutput System.Workflow.Activities.WebServiceOutput

Activity	
(Name)	WS_ReturnOutput
Description	
Enabled	True
InputActivityName	WS_GetInput
Handlers	WS_GetInput
SendingOutput	
Parameters	
(ReturnValue)	

InputActivityName
Please select a preceding WebServiceInputActivity name.

8. Before you can expose the workflow as a Web service, please make sure that you have set all the activity properties in the workflow. And in case that you need to add your custom code to control the **WebServiceInput/WebServiceOutput** activities, you can subscribe to the events raised by these activities at runtime. To generate handlers that manage input and output data, you simply right-click the input/output activities and click **Generate Handlers** from the context menu. Add all your custom code to the handlers being generated. After you are done with the code, you can expose the workflow as a Web service by right-clicking the project and selecting "**Publish as Web Service**" from the menu (as shown in the figure below). The Visual Designer creates a new ASP.NET Web service project with a corresponding *Web.config* file, an *.asmx* file and an *assembly* that contains the workflow code. You can then deploy this Web service e.g. to IIS and execute the workflow by invoking the workflow Web service.



Using Custom Workflow Activity

Another highlight of WF is the extensibility through custom workflow activities. WF delivers a set of out-of-box activities to help you get started with WF. However, it does not cover all the functional requirements that you may have in real business scenarios, where it is often required to model advanced control flows or to integrate with proprietary technologies. In this case, you can author domain-specific reusable custom activities to build workflow models. For instance, for our flight-booking scenario, we have to send emails to notify the person who is involved in the process. Therefore, we can define this activity as a custom workflow activity **SendingEmail** and encapsulating all the properties (like mail server (POP/SMTP server), sender address, receiver address, etc) and the events (like “sending email” or “email has been sent”) in the activity.

Before we go on with the implementation of the activity **SendingEmail**, let us examine at first the lifecycle of an activity at runtime, as depicted in Figure 10 .

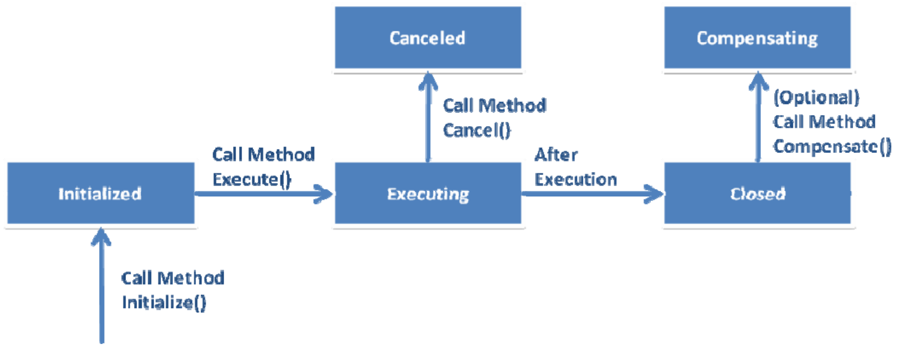


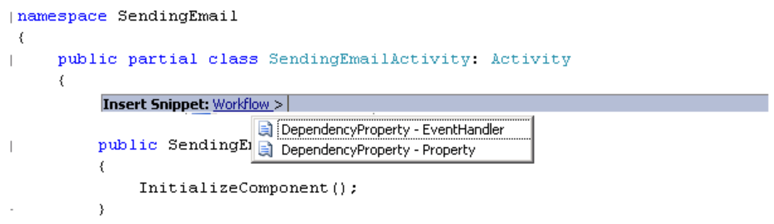
Figure 10: The Lifecycle of a Workflow Activity at Runtime

Each activity implements the methods *Initialize*(), *Execute*(), *Cancel*() and optionally *Compensate*(). The lifecycle of the activity begins with the invocation of the method *Initialize*() by the workflow runtime. After the initialization, the workflow runtime call the *Execute*() method to start the actual procedure defined by the activity. The runtime can call the *Cancel*() method to break the execution. Otherwise, after the activity has finished executing the predefined procedure, it returns a status code that indicates the execution status back to the runtime. Depending on the status code being returned, the runtime can either terminate the activity or call the optional *Compensate*() method for compensation.

To create a new custom activity, you can use the WF project template "**Workflow Activity Library**". The template creates a C# file that contains the code for the new activity. An activity can inherit from any built-in activity defined in the WF or if you want to create a complete new activity from scratch, you can inherit the new class from the base class **System.Workflow.ComponentModel.Activity**, which is the base class for all activities for WF, including the out-of-box ones. The following code sample shows the new class:

```
namespace SendingEmail
{
      public partial class SendingEmailActivity: Activity
     {
           public SendingEmailActivity()
           {
                InitializeComponent();
           }

           protected override ActivityExecutionStatus Execute(ActivityExecutionContext context)
           {
                   // writing code for sending email here
                   return ActivityExecutionStatus.Closed;
           }
      }
}
```

The next task for creating new custom activity is to add a list of properties and event handlers to the activity. It s recommended to use the dependency properties rather than the standard .NET properties. Dependency properties allow binding their value to underlying data source or to other properties of other activities in the same workflow. You can use the built-in code snippets to generate the dependency properties and event handlers, as follows:

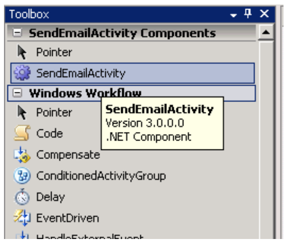


```
namespace SendingEmail
{
    public partial class SendingEmailActivity: Activity
    {
        Insert Snippet: Workflow > |
                          DependencyProperty - EventHandler
        public SendingE   DependencyProperty - Property
        {
            InitializeComponent();
        }
```

The code snippet creates a static instance of the type **DependencyProperty** and a wrapper property for this instance. The following code sample shows the definition of such a **DependencyProperty**. The wrapper property contains a set of attributes that are used by the Visual Designer at design time. In the similar way, you can generate all the necessary properties and event handlers for your custom activity.

```
public static DependencyProperty FromProperty =
          System.Workflow.ComponentModel.DependencyProperty.Register("From",
typeof(string),
                 typeof(SendingEmailActivity));

[Description("This is the description which appears in the Property Browser")]
[Category("This is the category which will be displayed in the Property Browser")]
[Browsable(true)]
[DesignerSerializationVisibility(DesignerSerializationVisibility.Visible)]
public string From
{
      get
      {
           return ((string)(base.GetValue(SendingEmailActivity.FromProperty)));
      }
      set
      {
           base.SetValue(SendingEmailActivity.FromProperty, value);
      }
}
```

To finish creating the custom activity, you have to implement the *Execute*() method in the source code and optionally the other methods *Initialize*(), *Cancel*() or *Compensate*(), according to the business logic that you want to realize through the activity. After you have implemented all the necessary methods and compiled it successfully, the Visual Designer will automatically list this new activity in the toolbox, so that you can add the custom activity to the workflow at design time.



You can find the full source code for the **SendingEmail** activity either in the Visual Studio solutions for the flight-booking scenario to this whitepaper or in the sample codes contained in the WF documentation.

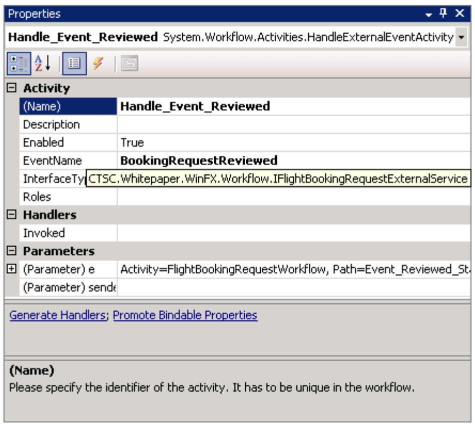
Interacting with external Process

At runtime, a real-world workflow has to interact with external world to complete its tasks. WF supports this approach through a dedicated service with specific attribute. The service enables a bi-directional communication between the workflow and the external world. On one side, the service raises events at runtime that event-driven activities in the workflow can subscribe to. In this way, WF provides an information flow from external world into the workflow. On the other side, the service provides a set of public methods that the workflow can call at runtime. This mechanism provides an information flow from the workflow to the external process.

The events and the public methods are defined in a .NET interface, which is explicitly marked as data exchange service for WF-based workflow by using the class attribute "**ExternalDataExchange**" (as shown in the code sample below). Each data exchange service contains an interface declaration and a .NET class that implements the interface. In the following sample interface, we have defined a data exchange service **FlightBookingRequestExternalService** that contains three public methods and two events.

```
/// <summary>
/// External Data Service Interface for the booking request workflow, which is used by the
workflow to
/// communicate with external source
/// </summary>
[ExternalDataExchange]
public interface IFlightBookingRequestExternalService
{
    void ApproveExpenseReport(BookingRequest request);

    void RejectExpenseReport(BookingRequest request);

    void SubmitBookingRequest(BookingRequest request);

    event EventHandler<RequestSubmittedEventArgs> BookingRequestSubmitted;

    event EventHandler<RequestReviewedEventArgs> BookingRequestReviewed;
}
```

The **[ExternalDataExchange]** class-attribute marks this service interface definition as an external data exchange service so that the workflow runtime knows to use this service to interact with external process. WF delivers two out-of-box activities to consume the external data exchange service. The **HandleExternalEvent** activity can subscribe to the events that the data exchange service may rise at runtime. And the public methods exposed by the data exchange service can be called from within the workflow by using the **CallExternalMethod** activity, as follows:



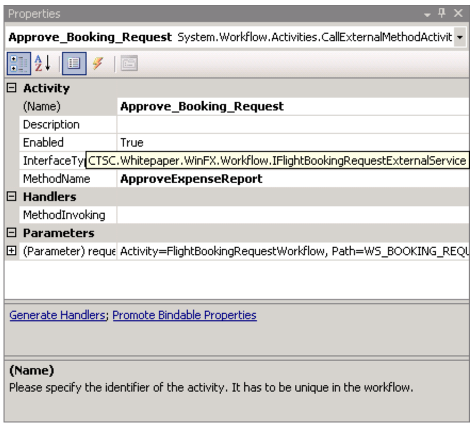
Properties

Handle_Event_Reviewed System.Workflow.Activities.HandleExternalEventActivity

Activity	
(Name)	**Handle_Event_Reviewed**
Description	
Enabled	True
EventName	**BookingRequestReviewed**
InterfaceTy	CTSC.Whitepaper.WinFX.Workflow.IFlightBookingRequestExternalService
Roles	
Handlers	
Invoked	
Parameters	
(Parameter) e	Activity=FlightBookingRequestWorkflow, Path=Event_Reviewed_St
(Parameter) send	

Generate Handlers; Promote Bindable Properties

(Name)
Please specify the identifier of the activity. It has to be unique in the workflow.



Properties

Approve_Booking_Request System.Workflow.Activities.CallExternalMethodActivit

Activity	
(Name)	**Approve_Booking_Request**
Description	
Enabled	True
InterfaceTy	CTSC.Whitepaper.WinFX.Workflow.IFlightBookingRequestExternalService
MethodName	**ApproveExpenseReport**
Handlers	
MethodInvoking	
Parameters	
(Parameter) reque	Activity=FlightBookingRequestWorkflow, Path=WS_BOOKING_REQU

Generate Handlers; Promote Bindable Properties

(Name)
Please specify the identifier of the activity. It has to be unique in the workflow.

The event passes external data into a workflow instance through the **EventArgs** parameter in the event. The **EventArgs** parameter derives from the **ExternalDataEventArgs** class defined the WF. As you can see in the following code sample, via the **EventArgs** parameter you can pass nearly every (serializable) data to the workflow instance that subscribes to the event.

```
[Serializable]
public class RequestReviewedEventArgs : ExternalDataEventArgs
{
        public RequestReviewedEventArgs(System.Guid instanceID, BookingRequest request,
ReviewStatus status)
            : base(instanceID)
        {
            this.BookingRequest = request;
            this.ReviewStatus = status;
        }

        private BookingRequest _request;

        public BookingRequest BookingRequest
        {
            get { return _request; }
            set { _request = value; }
        }

        private ReviewStatus _status;

        public ReviewStatus ReviewStatus
        {
            get { return _status; }
            set { _status = value; }
        }
}
```

The next step to create the data exchange service is to implement a .NET class that implement the interface. This task is quite straightforward, so that we skip over the details of its implementation. As usual, you can find the full implementation for this class in the sample Visual Studio solution for the flight-booking scenario.

Hosting Workflow in your Application

The goal of WF is to make the development of workflow-enabled applications easier than ever. In the last sections, we have outlined how to develop a WF-based workflow and how it works at runtime. A workflow requires the WF workflow runtime as the running environment, and the workflow runtime requires an external application/service to host it. At runtime, the hosting application interacts with WF through the **WorkflowRuntime** or a custom class that inherits from it. Here is a list of host responsibilities that the host application must provide for hosting the workflow:

- Initialize the workflow runtime
- Create custom and local workflow services, such as persistence service, tracking service, etc.
- Start new workflow instance at runtime
- Monitor the specific events on the workflow instance
- Configure the workflow instance for debugging, diagnostic, etc.

The initialization of the workflow runtime is straightforward, as follows:

```
// Get a new workflow runtime
WorkflowRuntime wr = new WorkflowRuntime();
// do something to configure the runtime here
// start the runtime
wr.StartRuntime();
```

Before you start the workflow runtime, you have to add all the necessary services to the runtime that your workflow instance may use at runtime, for instance, the external data exchange service or the persistency service for the workflow runtime. There are two possible ways to configure the services for the workflow runtime. The first one is to add the corresponding service(s) directly in the source code, as shown below:

```
// Add the external data service
ExternalDataExchangeService dataExchangeService = new ExternalDataExchangeService();
wr.AddService(dataExchangeService);
FlightBookingRequestExternalService  externalDataService = new
FlightBookingRequestExternalService();
dataExchangeService.AddService(this.externalDataService);
// Add system SQL state service
SqlWorkflowPersistenceService stateService = new SqlWorkflowPersistenceService(
          "Initial Catalog=WorkflowPersistence;Data Source=.\SQLEXPRESS;Integrated
Security=SSPI; ");
wr.AddService(stateService);
```

The other way to add runtime services to the workflow runtime is to use the configuration file, which gives you more flexibility because you can change the settings anytime directly in the configuration xml file, as follows:

```
<?xml version="1.0" encoding="utf-8" ?>
<configuration>
  <configSections>
```

```
    <section name="BookingRequestWorkflowRuntimeConfig"
type="System.Workflow.Runtime.Configuration.WorkflowRuntimeSection,
System.Workflow.Runtime, Version=3.0.0.0, Culture=neutral,
PublicKeyToken=31bf3856ad364e35"/>
  </configSections>

  <BookingRequestWorkflowRuntimeConfig>
    <Services>
      <add type="System.Workflow.Runtime.Hosting.SqlWorkflowPersistenceService,
System.Workflow.Runtime, Version=3.0.0.0, Culture=neutral,
PublicKeyToken=31bf3856ad364e35" ConnectionString="Initial
Catalog=WorkflowPersistence;Data Source=.\SQLEXPRESS;Integrated Security=SSPI;"/>
      <add type="System.Workflow.Runtime.Tracking.SqlTrackingService,
System.Workflow.Runtime, Version=3.0.0.0, Culture=neutral,
PublicKeyToken=31bf3856ad364e35" ConnectionString="Initial Catalog=WorkflowTracking;Data
Source=.\SQLEXPRESS;Integrated Security=SSPI;" IsTransactional="false" UseDefaultProfile="true"
TrackXomlDocument="True"/>
    </Services>
  </BookingRequestWorkflowRuntimeConfig>
</configuration>
```

To create a specific workflow instance, you have to pass necessary information about the new workflow instance to the workflow runtime. Among other things, the workflow runtime needs to know the .NET CLR type of the workflow, for which a new instance to should be created. The following code demonstrates how to create the workflow instance for our flight-booking scenario. As usual, you can find the full source code in the Visual Studio solutions for the scenario.

```
//create workflow instance id
System.Guid workflowInstanceId = System.Guid.NewGuid();

// Load the CTSC.Whitepaper.WinFX.Workflow.FlightBookingRequestWorkflow workflow through
reflection
Type wType = typeof(CTSC.Whitepaper.WinFX.Workflow.FlightBookingRequestWorkflow);

// prepare the parameter for the workflow instance
Dictionary<string, object> parameters = new Dictionary<string, object>();

parameters.Add("WS_BOOKING_REQUEST", request);
parameters.Add("APPROVAL_TIMEOUT", this.Timeout);

// start the workflow instance
WorkflowInstance wInstance = this.runtime.CreateWorkflow(wType, parameters,
workflowInstanceId);
wInstance.Start();
```

Windows Communication Foundation

The idea behind Windows Communication Foundation is to provide a unified communication programming model that satisfies diverse communication requirements for distributed applications, such as security, interoperability, or transaction. On top of the .NET framework Common Language Runtime (CLR), WCF provides a set of APIs that allows developers building distributed applications in a familiar way as they do for object-oriented applications. The APIs in WCF exhibit the following three most important aspects:

- *Unification*: WCF unified the programming models of .NET framework for building distributed applications. And they are ASP.NET Web services (ASMX), .NET Remoting, Enterprise Services, Web Services Enhancements (WSE) that implements the WS-* specifications in addition to the standard SOAP protocol stack and the Microsoft Message Queuing (MSMQ).
- *Service Orientation*: WCF codifies best practice for building distributed application by encapsulating functionalities as WCF services, which can be accessed via different accessing channel
- *Integration*: WCF interoperates with applications running on other platforms and ensures reliability, security and transactions during the communication process.

In this section, we cover the following topics about WCF:

- The key concepts of WCF: the architecture and the components in WCF
- Providing services using WCF: we explain how to applying the concept of WCF to providing services
- Accessing WCF services: we explain how a WCF client application is constructed and how to develop it

WCF is a powerful technology platform with a lot of customizing options to control the behaviors of the distributed applications being built with it. In this regard, we can only cover the basic part of WCF in this whitepaper. For those, who want to know more about WCF, here is a list of good information sources to start with:

- http://www.windowscommunication.net: the information portal for WCF hosted by Microsoft. Particularly, you can find there a lot of technical articles as well as sample applications for WCF.
- http://msdn.microsoft.com/webservices/indigo/: The WCF developer center at MSDN
- http://msdn.microsoft.com/webservices/: the Web services developer center at MSDN
- The book "*Programming Indigo*" by David Pallmann published by Microsoft Press

The Concept of Windows Communication Foundation

The key concept of WCF is **Endpoint**. Each WCF-based service exposes a set of endpoints, through which the service communicates with the external world. Figure 11 illustrates the structure of WCF-based service. Each endpoint is composed of three parts: **Address**, **Binding**, **Contract**, or simply "**ABC**" of WCF. *Address* explains the question "Where is the service?" and provides the network address where the service resides. *Binding* addresses the question "How to communicate with the service?" and specifies things like transport protocol of the messages (e.g. HTTP or TCP), security requirements (e.g. WS-Security or SSL), etc. *Contract* explains the question “What does the service do?” and specifies things like the interface of the service, the data exposed by the service, etc.

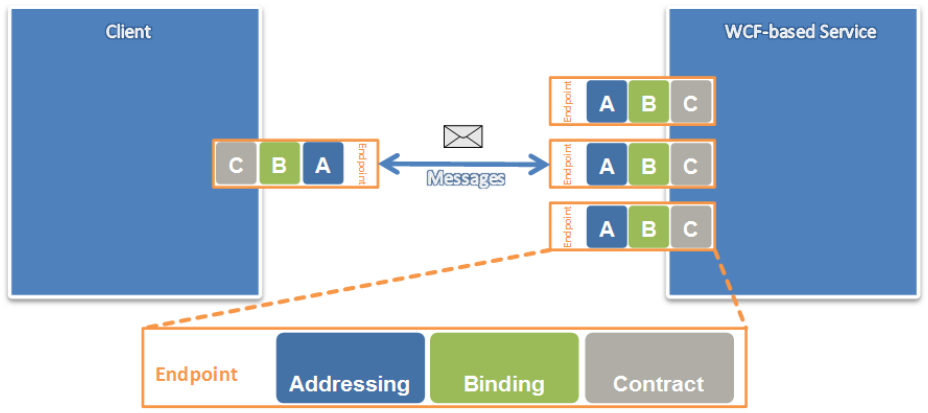


Figure 11: The Key Concepts of Windows Communication Foundation

The term “**ABC**” of WCF indicates that there are three tasks to complete in order to create a WCF-based service:

- *Contract*-related task: you specify the contract of the service and implement it to the service.
- *Binding*-related task: you specify the service binding along with the transport protocol and other service-level parameters, such as security, reliable messaging, etc.
- *Address*-related task: you specify the network address of the service and deploy the service to the specified network address.

In the following sections, we demonstrate this approach by implementing a WCF-based service for our flight-booking scenario. In our scenario, we need a service where all the booking requests run together. Using the service, we can submit a new request, change the status of a request or get a list of all submitted requests.

Providing Services with Windows Communication Foundation

In this section, we follow the aforementioned “**ABC**” task list to create the booking-request service.

At first, we need to define the contracts for the new service, as described in the task list. WCF uses a set of contracts to control the behavior of the WCF-based services. The most important contracts are **ServiceContract**, **OperationContract**, **DataContract** and **MessageContract**. Those people, who have ever implemented an ASP.NET Web service, may still remember the *code-first* approach to implement a .NET Web service. There, developers can simply implement a normal .NET class in their preferred programming language. Afterwards, they can use the class attribute **[WebService]** to mark a class definition as a Web service and use the method attribute

[WebMethod] to mark a public method definition as a Web method. These attributes give the .NET Framework the necessary information about the properties and the behaviors of such classes/methods. WCF follows the similar way to let developers to define contracts for the WCF-based services. In the following, we explain what these particular contracts affect for our booking-request service.

The following code sample shows the **ServiceContract** definition for the booking-request service.

```
//This interface definition serves as the service contract for the FlightBookingRequestService
[ServiceContract(Namespace = "urn:microsoft-sap:ctsc:whitepaper:winfx:wcf")]
[XmlSerializerFormat]
public interface IFlightBookingRequestService
{
        // submit a new flight booking request to the service
        [OperationContract]
        System.Guid SubmitFlightBookingRequest(BookingRequest request);

        // approve or reject an existing flight booking request
        [OperationContract]
        System.Guid ReviewFlightBookingRequest(BookingRequest request, bool approved);

        // return all the booking requests as a list
        [OperationContract]
        List<BookingRequest> GetBookingRequestList();
}
```

In the sample code, the interface declaration is explicitly marked as a Service Contract by the **[ServiceContract(...)]** attribute. The attribute tells the WCF runtime environment that this interface definition carries contract metadata for the WCF service. Each method in the interface is also explicitly labeled with the **[OperationContract]** attribute, which is equivalent to the **[WebMethod]** attribute of an ASP.NET Web service.
The attribute **[XmlSerializerFormat]** is the key between the new WCF service and the existing ASMX services. This attribute tells the WCF runtime explicitly that the runtime should use the **XmlSerializer** from the namespace **System.Xml.Serialization** for all types in the current ServiceContract at runtime. This attribute makes it possible to move existing ASMX services to the new WCF platform.

Now the methods exposed by the booking-request service have been determined through the **ServiceContract** and the **OperationContract** attributes. The booking-request service exchanges data with its client at runtime through input/output XML messages. Therefore, in the next step, we have to specify the data exchanged between the service and its client at runtime, so that the WCF runtime can serialize the data properly into XML message as well as de-serialize XML messages into data. In WCF, this can be done by using the **[DataContract]** attribute, as follows:

```
[DataContract]
```

```
public class BookingRequest
{
	[DataMember]
	public System.Guid RequestID;

	// other data member definition
	.....
}
```

The next attribute to control the messages exchanged between WCF-based services and their clients are the **MessageContract** attribute. Using this attribute, WCF allows you to control the structure of the SOAP messages. You can explicitly specify which fields in your class should be mapped to the SOAP headers and which field to the SOAP body. For more information for using this attribute, please consult the WCF document.

After having defined all the contracts necessary, you have to implement the interface by building all your custom business logics into the public methods of the service. Because this task is straightforward and varies from case to case, we skip over the implementation details for our booking-request service. You can find the full source code in the sample Visual Studio solutions for the scenario.

Now we have finished defining the contracts for our booking-request service. According to the task list that we defined before, the next step is to configure the binding settings for the new service. The binding settings contain three aspects that control how messages are transported between the service and the client: the transport protocol, including HTTP, TCP and MSMQ; the message encoding like XML 1.0, binary, or Message Transmission Optimization Mechanism (MTOM) for attachments; and the service-level agreements, including WS-Security, WS-Reliability, etc. Based on these three aspects, you can create any possible binding settings for your WCF-based service. This task may have some overhead due to the complexity of the transport aspects; therefore, WCF provides a set of predefined bindings for the most common use cases, as shown in the following table (excerpt from the WCF document):

Binding	**Description**
BasicHttpBinding	This binding uses HTTP as the transport and Text/XML as the default message encoding.
WSHttpBinding	A secure and interoperable binding that is suitable for non-duplex service contracts.
WSDualHttpBinding	A secure and interoperable binding that is suitable for duplex service contracts or communication through SOAP intermediaries.
WSFederationBinding	A secure and interoperable binding that supports the WS-Federation protocol, enabling organizations that are in a federation to efficiently authenticate and authorize users.

NetTcpBinding	A secure and optimized binding suitable for cross-machine communication between WCF applications.

For our booking-request service, we only need the basic Web service binding, because our service is only available in intranet via HTTP. For configuring the binding setting for the service, you can either do it in the source code or in the configuration file, as shown below. Both options are equivalent respecting the impact. However, the approach with the configuration file is much more flexible for you, if the use case changes later.

```
WSHttpBinding binding = new WSHttpBinding();
// 1 minute send timeout
binding.SendTimeout = new TimeSpan(0, 1, 0);
// basic authentication
binding.Security.Transport.ClientCredentialType =
System.ServiceModel.Channels.HttpClientCredentialType.Basic;
```

```
<system.serviceModel>
   <services>
     <service name="CTSC.Whitepaper.WinFX.WCF.Service.FlightBookingRequestService">
       <endpoint address="http://localhost:8080/CTSC/BookingRequestService"
         binding="wsHttpBinding"
contract="CTSC.Whitepaper.WinFX.WCF.Service.IFlightBookingRequestService" />
     </service>
   </services>
   <bindings>
     <wsHttpBinding>
       <binding sendTimeout="60">
         <security>
           <transport clientCredentialType="Basic"/>
         </security>
       </binding>
     </wsHttpBinding>
   </bindings>
</system.serviceModel>
```

The last step in the task list is **Addressing**, which specifies the network address of the new WCF service. It can be done in combination with the **ServiceHost**. Each WCF service needs a .NET-based application for hosting. You can instantiate a new **ServiceHost** by passing the service type and the URI of the new service (Address in our task list) to the **ServiceHost** instance, just as in the following sample code:

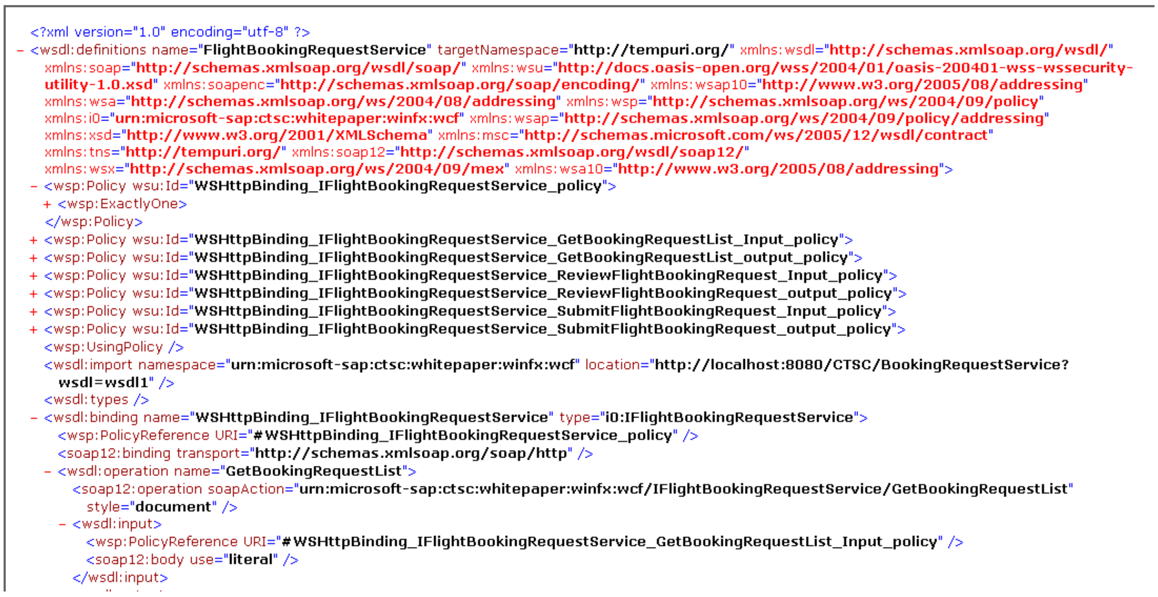
```
Uri uri = new Uri("http://localhost:8080/CTSC/BookingRequestService");

if (this.flightBookingServiceHost == null)
{
      this.flightBookingServiceHost = new
          ServiceHost(typeof(CTSC.Whitepaper.WinFX.WCF.Service.FlightBookingRequestService),
uri);
}

// start the service
this.flightBookingServiceHost.Open();
```

At runtime, the service is accessible under the address "http://localhost:8080/CTSC/BookingRequestService" and this service allows consumers to communicate with it through the HTTP protocol.

Now we have gone through all the implementation details on the server side. In the next section, we will look at the client side and explain how to develop client application for WCF services.

Accessing WCF Services from your Application

The key for accessing WCF services is the metadata - the meta-information about the WCF services. In fact, WCF services use WSDL to describe their metadata. For our booking-request service, you can get the WSDL document for the service via the URL "http://localhost:8080/CTSC/BookingRequestService?WSDL", as follows:

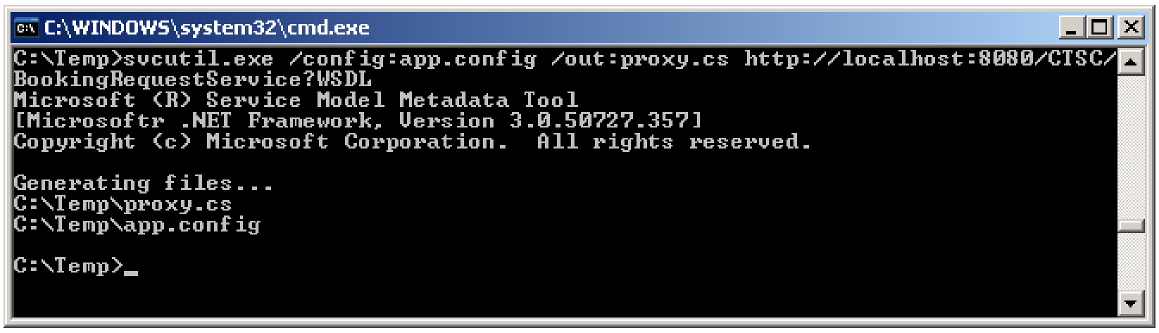


```
  <?xml version="1.0" encoding="utf-8" ?>
- <wsdl:definitions name="FlightBookingRequestService" targetNamespace="http://tempuri.org/" xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
    xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/" xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-
    utility-1.0.xsd" xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/" xmlns:wsap10="http://www.w3.org/2005/08/addressing"
    xmlns:wsa="http://schemas.xmlsoap.org/ws/2004/08/addressing" xmlns:wsp="http://schemas.xmlsoap.org/ws/2004/09/policy"
    xmlns:i0="urn:microsoft-sap:ctsc:whitepaper:winfx:wcf" xmlns:wsap="http://schemas.xmlsoap.org/ws/2004/09/policy/addressing"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:msc="http://schemas.microsoft.com/ws/2005/12/wsdl/contract"
    xmlns:tns="http://tempuri.org/" xmlns:soap12="http://schemas.xmlsoap.org/wsdl/soap12/"
    xmlns:wsx="http://schemas.xmlsoap.org/ws/2004/09/mex" xmlns:wsa10="http://www.w3.org/2005/08/addressing">
  - <wsp:Policy wsu:Id="WSHttpBinding_IFlightBookingRequestService_policy">
    + <wsp:ExactlyOne>
    </wsp:Policy>
  + <wsp:Policy wsu:Id="WSHttpBinding_IFlightBookingRequestService_GetBookingRequestList_Input_policy">
  + <wsp:Policy wsu:Id="WSHttpBinding_IFlightBookingRequestService_GetBookingRequestList_output_policy">
  + <wsp:Policy wsu:Id="WSHttpBinding_IFlightBookingRequestService_ReviewFlightBookingRequest_Input_policy">
  + <wsp:Policy wsu:Id="WSHttpBinding_IFlightBookingRequestService_ReviewFlightBookingRequest_output_policy">
  + <wsp:Policy wsu:Id="WSHttpBinding_IFlightBookingRequestService_SubmitFlightBookingRequest_Input_policy">
  + <wsp:Policy wsu:Id="WSHttpBinding_IFlightBookingRequestService_SubmitFlightBookingRequest_output_policy">
    <wsp:UsingPolicy />
    <wsdl:import namespace="urn:microsoft-sap:ctsc:whitepaper:winfx:wcf" location="http://localhost:8080/CTSC/BookingRequestService?
      wsdl=wsdl1" />
    <wsdl:types />
  - <wsdl:binding name="WSHttpBinding_IFlightBookingRequestService" type="i0:IFlightBookingRequestService">
      <wsp:PolicyReference URI="#WSHttpBinding_IFlightBookingRequestService_policy" />
      <soap12:binding transport="http://schemas.xmlsoap.org/soap/http" />
    - <wsdl:operation name="GetBookingRequestList">
        <soap12:operation soapAction="urn:microsoft-sap:ctsc:whitepaper:winfx:wcf/IFlightBookingRequestService/GetBookingRequestList"
          style="document" />
      - <wsdl:input>
          <wsp:PolicyReference URI="#WSHttpBinding_IFlightBookingRequestService_GetBookingRequestList_Input_policy" />
          <soap12:body use="literal" />
        </wsdl:input>
```

The WSDL document describes the interfaces (corresponding to **ServiceContract**), the messages (corresponding to **Data/MessageContract**), the binding information (corresponding to **Binding**) and the network address (corresponding to **Addressing**), which correspond to the building elements of a WCF-based service. This document gives client applications the necessary knowledge to invoke the desired WCF service, like where to access the WCF service, which methods the WCF service provides and how to invoke the methods (by knowing how to serialize and de-serialize messages it passes to and receives from the WCF service.). To consume the desired WCF service, WCF adopts the same approach as consuming a .NET Web service, namely accessing the WCF service through a proxy that encapsulates all the complexity associated with the service invocation in the underlying layers, like (de-)serializing, invoking, data transport, etc. To build the WCF service proxy for your client application, you can use either the *Service Model Metadata Tool* **SvcUtil.exe** or the build-in

Proxy Generator for Visual Studio 2005. Both tools are capable of importing metadata information from the desired WCF service to generate the appropriate proxies. Furthermore, they modify the application configuration file to embed the appropriate binding information for the client application in compliance with the WCF service binding settings.

To run SvcUtil.exe, you have to specify the URL pointing to the WCF service's WSDL document. With the ***/config*** switch, the tool generates the compatible configuration file with the necessary endpoint information, which you have to merge manually with your client application configuration file. The following figure illustrates the output of SvcUtil.exe for the booking-request service. This tool creates two files: *proxy.cs*, which contains an equivalent interface with the identical signature as the WCF service; *app.config*, which contains the binding information for the client to access the WCF service.



```
C:\Temp>svcutil.exe /config:app.config /out:proxy.cs http://localhost:8080/CTSC/
BookingRequestService?WSDL
Microsoft (R) Service Model Metadata Tool
[Microsoftr .NET Framework, Version 3.0.50727.357]
Copyright (c) Microsoft Corporation.  All rights reserved.

Generating files...
C:\Temp\proxy.cs
C:\Temp\app.config

C:\Temp>_
```

If you open the file proxy.cs, you can see the following class definition in the source code:

```
[System.CodeDom.Compiler.GeneratedCodeAttribute("System.ServiceModel", "3.0.0.0")]
public interface IFlightBookingRequestServiceChannel : IFlightBookingRequestService, System.ServiceModel.IClientChannel
{
}

[System.CodeDom.Compiler.GeneratedCodeAttribute("System.ServiceModel", "3.0.0.0")]
public partial class FlightBookingRequestServiceProxy : System.ServiceModel.ClientBase<IFlightBookingRequestService>, IFlightBc
{

    public FlightBookingRequestServiceProxy()
    {
    }

    public FlightBookingRequestServiceProxy(string endpointConfigurationName) :
            base(endpointConfigurationName)
    {
    }
```

With the help of the proxy file, you only need to create an instance of the proxy class and call the corresponding methods directly from the proxy instance. For our booking-request service, the corresponding code block may look like:

```
// set the binding information
System.ServiceModel.WSHttpBinding binding = new System.ServiceModel.WSHttpBinding();
System.ServiceModel.EndpointAddress endpoint = new

System.ServiceModel.EndpointAddress("http://localhost:8080/CTSC/BookingRequestService");
```

```
// create the proxy using the binding information
FlightBookingRequestServiceProxy proxy = new FlightBookingRequestServiceProxy(binding,
endpoint);

// call the WCF service method, e.g. get all booking requests for our scenario
BindingList<localhost.BookingRequest> requests = proxy.GetBookingRequestList()
```

The Proxy wizard integrated in the Visual Studio for WCF services works in a similar manner. The only difference between it and SvcUtil.exe is that, the proxy wizard integrates the generated files directly into your VS projects, so that you can use the generated proxy classes directly in your project.

Smart Clients

In the last two chapters, we have discussed how to exposing business capabilities via Web services and how to build composite business capabilities from existing ones, e.g. SAP Web services. By using Web services to provide access to the actual business data across disparate backend systems, today's business can build end-to-end applications involving partners, customers and employees. Just as Bill Gates has addressed in his executive email, the largest challenge to avoid the information overload/underload is to create software that streamlines how to find, use and share business information. In today's business computing landscape, business information are often stored and managed by a set of separate LOB applications with separate client applications. Information workers need access to such business data anywhere and anytime to get their work done. Therefore, the software has to be integrated into the information worker's normal working workflow and it should hide the complexity of the data through powerful data analysis, collaboration, reporting and representation features to turn business data exposed by Web services into high-level information and even business opportunities.

Smart Clients can address such challenges. Smart Clients are "intelligent" client applications that can adapt to various clients runtime environments. Through unified front-end integrated with desktop applications or environment, smart clients enable bidirectional connections between the front-ends and the backend LOB applications: smart clients deliver transparently business data from various systems via Web services to the users and push user modifications on the data back to the systems. With smart clients designed for user-specific activities, e.g. based on either the "Role" or the "Task" of the information workers, the clients only display data relevant to the current activities. This property of smart client can efficiently reduce the effect of information overload and makes the information worker more productive with the business data.

In the following sections, we explain the characteristics of Smart Clients by introducing Office Smart Clients together with Visual Studio Tools for the Microsoft Office Systems (VSTO) and our flight-booking scenarios.

What is behind "Smart Clients"?

In order to fully understand the idea behind "smart clients", it is useful to shortly review the concepts and the underlying characteristics behind "thin client"- and "rich client" applications.

Thin client applications are normally browser-based applications that are deployed on Web servers and expose business functionalities to broad audiences, including diverse external audiences. Thin clients are easily to deploy and maintain, because they can centrally managed directly on the Web servers. In addition, thin clients have only minimal software and hardware requirements on client computers, because the most computation takes place on the server side. Nevertheless, thin clients have also some disadvantages. Thin clients are heavily network-dependent; the browser requires network connection to interact with the web server. Also, common application features that we know from desktop applications such as drag&drop, undo/redo, etc, are difficult to implement for thin clients. This reduces the usability of the application considerably.

Rich client applications are designed to take advantage of the local hardware resources and the features provided by the operating system platforms, such as the Microsoft Office client applications. Comparing to thin client applications, rich client applications show better usability and are more responsive to the user actions. Despite of the advantages of rich client application, they have limitations, too. Rich client applications have to be completely deployed to each client computer and are difficult to deploy and maintain. Moreover, the fact that many client applications share components/libraries makes the deployment even more complicates, since any incompatible share component/library can easily break another application that depends on it (so called "DLL Hell").

Smart clients are applications designed to combine the advantages of rich client application, namely the usability, together with the ones of thin client applications, namely the easy deployment and manageability. Smart clients may have very diverse functional requirements depending on the particular business scenario; however, all the smart clients should have some or all of the following characteristics:

- Have a rich user interface: smart client should have rich user interface that fully utilizes the advantages of Windows operation systems. A rich user interface ensures a responsive user experience and therefore a better usability of the application.
- Make use of network resources: smart client should be network enabled and can consume diverse services and data over the network, including Web services.
- Support occasionally connected users: for users who are occasionally connected to the network, smart client should provide such users support to continue to work efficiently when they are offline or when the connectivity is intermittent.
- Support easy deployment and maintenance: smart client should be designed to manage their deployment and maintenance in a much more easy and flexible way like the thin client applications.

Figure 12 illustrates the relationship between the Smart Clients, Thin Clients and Rich Clients again.

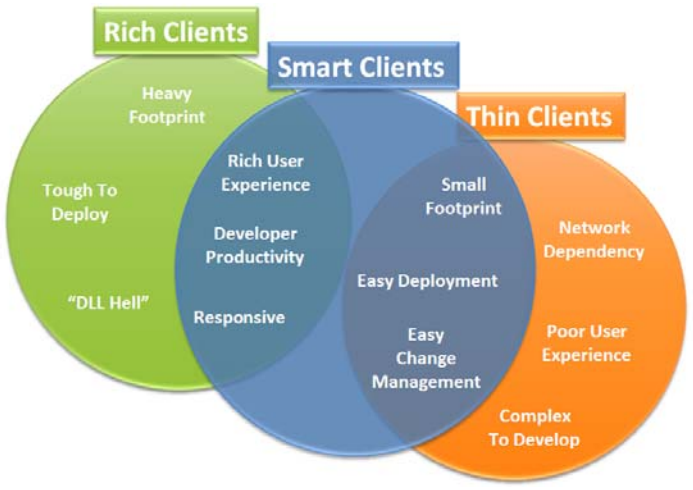


Figure 12: Smart Clients in Comparison with Thin Clients and Rich Clients

As aforementioned, smart clients may vary strongly in design and implementation due to different functional requirements based on the particular business scenarios. Therefore, smart clients may take different forms, which can be broadly classified into three categories:

- **Windows Smart Client Applications** target the desktop platform and are desktop applications that fully utilize the available system resources to provide rich user interfaces. Such smart clients may range from applications deployed via HTTP to very sophisticated applications. With windows smart clients, business can build applications that run on desktop, laptop or tablet PCs and provides functionalities adapting to particular tasks with domain specific capabilities. In general, such smart clients are not associated with particular document type and are suitable to be used as front-end for LOB, financial, or collaborative applications. If you would like to know more about windows smart clients application based on Windows Forms, MSDN provides a specific column on this topic: http://msdn.microsoft.com/netframework/windowsforms/
- **Office Smart Client Applications** target the Microsoft Office System 2003 as platform. With Office smart clients, you can integrate business data exposed by e.g. Web services with the features of Word 2003, Excel 2003, etc. Comparing to the Windows smart clients, Office smart clients are normally document specific and provide context sensitive data as the user works within a document. Furthermore, they provide further features, such as data analysis, collaboration, reporting and presentation features for the business data exposed by Web services. There are a set of key features of Microsoft Office 2003 for building office smart clients: Smart Tags, Smart Documents, and Microsoft Visual Studio Tools for Office (VSTO). In the next section, we will demonstrate how to use VSTO to build context-sensitive Smart Document. For those, who need more

detailed information about VSTO and other office smart clients related topics, they can visit the Office Developer Center on MSDN: http://msdn.microsoft.com/office/tool/vsto/default.aspx

- **Mobile Smart Client Applications** target the smart devices – Pocket PCs, Smartphone, etc. – as the platform. Such smart clients are built upon the .NET Compact Framework, which is a subset of the normal .NET Framework and is optimized for use on the smart devices. With the appropriate tools integrated in Visual Studio, you can develop, debug and deploy mobile smart clients on an emulator or directly on a real device. Mobile smart clients provide typically mobile access to business data and capabilities offered by Web services. On the other hand, it is also suited as front-end to collect data with extended support for offline use. The .NET Compact Framework developer center on MSDN http://msdn.microsoft.com/netframework/programming/netcf/default.aspx provides a comprehensive overview about the development of mobile smart clients.

The most prominent examples for Smart Client applications are the Information Bridge Framework[12] (IBF) and Duet[13] (formerly codenamed "Mendocino") as a joint product of SAP and Microsoft. Both products enable the usage of the Office applications as the front-end for the business processes based on a set of Web services, which enables the direct integration of business data from the backend system into the familiar working tools for information workers.

In this section, we have addressed the main characteristics of smart clients. Smart clients are a comprehensive approach to turn business data into more expressive business information via domain specific capabilities for data analysis and representation. Since we can only cover a very small part of the concept behind "Smart Clients" in this whitepaper, it is recommended to read the following literatures to get further information about Smart Clients:

- http://msdn.microsoft.com/smartclient/ : the entrance portal for Smart Clients at MSDN
- http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnpag/html/scag-ch01.asp: the Smart Client Architecture and Design Guide that covers the architectural challenges for domain specific business scenarios and how to overcome them when building smart clients applications.
- http://msdn.microsoft.com/library/?url=/library/en-us/dnpag2/html/scbatlp.asp: the Smart Client Baseline Architecture

[12] Information Bridge Framework: http://msdn.microsoft.com/office/tool/ibf/default.aspx

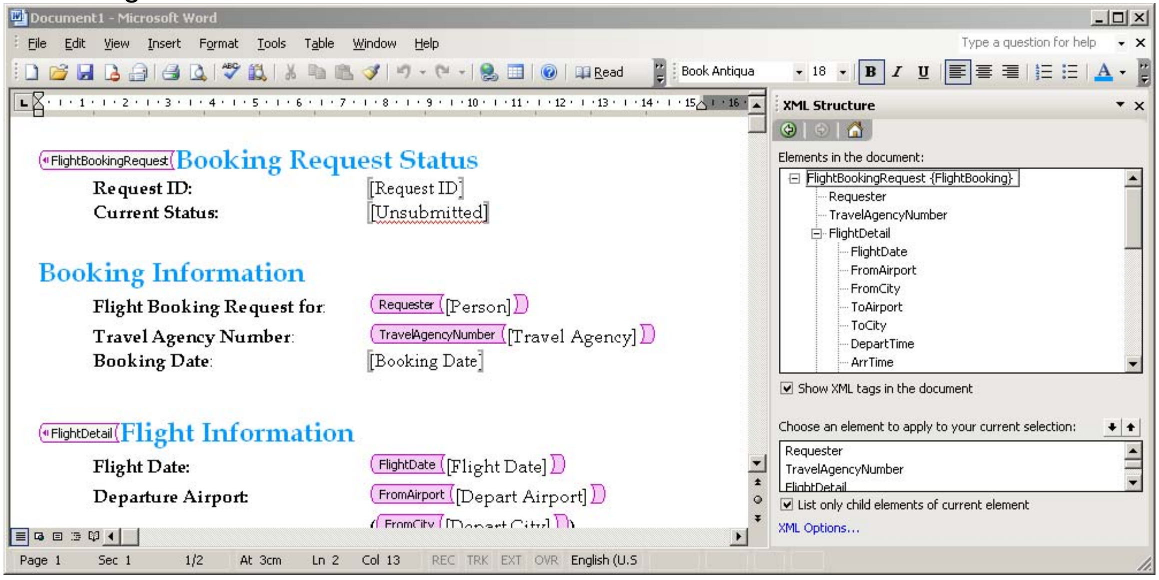
[13] Duet: http://www.duet.com

Toolkit provides a set of guidance to help create Smart Client applications.

Building Word Application using VSTO 2005

In this section, we demonstrate the capabilities of VSTO 2005 by implementing a Smart Document for our flight-booking scenario. In our scenario, a user has to submit a booking request for the flight(s). To assist him in filling the formula directly in Word 2003, we implement a smart document that performs all the actions, such as searching flight connections or submitting the request, directly from inside of the Word document.

After the installation of VSTO 2005, it adds a set of predefined project templates to Visual Studio 2005. Using the project templates, you can create **Word Document/Template**, **Excel Workbook/Template** and **Outlook Add-in**. For our scenario, we use the "**Word Template**" project template to create the corresponding project. The project template adds a new word template document ".dot" and the code-behind class file into your Smart Document project. The next step is to create a new XSD schema and reference the XSD schema with the word template. The XSD schema is used to markup the word template with the elements from the XSD schema. Please consult the VSTO documentation for how to do it. The result after mark-up looks like the following:


Document1 - Microsoft Word
File Edit View Insert Format Tools Table Window Help
Type a question for help
Read
Book Antiqua
18
XML Structure
Page 1 Sec 1 1/2 At 3cm Ln 2 Col 13 REC TRK EXT OVR English (U.S
XML Options...

FlightBookingRequest Booking Request Status

Request ID: [Request ID]

Current Status: [Unsubmitted]

Booking Information

Flight Booking Request for: Requester [Person]

Travel Agency Number: TravelAgencyNumber [Travel Agency]

Booking Date: [Booking Date]

FlightDetail Flight Information

Flight Date: FlightDate [Flight Date]

Departure Airport: FromAirport [Depart Airport]

FromCity [Depart City]

Elements in the document:

- FlightBookingRequest {FlightBooking}
  - Requester
  - TravelAgencyNumber
  - FlightDetail
    - FlightDate
    - FromAirport
    - FromCity
    - ToAirport
    - ToCity
    - DepartTime
    - ArrTime

☑ Show XML tags in the document

Choose an element to apply to your current selection:

Requester
TravelAgencyNumber
FlightDetail

☑ List only child elements of current element

In the following sections, we demonstrate how to create customized action pane in Word 2003 and how to manipulate the content of the word document programmatically.

Developing Customized ActionPane

A significant improvement of VSTO 2005 comparing to the older version is that, VSTO 2005 supports customized **ActionPane** in

Word/Excel/Outlook. The way to create customized **ActionPane** is also very straightforward and is similar with the initialization of a WinForm-container. You only need to instantiate the customized user controls and add them to the **ActionPane** container, just as the following code does:

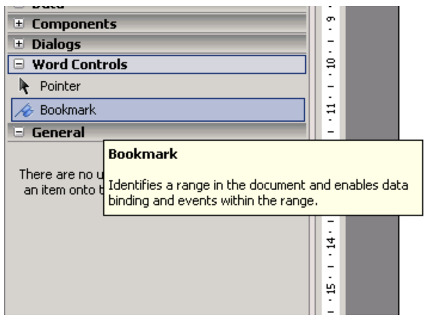
```
/// <summary>
/// Initializes user controls, adds them to the Actions Pane.
/// </summary>
private void CreateActionsPane()
{
        // Create all of the user controls used in the Actions Pane.
        helpCtrl = new HelpControl(ThisApplication.ActiveDocument);
        planCtrl = new FlightPlanControl(this);
        // Add user controls to the ActionsPane.
        ActionsPane.Controls.Add(helpCtrl);
        ActionsPane.Controls.Add(queryCtrl);
        ……
}
```

Accessing Word Document Programmatically

In General, there are two ways to access the content in a Word document. After marking-up the Word document with an XSD schema, the text segments in the word document are marked-up with some XML elements from the XSD schema. VSTO 2005 has built-in support for Word document marked-up with XSD schema. For each XML elements defined in the XSD schema, you can directly access them programmatically, just as the following code shows:

```
private void PopulateFlightHop(int rowId, localhost.BAPISCOHOP hop)
{
        // select the xml node
        // VSTO generate for each XML element in the schema a correpsonding class property 
with the same name
        Word.XMLNode airlineNode =
          Globals.ThisDocument.FlightHopeAirlineNodes[rowId];
        // change the content of the XML node
        airlineNode.Text = hop.AIRLINE;

        Word.XMLNode planeNode = Globals.ThisDocument.FlightHopePlaneTypeNodes[rowId];
        planeNode.Text = hop.PLANETYPE;

        Word.XMLNode fromAirportNode =
Globals.ThisDocument.FlightHopeFromAirportNodes[rowId];
        fromAirportNode.Text = hop.AIRPORTFR + ",\n" + hop.CITYFROM + ",\n" + hop.CTRYFR;
        ……
}
```

Another way to access the content in a Word document is to use Bookmark in the Word document. With VSTO 2005, you can directly create a Bookmark in your Word document by dragging a **Bookmark** item from the toolbox onto the design surface, as follows:



Bookmark

Identifies a range in the document and enables data binding and events within the range.

To access a predefined bookmark in a Word document, you can directly access the bookmark variable in your code. For example, we have defined a bookmark "bookingData" in the Word document, and then the corresponding bookmark can be accessed in the code-behind class as follows:

```
this.doc.bookingDate.Text = System.DateTime.Today.ToLongDateString();
```

In this section, we have briefly introduced some features of VSTO 2005 and how to use it to create an Office smart client application - in our case a smart document for Word. The full implementation of the smart document is available in the Visual Studio solutions for the flight-booking scenario.

Summary

Using the next-generation technologies from .NET Framework 3.0, especially Windows Communication Foundation and Windows Workflow Foundation, together with the enterprise services exposed by SAP NetWeaver and the Smart Clients technologies, customers can build domain specific applications that fit the individual functional requirements of diverse scenarios in the daily business. Since SAP NetWeaver Application Server 6.40, customers can expose each ABAP-based functional module, including the customized ones, via standard-based Web services. Based on the enterprise services offered by SAP NetWeaver, WF lets you to create workflows that coordinate the execution of various activities either as a sequential workflow or a state-machine workflow. WF allows you to integrate the functionality of workflow into every possible Windows applications, e.g. Windows services, desktop application or even Web services. WCF provides a robust but flexible basis for creating as well as consuming services with different accessing channels in a unified manner. Moreover, with Smart Clients technologies, especially the development tools integrated in Visual Studio 2005, e.g. VSTO 2005, you can build domain-specific applications for your business that fulfill the functional requirements of the specific scenarios.

In this whitepaper, we have implemented a flight-booking scenario to demonstrate how to utilize the new technologies from the backend SAP NetWeaver via the middleware with .NET Framework 3.0 technologies, especially WCF and WF, to the front-end with Smart Clients technologies to demonstrate how the new techniques can work together to provide workflow enabled capabilities for your business. The full implementation is done with Visual Studio 2005 and is available for downloading on http://www.microsoft-sap.com. Since the implementation is a Proof-of-Concept implementation and has its focus on the feasibility of the workflow-based end-to-end applications using the newest technologies, we have not taken security and transaction into account in our implementation. Therefore, we give some considerations to these two aspects here.

Consideration concerning Security

In an SOA-based world of enterprise computing, the most communication between the service provider and the service consumer take place via Internet or intranet. Therefore, security plays an important role in the service-level agreements between various parts in the computing infrastructure. For our flight-booking scenario, two security-related aspects are of special interest: authentication and message integrity.

SAP NetWeaver AS provides different security features to secure Web services. As authentication mechanisms, SAP NetWeaver supports basic authentication with username and password, authentication using certificates and SAP specific authentication with SAP Logon Ticket for Single Sign On. WCF has also built-in support for basic authentication using username and password as well as certificate-based authentication. However, WCF has no native support for SAP Logon Ticket. Instead, WCF follows another more generic way for SSO, namely *federated security*. It allows for separating the implementation of a service with its authentication procedures for clients consuming the service. In addition, based on the industry standard WS-Federation[14], federated security creates a federated security realm across several systems, networks and organizations with different security realms. WCF provides out-of-box support building and consuming applications that employ federated security.

The other security-related aspect is message integrity. To ensure the transport level security, SAP NetWeaver AS uses HTTPs with SSL, which is supported by WCF, too. However, this mechanism is not flexible and can only be used to create *point-to-point* transport security for SOAP messages. A much-secured way is to use XML Signature and XML Encryption defined in WS-Security to ensure *end-to-end* security that may span several hops across the network. Both SAP NetWeaver AS and WCF have out-of-box support for this security feature.

One issue that has to be figured out is the extensibility of both SAP NetWeaver AS and WCF regarding security features. SAP NetWeaver AS supports the adoption of plug-ins for additional transport protocols, Web service specifications and other message-level features. WCF has also an extensible programming model, which allows you to modify and extends its runtime components to precisely control and extend the capabilities of the services, e.g. customized binding, channels or security components. Please consult the documentation of both products to get more information about this feature.

Consideration concerning Transaction

Another crucial aspect of enterprise business applications is the transactional behaviors of the solutions. In general, we can distinguish between transactional behaviors in the backend systems, in our case, the SAP ECC, and the transactional behaviors in the middleware, namely the workflows based on WF, although both are related to one another at runtime and contribute together to the transactional behaviors of the entire business application.

[14] IBM & Microsoft, *Federation of Identities in a Web Services World,* http://msdn.microsoft.com/webservices/webservices/understanding/advancedwebservices/default.aspx?pull=/library/en-us/dnglobspec/html/ws-federation-strategy.asp

For BAPI functions that provide writing operations on SAP data, such as creating a new flight connection or updating an existing flight booking, SAP provides the necessary support for either committing the result or canceling the operation being performed. However, this mechanism is no longer valid for Web services-based processes. Because Web services are stateless and each BAPI Web service invocation takes place in a separate context, the invocation of the BAPI function **BAPI_TRANSACTION_COMMIT** or BAPI_TRANSACTION_ROLLBACK has no impact on the transactional behavior of the BAPI call via Web services. To resolve this side effect due to the adoption of Web services, you have to activate the session-oriented communication between the SAP NetWeaver AS and your client applications. The session-oriented communication works with the help of HTTP sessions. The basic idea is to announce explicitly the start and the end of a HTTP session through the URL string. The following code demonstrates how it works and has been successfully tested for Web services that are exposed via the SOAP Processor (the approach used since SAP WAS 6.20). The key step in the sample code is to define a common-used CookingContainer for all the Web services proxies.

```
// CreateTrip is the proxy class for creating the Flight trip
// CommitTransaction is the proxy class for commiting the transaction

// ensure that the both proxies use the same CookingContainer, which is used by the SAP
// Web services to hold the session state
System.Net.CookieContainer cookiesContainer = new System.Net.CookieContainer();
CreateTrip.CookieContainer = cookiesContainer;
CommitTransaction.CookieContainer = cookiesContainer;

// get the original URL without session parameter
string tmpUrl = CreateTrip.Url;
// indicate explicitly the start of the current session
// this must happen before the first Web service call
CreateTrip.Url = tmpUrl + "&session_mode=1";
// execute writing operations on the SAP ECC via BAPI Web services
FlightTrip.BAPIPAREX[] ext = CreateTrip.FlightTripCreate(extensionIn, trip, ref passengers, ref
results, out price, out agency, out tripnumber);
// indicate explicitly the end of the current session
CommitTransaction.Url = tmpUrl + "&session_mode=2";
// make the last Web service call
Commit.BAPIRET2 re = CommitTransaction.BapiServiceTransactionCommit("");
```

However, the same code snippet does not work with the 6.40 Web services in our test environments, which are created using the *Create Web Service Wizard* in SAP ECC (cf. the section "*Accessing Web Services from SOAP Runtime in SAP NetWeaver 6.40*" in this whitepaper). For committing Web services that are created by the wizard on SAP NetWeaver AS 6.40, you can e.g. create a new BAPI function that combines several BAPI functions one after another and call at the end either directly the **BAPI_TRANSACTION_COMMIT** BAPI function or commit directly the transaction with "**COMMIT WORK**".

The transactional behavior of a WF-based workflow is compensation-based. Compensation defines the execution of business logic that results from a business exception. For WF-based workflows, WF provides several out-of-box activities to define the compensation of the workflow. At runtime, WF uses the methods defined in the interface **ICompensatableActivity** of a completed activity to perform the compensation. Currently, the only activity that implements this interface is the **TransactionScopeActivity**. However, you can write custom activities that support compensation using this interface. By default, the compensation code of any nested transaction runs automatically by invoking compensation of all nested children in the reverse order of their completion. If you need to invoke selectively compensation among the completed activities, then you can use explicitly the out-of-box activity **CompensateActivity** to do it. For more information about how to control the compensating behavior of your workflow, please consult the WF documentation.