B1WS: B1 Web Services wrapper

Applies to:

SAP Business One application and SDK DI Server development.

Summary

This article describes the B1WS tool for SAP B1: this tool leverage on the B1 DI Server, on the .NET platform and on Microsoft Visual Studio .NET environment. It can be also used in Java environments as for example Netbeans.

Author: SAP B1 Solution Architects & Development team

Company: SAP

Created on: December 2007

Updated on: 18 February 2010

Table of Contents

Introduction

The SAP Business One SDK provides several programming interfaces to build solutions on top of the SAP Business One application. In this document we will present a wrapper, called B1 Web Service (B1WS) that makes easier and quicker to develop add-ons based on the DI Server.

SAP Business One – DI Server (Data Interface Server) is a Component Object Model (COM) service running on a server that enables multiple clients to access and manipulate SAP Business One company database, using SOAP version 1.1 messages.

B1WS will expose the DI Server functionality as Web Services to provide a better usability. Developers only need to add the web references to the B1WS web services (presented through wsdl files) and use the services with the auto generated object model. Development time is then reduced with the use of B1WS.

This wrapper has been developed using Visual Studio .NET 2005 but can also be used with Java environments.

B1WS is delivered within the SDN license structure and therefore no support is provided.

A setup will allow you to easily install and run the application. You can also download the complete source code.

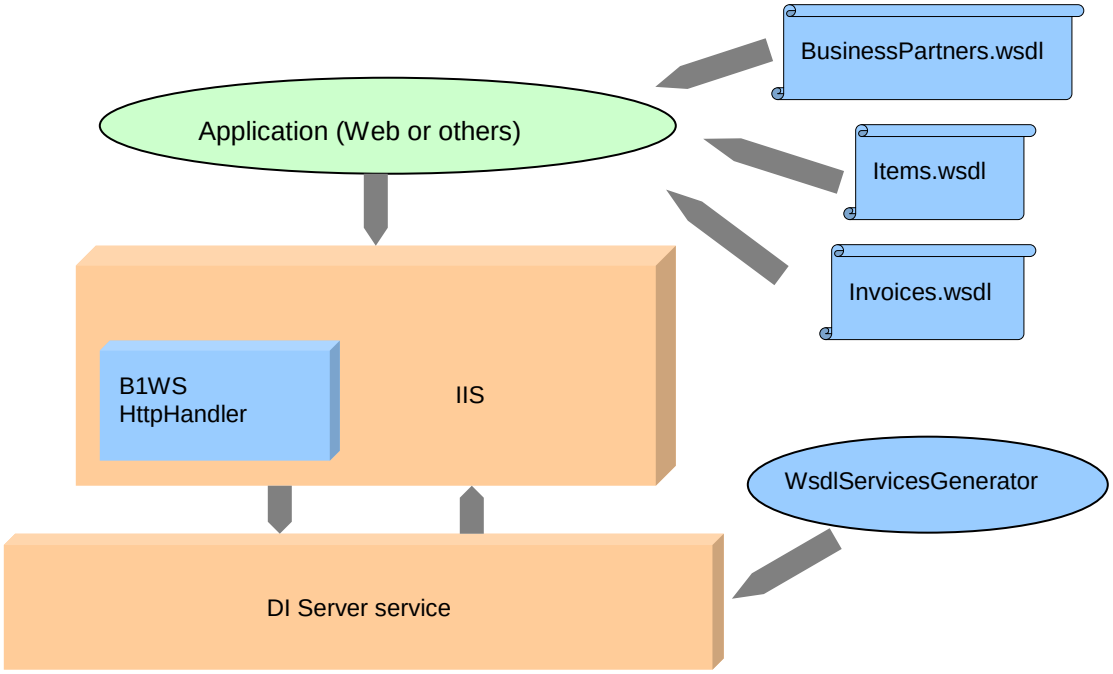
Architecture

B1WS is composed by:

- a list of wsdl files, each wsdl file representing one service
- an http handler
- a wsdl services generator

The figure below shows the different components and their integration.

In blue color are represented the B1WS components, in orange color the Windows and SAP Business One components and in green color a partner application using B1WS.



B1WS main component is B1WSHttpHandler, an Http Handler registered inside IIS (Internet Information Services) as a virtual directory. B1WSHttpHandler automatically intercepts all calls to B1WS, redirects the requests to the DI Server and returns the SOAP response to the client.
B1WSHttpHandler needs to be located in the same server machine as the DI Server.

Partner applications need to add a Web Reference (pointing to the corresponding wsdl file) per each DI Server service they use. In the client application the location of the B1WS server must be defined in order to redirect the calls to the B1WSHttpHandler. In .NET for example Web References have a configuration file (web.config) where the server address can be configured.
Partner applications can be located in a server machine or a client machine having a network access to the server.

No formatting in the http handler is needed, the wsdl files containing the services definitions are already formatted as required by the DI Server. The Web Services will then automatically build the requests following the right format.

The WsdlServicesGenerator is an application that automatically recreates all B1WS wsdl files. The wsdl files provided with the B1WS setup are the ones corresponding to a basic SAP Business One database without user defined fields. If the structure of the database your application will be running on contains user defined fields in B1 tables you will need to regenerate the wsdl files in order to have access to all your user defined fields from the services.

Installation

The installation of B1WS is straightforward: run the setup and accept the SDN license terms. Once installed, the wrapper files will be saved by default under:

C:\Program Files\SAP\SAP Business One Web Services

and accessible under:

Start -> All Programs -> SAP Business One -> Development Tools -> B1 Web Services

Configuration

After having B1WS installed in your server machine you need to configure the different pieces composing the wrapper.

DCOM configuration for DI Server

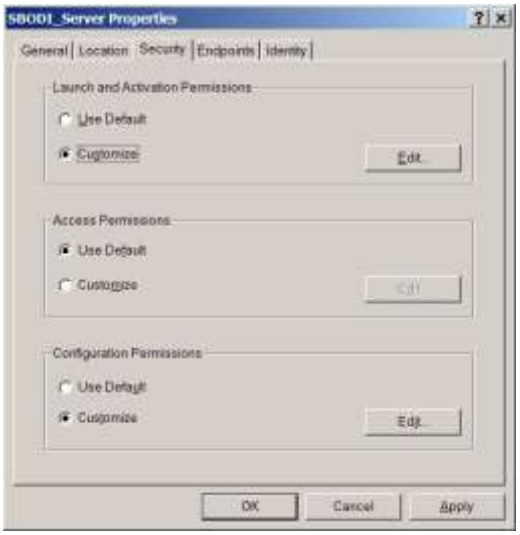
In order to be able to access to the DI Server from the http handler the DCOM configuration of the DI Server must be modified.

The B1WSHttpHandler while running creates a new instance of the DI Server and redirects all calls from your application to the DI Server. For doing it the http handler uses the ASP.NET account and therefore this account must be configured to have the right security access.

In order to give the right security access to the ASP.NET account you will need to open the DCOM configuration tool to add the ASP.NET user the SBODI_Server Launch, Activation and Configuration Permissions.

Here you have the detailed steps to follow:

- Go to the Start -> Run menu item.
- Type in "DCOMCNFG" and hit enter.
- This should load the "Component Services" MMC (you can also open it from Administrative Tools - Component Services)
- Expand "Component Services"
- Expand "Computers"
- Expand "My Computer"
- Select the "DCOM Config" item
- Select the "SBODI_Server" item.
- Right click and select Properties
- Select the Security Tab and you should see the following:

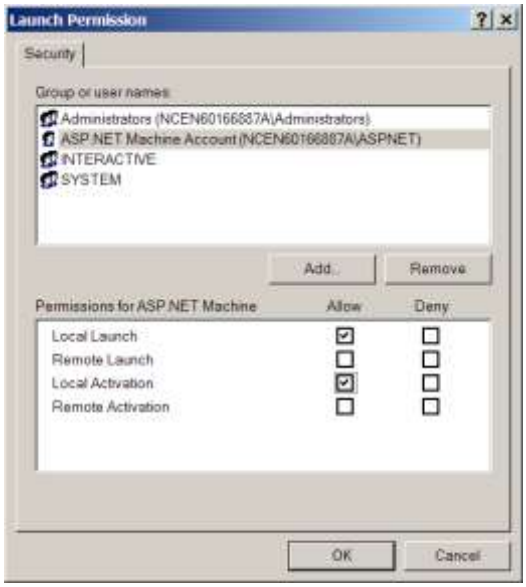


Set Launch and Activation Permissions

- Under "Launch and Activation Permissions" select the "Customize" option.
- Click the "Edit" button

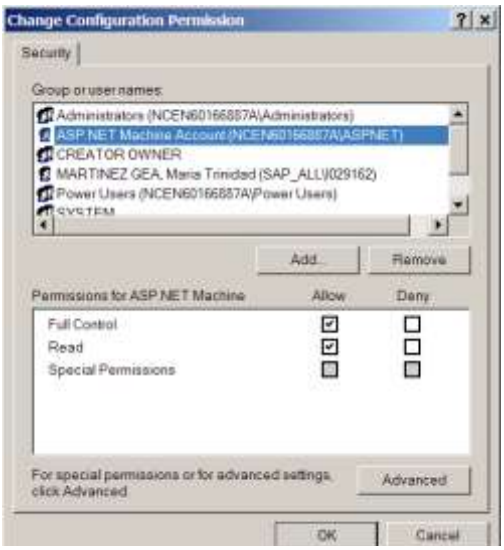
- Click the "Add" button to add a new account to the list.
- On the dialog that is displayed click the Locations button. In this dialog scroll the list to the top (sometimes the first item is not visible) and select the first item which is your computer name.
- On the dialog that is displayed enter "ASPNET" as the account name (make sure location is set to the name of the computer that IIS is on) on Windows XP or if you are running on Windows 2003 Server you must enter the account that the Application Pool is running as, by default "Network Service"

- Click the OK button
- Now make sure you select the Local Launch and Local Activation options for the "ASP.NET Machine Account" or the account that is the application pool identity (by default Network Service)



Set configuration permissions:

- Under "Configuration Permissions" select the "Customize" option.
- Click the "Edit" button
- Click the "Add" button to add the ASPNET account (follow the same steps as for the creation of the ASPNET account in the Launch and Activation Permissions steps above).
- Click the OK button
- Now make sure you select the Full Control option for the "ASP.NET Machine Account" or the account that is the application pool identity (by default Network Service)



Now your DI Server is ready to be called from the B1WSHttpHandler.

Configuration automatically done during installation

This section describes the configuration steps done by B1WSSetup during installation, if installation finished successfully you don't have to manually run them. If one of these steps fails because your machine doesn't mach the required configuration they will show you a message box asking you to proceed manually. To do it please follow this section.

B1WSHttpHandler configuration

In order to intercept all B1WS requests IIS must have a virtual directory pointing to B1WS http handler.
How to add a virtual directory into IIS:

1. Open IIS (Control panel -> Administrative Tools -> Internet Information Services).
2. Right click under Default Web Site (under local computer, Web Sites) and select New -> Virtual Directory
3. A wizard will open with 3 steps:
   3.1. Alias: enter B1WS
   3.2. Directory: point to the directory under B1WS installation called B1WS (The one that contains App_Code directory with B1WSHandler.cs file inside it).
   3.3. Access Permissions: select Read and Run scripts

Note: You need IIS installed and running in the server machine in order to be able to use B1WS.

Register ASP.NET into IIS

In order to debug your application together with the B1WSHttpHandler you need to:

- ❑ Register ASP.NET into IIS.

Run "aspnet_regiis /i" command from the Windows/Microsoft.NET/Framework/v2…" directory.

Note: You need .NET Framework version 2 to be able to register ASP.NET into IIS.

How to develop using B1WS in a .NET environment

In order to use B1WS you have to add a web reference for each DI Server Service you need to use. All wsdl files will be located inside the Web References directory under your B1WS installation directory. Once the reference added you will be able to access to all services provided by B1WS from you code and to manage them as simple commands with input and output parameters.
You can also have all B1WS web services automatically listed in .NET environment by opening the Add Web Reference window and then:

- Selecting “Web services on the local machine”, if your development environment is located in the same machine as the B1WS Http Handler
  or
- Typing the name of the machine in the URL text box.
  Ex: http://serverName/B1WS/

Samples

Several samples of how to use different B1WS services are provided with the B1WS installation. The samples are in Vb.NET language but C# is also suitable for developing and converting them to C# shouldn’t be a problem.
In this document 2 very simple cases are show: login and add business partners in both programming languages C# and Vb.NET.

Login

In order to connect to the DI Server you need to call the login method. Login and Logout operations are contained in the Login.wsdl file. After adding a web reference to this file you will be able to use the login and logout methods.

Following picture shows a code sample calling the login method.

```
// Create Service WebReference
LoginWebRef.LoginService loginService = new LoginWebRef.LoginService();

// service command login call
string sessionID = login.Login(
          "localhost",
          "COM_OBS_AUT_TEST",
          LoginWebRef.LoginDatabaseType.dst_MSSQL2005, true,
          "sa", "xx",
          "manager", "manager",
          LoginWebRef.LoginLanguage.ln_English, true,
          "localhost:30000");
```

Figure 1: Login C# sample

```
‘ Create Service WebReference

Dim login As LoginService = New LoginService

' service command login call

sessionID = login.Login(
        "localhost",
        "COM_OBS_AUT_TEST",
        LoginWebRef.LoginDatabaseType.dst_MSSQL2005, true,
        "sa", "xx",
        "manager", "manager",
        LoginWebRef.LoginLanguage.ln_English, true,
        "localhost:30000")
```

Add a BusinessPartner

Each one of the objects and services offered by B1WS are represented by a wsdl file. You have to add a Web Reference per each one of them you need to use.

After adding the Web Reference you simply have to use the objects and commands offered by the service. Here you have some lines showing the steps in your code to call one of the commands offered by the BusinessPartnersService:

```
// Create Service WebReference
BPWebRef.BusinessPartnersService bpsService = new
BPWebRef.BusinessPartnersService();

// MsgHeader class
BPWebRef.MsgHeader msgheader = new BPWebRef.MsgHeader();
msgheader.SessionID = DIServerSessionID;
msgheader.ServiceName = MsgHeaderServiceName.BusinessPartnersService;
msgheader.ServiceNameSpecified = true;
bpsService.MsgHeaderValue = msgHeader;

// fill business partner class
BPWebRef.BusinessPartner bp = new BPWebRef.BusinessPartner();
bp.CardCode = "MyBPCardCode";
…

// service command call
BPWebRef.BusinessPartnerParams bpParams = bpsService.Add(bp);
```

Figure 3: Business Partners add C# sample.

```
' Create Service WebReference
Dim bpsService As BPWebRef.BusinessPartnersService = New
BPWebRef.BusinessPartnersService()

' MsgHeader class
Dim msgheader As BPWebRef.MsgHeader = New BPWebRef.MsgHeader()
msgheader.SessionID = DIServerSessionID
msgheader.ServiceName = MsgHeaderServiceName.BusinessPartnersService
msgheader.ServiceNameSpecified = True
bpsService.MsgHeaderValue = msgHeader

' fill business partner class
Dim bp As BPWebRef.BusinessPartner = New BPWebRef.BusinessPartner()
bp.CardCode = "MyBPCardCode"
…

' service command call
Dim bpParams As BPWebRef.BusinessPartnerParams = bpsService.Add(bp)
```

Figure 4: Business Partners add Vb.NET sample.

Notes on the sample code

1. One very important point not to forget when using B1WS is to fill the MsgHeader class with the relevant information regarding the DI Server session ID you are connected to and the name of the service you want to access. Each service has a MsgHeaderValue property to be set with the MsgHeader class.
2. Per each object property with a type different of string an associated Boolean property with the same name but finishing with “Specified” is automatically created by .NET environment. You are forced to set this property value to “true” if you want the associated property to be correctly set in the DI Server SOAP message automatically formatted.

Important to know

User Defined Objects (UDOs)

UDOs can be managed with B1WS using a specific service available in DI Server per each UDO in your database (as DI API can access UDOs through the GeneralService).

In order to be able to access your UDOs through B1WS you have to generate a specific wsdl per each one of your UDOs.
Run the WsdlServicesGenerator.exe and select the UDOs you want to generate a wsdl file for. This application will ask you for your database details in order to connect to the DI Server and obtain all your database details. The generator creates a directory called wsdlFiles close to the exe, copy all the wsdl files to the WebReferences directory in your B1WS installation directory.

Once the wsdl files generated per each one of your UDOs you will be able to use the available methods (Add, Update,…) as for any other B1WS service.

Important Notes:

- UDO services in DI Server will only work for UDOs having a UDO Name without spaces. If your UDO Name contains spaces please change it and remove the spaces before running the WsdlServicesGenerator.exe.
- DI Server service needs to be restarted after you create or modify a User Defined Object.
- Every time you regenerate new wsdl files you need to refresh your web references of your development environment in order to have the latest web services version available for your coding.
- Don't forget to copy the new generated wsdl files to the WebReferences directory in your B1WS installation directory.

User Defined Fields

Wsdl files provided by B1WS contain the schema of the commands and objects offered by each service based on a standard B1 database (without User Defined Fields).

If the structure of the database your application will be running on contains user defined fields in B1 tables you have to regenerate the wsdl files in order to have access to all your user defined fields from the services.

The regenerated files will contain all standard properties plus a new property per each User-Defined Field you added in the corresponding object. The name of the property is the name of your user defined field ("U_xxx").

All wsdl files can be regenerated by simply running the WsdlServicesGenerator.exe application installed with B1WS setup. This application will ask you for your database details in order to connect to the DI Server and obtain all your database details. The generator creates a directory called wsdlFiles close to the exe, copy all the wsdl files to the WebReferences directory in your B1WS installation directory.

Important notes:

- DI Server service needs to be restarted after you create new User-Defined Fields.
- Every time you regenerate new wsdl files you have to refresh your web references of your development environment in order to have the latest web services version available for your coding.
- Don' forget to copy the new generated wsdl files to the WebReferences directory in your B1WS installation directory.

Web References available

B1WS wrapper offers as web references the services exposed by the DI Server (all DI API services plus a long list of DI API objects). B1WS cannot offer access to all DI Server objects, only to those objects that are exposed as services by the DI Server. B1WS wrapper builds the wsdl files automatically from the xsd information returned by the DI Server, therefore only DI Server services (defined with a standard xsd format) can be offered.

The offered services are listed here below classified in 2 lists: services for DI objects and DI Services.

DI Objects

- `BillOfExchangeTransactionsService`
- `BudgetsService`
- `BusinessPartnersService`
- `CorrectionInvoiceReversalService`
- `CorrectionInvoiceService`
- `CorrectionPurchaseInvoiceReversalService`
- `CorrectionPurchaseInvoiceService`
- `CreditNotesService`
- `DeliveryNotesService`
- `DownPaymentsService`
- `DraftsService`

- DynamicSystemStringsService
- IncomingPaymentsService
- InventoryGenEntryService
- InventoryGenExitService
- InvoicesService
- ItemsService
- JournalEntriesService
- JournalVouchersService
- MaterialRevaluationService
- OrdersService
- PaymentDraftsService
- ProductionOrdersService
- ProductTreesService
- PurchaseCreditNotesService
- PurchaseDeliveryNotesService
- PurchaseDownPaymentsService
- PurchaseInvoicesService
- PurchaseOrdersService
- PurchaseReturnsService
- QuotationsService
- ReturnsService
- SalesForecastsService
- SalesTaxAuthoritiesService
- ServiceCallsService
- SpecialPricesService
- StockTransferDraftService
- StockTransferService
- UserFieldsMDService
- UsersService
- VendorPaymentsService
- WithholdingTaxCodesService
- WizardPaymentMethodsService

DI Services

- AccountCategoryService
- AccountsService
- AlertManagementService
- AlternativeItemsService
- ApprovalStagesService
- ApprovalTemplatesService
- BankStatementsService
- BOEDocumentTypesService
- BOEInstructionsService
- BOEPortfoliosService
- BranchesService
- BusinessPartnerPropertiesService
- CashDiscountsService
- CertificateSeriesService
- CompanyService
- CountriesService
- CustomsDeclarationService
- DepartmentsService

- DimensionsService
- DistributionRulesService
- DNFCodeSetupService
- DunningTermsService
- EmployeeRolesSetupService
- ExternalCallsService
- FinancialYearsService
- FormPreferencesService
- MaterialRevaluationFIFOService
- MessagesService
- NatureOfAssesseesService
- NCMCodesSetupService
- PredefinedTextsService
- ProfitCentersService
- ProjectsService
- ReportFilterService
- ReportLayoutsService
- SalesOpportunityCompetitorsSetupService
- SalesOpportunityInterestsSetupService
- SalesOpportunityReasonsSetupService
- SalesOpportunitySourcesSetupService
- SectionsService
- SeriesService
- ServiceCallOriginsService
- ServiceCallProblemTypesService
- ServiceCallSolutionStatusService
- ServiceCallStatusService
- ServiceCallTypesService
- StatesService
- UserMenuService

Please note that in every new release of SAP Business One DI Service new services and access to DI API missing objects are added, for example SAP Business One 8.8 version exposes 34 new services and 12 new objects.

How to debug B1WSHandler

If you need to debug the B1WSHandler just:

- ❑ Open a VS environment and open the B1WSHandler.cs file on it.
- ❑ Go to Tools -> Attach to Process … and select the aspnet_wp.exe process.
- ❑ Put a break point inside the B1WSHandler.cs code, next time your application calls B1WSHandler the breakpoint will be activated.

Limitations

- ❑ Not all DI Server objects are available through B1WS as Web References, only those that have been exposed by DI Server as Services. Please refer to chapter Web References available for more details on the current available B1WS Web References.

- ❑ B1WS must be installed on the same machine as the DI Server (to avoid DCOM security problems)
- ❑ Batch operation is not supported

Note on B1WS source code

You can download from SDN the complete source code of B1WS. It is not needed in order to use B1WS to develop your applications in top of DI Server as the setup contains all needed components. Nevertheless you can use the code to customize the generator to your own needs or simply to understand how it works.

B1WS code contains two Microsoft Visual Studio 2005 projects:

- B1WS

Http handler that must be pointed out from IIS as a virtual directory. This is the core of the B1WS wrapper as the http handler will redirect all B1WS requests to the DI Server and will return the response from DI Server to your application.

- WsdlServicesGenerator

This project is in charge of the wsdl files generation. After you install B1WS the exe obtained from compiling this project is copied into the WsdlServicesGenerator directory, by simply running it all wsdl files for your specific database are generated. You don't need to modify this code in order to use B1WS but you can use it to customize the generator to your needs.

Related Content

You can find other tools related with SAP Business SDK development at:

B1 Development Environment, compendium of software tools that make easier and quicker to develop and package add-ons based on SAP B1 SDK interfaces.

B1 Test Environment, set of software tools helping to profile the usage of the SAP B1 SDK interfaces by a solution. These tools are also used by SAP during solution certification phase.

B1 DI Event Service, service that will provide you with events notification related to SAP Business One DI API objects through a listener-based interface directly in your addons.

B1TestComposer, tool allowing to easily record and replay test scripts in order to automate the Testing process of your addons.

Copyright