Data Level Integration with SAP Business One SDK

Applies To:

SAP Business One / SAP Business One SDK. For more information, visit the Business One homepage.

Summary

This article shows the basic mechanisms provided by SAP Business One SDK to support data level integration to a SAP Business One system together with a few generic scenarios and possible approaches.

A few real world integrations are also described in the context of the described scenario.

Author: SAP Business One Solution Architects

Company: SAP

Created on: 01 June 2005

Last Update: May 2009

Table of Content

Introduction

Data level integration is often the first step an ISV does to integrate its own proprietary solution with SAP Business One. In this case the cost of integration is limited and the existing GUIs are unaffected by this integration. Of course the resulting system does not look integrated but that can be OK in some business cases, or it cannot just be possible to further integrate the systems, or it can be a first acceptable step towards a deeper integration that includes GUI too.

In this first step the resulting overall system is then composed of two separated systems each of them with its own data level, business logic level and presentation level and the integration is kept at data level. The mechanisms that SAP Business One SDK provides in order to support this integration are:

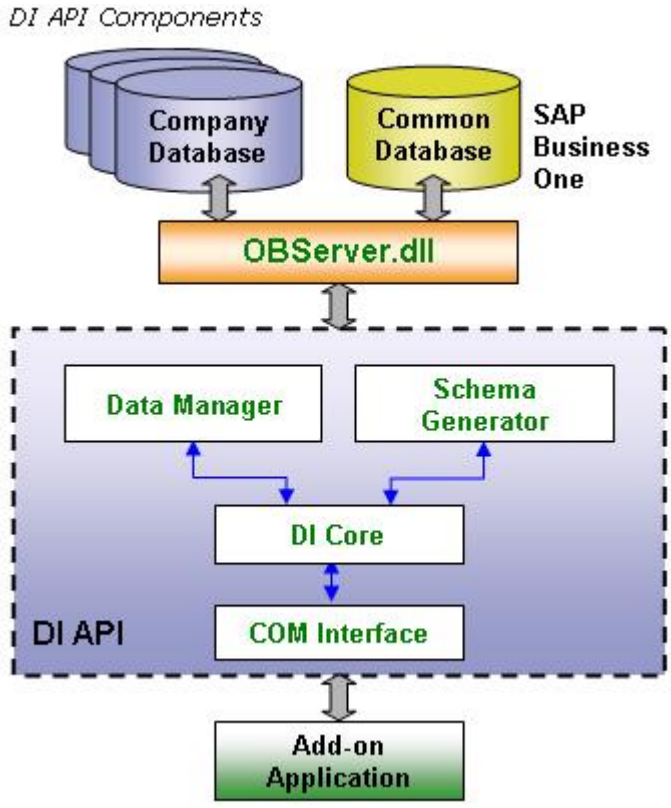
- SAP Business One DI-API
- SAP Business One DI Server

SAP Business One DI-API

DI-API exposes an interface to the SAP Business One objects by which it is possible to read, browse, create, update and delete objects in a SAP Business One database, plus carry out a set of some advanced metadata and XML operations. DI-API internally handles communication with the underlying SAP Business One database, ensures full data validation, and automatically populates default field values based upon the business rules of SAP Business One.

In fact by using the DI-API the external system not only accesses the SAP Business One database but activates all the business logic underlying SAP Business One itself, ensuring coherence of the data in the SAP Business One database itself against the SAP Business One business logic. Accessing directly the SAP Business One DB in write operations – e.g. with an ODBC connection – is against the programming guidelines of SAP Business One SDK as it can introduce inconsistencies in the system data.

DI-API is implemented as a DLL based upon Microsoft COM. It can be easily used with any COM or .NET - compliant development environment including Microsoft Visual Studio or Visual Studio .NET.



A client application loads then the DI-API and by using the objects exported accesses the underlying database tables through the business logic of SAP Business One itself. The picture above describes the architecture of the DI-API.

The DI-API acts in fact for the external system as a COM-based façade encapsulating and hiding many internal services providing:

- access to the database and metadata functions (Data Manager)
- advanced XML functionality (Schema Generator)
- and finally the whole SAP Business One business logic (OBServer)

Note that DI-API is a mono-direction and synchronous interface. Data integration requires also asynchronous interfaces to receive notifications of creation, deletion and update of objects in the SAP Business One DB. SAP Business One supports this necessity by exposing a stored procedure called Transaction Notification. This procedure is always called by Business One business logic after a successful transaction and then can be used to notify them (see here for more details: Transaction Notification).

Note also that DI-API is also available as a set of Java wrapper classes (JCO) for applications running in a JVM. This wrapping is made via a Java Native Interface (JNI) wrapper and then requires a Windows operating environment.

Important note

Solutions are required to use DI interfaces to access data in the database - especially in write accesses. Direct access to the underlying data in the database is a dangerous operation that can introduce inconsistency and possibly results in illegal behavior in the system, and as such is not supported by SAP. Besides, this kind of unsupported access is conflicting with the rules applied by SAP during solution certification process.

- Note that DI interfaces abstracts the programmer from having to understand what an update to SAP Business One means to the database: for instance the DI API understands all the implications in many tables of SAP Business One behind a creation of a new a Sales Order.
- Besides, the set of tables and columns is not ensured to be maintained when moving from a version to the next one. In contrast to that, DI interfaces are maintained compatible across different versions. That means that if a solution is implemented on top of the DI interfaces, it can remain unchanged when SAP Business One is upgraded to a new version.
- Finally, programming on top of the DI interface ensures the portability of a solution across all the databases supported by SAP Business One, as the DI interface abstracts from the contingent RDBMS. Currently supported RDBMS are: Microsoft SQL Server, IBM DB2 and Sybase ASE.

SAP Business One DI Server

DI Server provides a SOAP interface to SAP Business One. The set of operations allowed by this interface is a subset of the methods exported by DI-API. Basically it allows the CRUD operations on the objects exposed by DI-API (Create, Read, Update and Delete) with one limitation:

Metadata functions are not exposed by DI Server. Using DI-API it is possible to change the structure of your data by adding, updating or removing user tables, user fields and user objects. These operations are not exported by DI Server.

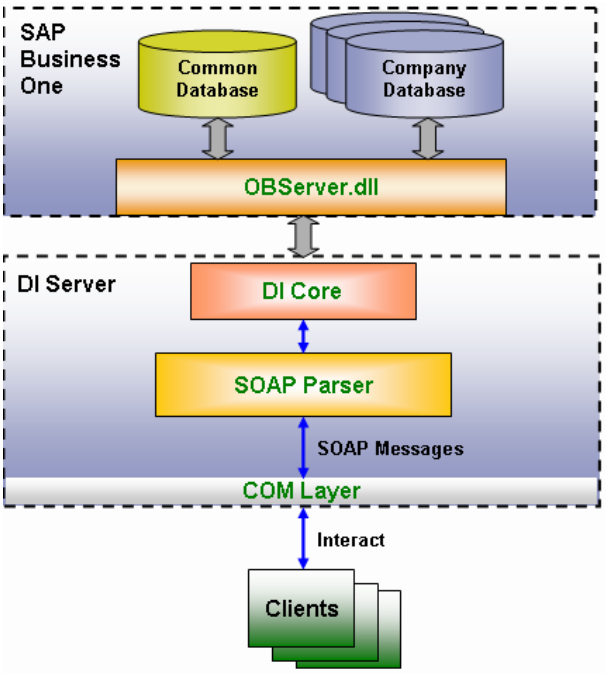
As the DI API the DI Server can group several operations in a single transaction, for doing that you have to use the Interact method. By using Interact you can group several operations in a unique transaction. By using BatchInteract each envelope operation will be executed as a single transaction.

DI Server is intended for high-volume data integration, where numerous client connections must be managed simultaneously and optimized for memory load and number of database connections. It is also suitable where Web-services architecture is preferred (with remote clients, for example) or in environments where a COM-based interface is not suitable (for instance integration with J2EE-based systems).

Note that as a rule-of-thumb, DI Server operations are slower than the corresponding DI-API operations and uses more network bandwidth: this is due to the fact that XML is string-based as opposed to COM that is a binary protocol.

DI Server supports a wider range of client technologies than the DI-API, and allows use of COM, CORBA, or TCP/IP to interface with SAP Business One using XML. It can be considered as a SOAP wrapper around the internal SAP Business One data services as opposed of DI-API as a COM-wrapper.

The same SAP Business One business logic (validation, default fields, and so on) applied in DI-API is applied for DI Server. The diagram below illustrates the architecture for the DI Server.



Integration scenarios

The different particular data integration cases can fall into a few high-level integration scenarios, depending on several dimension of the problem.

- Data replication: is there a subset of tables in the SAP Business One database that acts as a shared single master DB or an external database keeps a copy of these data?
- Data synchronization: if data are replicated, are they kept always in synch or are they synchronized only periodically?
- Which is the run-time environment required?

Business Scenario: Data replication not required

Data are not replicated in two databases but stored only in the SAP Business One database. The third party application is always online and simply uses one of the interfaces outlined to read and write data into the SAP Business One database – always passing through the business logic.

So the first choice to make is the list of data SAP Business One is the master for, resulting then in a set of Business One objects. The second point to understand is which different scenarios in the external application requires access in read and in write mode to these data and implement these accesses using the selected Business One objects via the interfaces defined above.

This is the case for instance in which all the marketing documents are stored in the SAP Business One database only – that acts them as the master repository for these objects. The external system accesses them via `Document` objects in the DI-API. DI-API or DI Server can be used here to move data from a third party application into SAP Business One depending on the product and run-time requirements.

As a rule-of-thumb:

- DI-API is recommended for client-based usage. If you are running in a Java environment deployed in a Windows run-time a possibility is to use JCO, a set of Java classes that wraps DI-API.
- DI Server is recommended for server-based usage i.e. cases where the integration point is the server.
- DI Server is the only option in run-times different than Windows.

Business Scenario: Data replication required

Data replication is needed when some data are replicated in two databases. The external system can be online or in some cases it can be offline and working with its local copy of the data. In this case then any change of data in one database should be reflected in the other database to ensure consistency to the overall integrated system.

- A change in the external database can be reflected in the SAP Business One database always using one of the two interfaces outlined above.
- In the opposite direction, a change in the SAP Business One database (for instance originated by an update via a SAP Business One client application) must be reflected in the external database.

The mechanisms to implement these 2 points depends whether the synchronization is intended to be at real-time or whether a periodic mutual batch synchronization is considered acceptable for the functionality of the integrated system.

Data always in synch

If the two systems must always be in synch there should be logic integrated with the system scenarios in order to access always live data and in order to update data back into the master DB.

- A change in the external system must be reflected in SAP Business One by using the DI interface, as described in the previous section
- A change in SAP Business One must be reflected back to the external database by

  adding the update logic to the `SBO_SP_TransactionNotification` (see here for more details: Transaction Notification). Every time a change is successfully executed via the business logic (i.e. by an operation from a SAP Business One client or a successful call to a SAP Business One object via a DI-API or DI Server call) the SAP Business One standard stored procedure is called with a set of parameters. By this method it is possible to detect the kind of transaction and the object involved and then it is possible to reflect this change back to the external system.

Data updated only periodically

Otherwise if it acceptable that the two systems are updated only periodically (or it is not possible to keep them always in synch as for instance in the case of Mobile Application) this logic can be kept in a set of offline tools that periodically export batch of data from the external system and import them in the SAP Business One system and vice-versa.

These tools can be deployed and activated in many ways, depending on the requirements of the system. They can explicitly activated by a menu in the external application or added to the SAP Business One desktop; or can be activated periodically in not-working hours by a silent service or a scheduled task; they can use files to exchange data or they can use a network connection – and so on so forth.

A first possibility is to use the ready-to-go functionality of SAP Business One. More information on how to use these options can be found in the help of the application or on SAP Service. Anyway under "Administration" / "Data Import/Export" you will find some of the possible options:

- .. / "Data Import" / "Import from Excel" can be used to import business partners or items. Any definition that is to be used in these records must already be there and matching.
- .. / "Data Import" / "Import Transactions from SAP Business One" can be used to import a subset of transaction types from another SAP Business One database.
- .. / "Data Export" / "Export Transactions to SAP Business One" can be used to export a subset of transaction types from a SAP Business One database.

In addition there are several options to export data through the menu bar ("File" / "Export to" / …), the tool bar (including export of results of queries).

Note that one possible option for this periodic update into the SAP Business One database is to use the SAP Business One Data Transfer Work Bench. The Data Transfer Workbench or DTW is an SAP created add-on with the sole purpose of integrating data into SAP Business One from external data sources. When installed, the DTW provides the user with a graphical interface able to map source data, from that source data pick the elements that are to be integrated and then map them to SAP Business One business objects, without any required code. Templates are provided in MS-Excel format to assist with setting up the external data to be integrated in a usable format as well as assisting the user with what fields are required to integrate into any given business object. Note that DTW uses internally the DI API and therefore its use is safe against inconsistencies.

On the other direction, it is possible to detect the changed objects in a SAP Business One database by a column in the corresponding SAP Business One table holding the last change date. As this is a read operation, accessing directly data from the database – for instance with an SQL statement – does not bring any inconsistency into the SAP Business One database itself.

Sample 1: Mobile Applications

Let's assume a simple case in which Business Partner details are kept on mobile devices and periodically saved in the SAP Business One master database. This is a sample of a solution in which data is replicated in several databases but the synchronization can happen only periodically.

A typical approach is to deploy a local mobile database on the mobile device and build a mobile application on it. Note that no SAP Business One software is loaded on the mobile:

- the local DB is not the SAP Business One database: replicating the SAP Business One table structure can be helpful but not required,
- the mobile application accesses this data locally with the native OS mechanisms being disconnected from the SAP Business logic unless natively recoded in the mobile environment,
- And the GUI uses only native OS widgets, possibly mimicking the SAP Business One look-and-feel with an SAP Business One skin.

The synchronization between the local copy of the data on the mobile device and their master copy in the SAP Business One database is taken care of by a server application. This application uses on one side a custom protocol based on the connectivity protocols provided by the mobile device and on the other side by the DI interface to access the master data. Based on these mechanisms:

- It reads the local data from the mobile device and tries to update the master database. This is when the business logic is enforced.
- It reads the last updated version of the data in the master database and updates the local copy of the data stored on the mobile device.

The first point can be achieved by exporting the data on the server in an Excel-compliant file and then use the DTW with loaded a custom template.

Sample 2: Integrate a vertical application with SAP Business One

Let's assume that a vertical application wants to integrate with SAP Business One leveraging on its ERP functionality and adding its own vertical added-value. For instance it provides some advanced and highly customized module for a vertical area in a well defined local geography but reuses all the standard and generic modules present in SAP Business One.

For simplicity sake's let's assume it reuses the Business Partner objects and that this data is not duplicated but kept only in the SAP Business One database. In this case any time there is a need in the vertical solution to access a the SAP Business One Business Partner object, the application will have to use the DI object `BusinessPartners` and its standard methods in order to read, browse, create, update and remove business partner data.

Let's make things a bit more complex and imagine that the external system maintains a local database of the Business Partners – say for performance reasons. Then:

- When a change to a Business Partner is required, this change needs to be performed twice: first on the local copy with proprietary mechanisms and second on the master copy utilizing the DI interface.
- When a Business Partner is changed by an operation from a SAP Business One client, this change needs to be reflected on the local copy too. The change logic is added into the `SBO_SP_TransactionNotification.` Note that this will capture also the changes coming from other operations carried out on Business Partners through the DI interface.

  Depending on the required run-time, there are several options to access DI objects
- If the application is a desktop application based on Windows, it can use the COM-based DI-API interface.
- If the application is a Java application always deployed on Windows machines, it can use the JCO set of classes wrapping the DI-API interface
- If the application is a web-based application based on J2EE-based technology (servlets, JSP, Struts, etc) and the options where the server can be deployed are not limited to Windows, it can use DI Server and access the DI functionality through a set of SOAP messages.

Related Contents

For more information, visit the Business One homepage.

Copyright