Integration of

ERP systems and ePages 6

with Web services

- Version 10/2011 -

Introduction

This document describes the basic principles and a possible procedure for integrating ePages and ERP systems.

The following requirements must be met to successfully integrate the systems:

- Knowledge of the programming and use of Web services
- Knowledge of the shop URL and the access data for shop administration
- Access via Web services must be released for the shop type (e.g. using the "ePages Flex" shop types or "ePages Enterprise" shop types).
- Knowledge of database queries from the ERP system
- Network connection between the ERP system and the ePages shop system

Useful information and experience in relation to ePages and Web services can also be found in the ePages Developer Forum: http://www.epages.com/forum/.

Basic principles

The following functional division is made when connecting the ePages shop system to an ERP system:

ERP system

Creation, editing and administering of data relating to products, customers and orders.

ePages system

Functions and procedures for the presentation and sale of products via online channels – order process, customer communication, customer registration, etc.

The ERP system is generally the leading system. Data is modified and processed (e.g. orders, customers) in the ERP system. In this case, the shop acts almost as simply a satellite. However, actions such as eBay and Amazon sales can be launched in the shop, if the ERP system does not offer these functions.

ePages is prepared for integration in ERP systems. The system offers completed APIs based on Web services.

All the necessary business objects such as products, customers, etc. are available. No development effort is required on the ePages side for standard data exchanges.

An overview of all available ePages Web services can be found here: http://www.epages.com/WebService/WebServices.html. The list shows all versions. The URN specifies how up to date a Web service is. All versions of a Web service can be used in all cases. The functionality of older versions is also ensured. New versions provide new functions but also include the functions offered by the previous versions.

The corresponding Web service clients on the ERP system side must be developed. The Web service client can be a service within the ERP system or a separate program "between" the ERP system and the shop. For ePages Web services, ePages delivers corresponding sample clients in the programming languages Java, .net and PERL. These examples should help you to develop your integration.

To collect new orders from the shop system, a spooler must be implemented on the ERP system side. The shop does not send new orders automatically to the ERP system. The ERP system must query these periodically (e.g. every hour).

Overview of data exchange

The following data is generally exchanged between the shop and the ERP system:

Products

- Creation and maintenance in the ERP system, export to ePages
- Export of warehouse stocks to ePages
- Export of customer-specific prices to ePages
- Export of images
- Export of product types (categorisation)

Categories

- Creation in ePages, export to the ERP system
- Creation in the ERP system, export to ePages
- Product assignment to categories in the ERP system

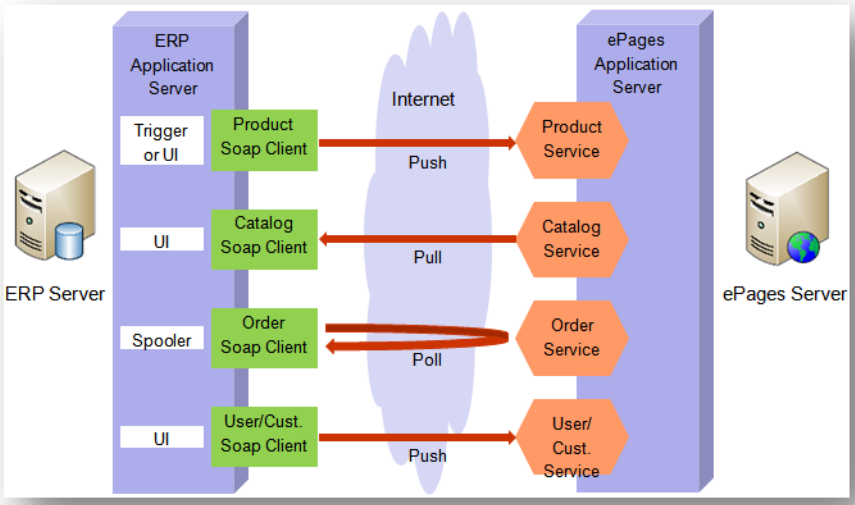
Customers

- Creation and administration of customers in the ERP system, export to ePages
- Creation of new customers in the ERP system from the order data

Orders

- Calling of new orders from ePages by the ERP system
- Administration and processing in the ERP system
- Export of order status changes to ePages

The graphic below shows the data exchange between the systems:



In principle, the data exchange is triggered by the ERP system.

Procedure

Practical experience shows that you should proceed as follows when connecting ePages to an ERP system:

1. Connection test
2. Tests with example data
3. Data mapping
4. Definition of desired functions
5. Implementation and testing

Some of these steps are explained in more detail in the following sections.

Connection test

The first step involves checking whether the connection has been set up between the two systems. This means that you must be able to access the shop system via the Web service.

The aim is therefore to establish and test the connection between the Web service client on the ERP system side and the ePages shop.

To do this, download the demo clients from this website:

ftp://epages-software.de/pub/products/epages6/current/WebServiceExamples/

Select a Web service, e.g.*PageCacheTestCase.java*.

Amend the connection parameters in *WebServiceConfiguration.java*:

```
// public final String WEBSERVICE_URL = getWebserviceURL();
...
public final static String WEBSERVICE_URL = http://[server_name]/epages/Store.soap";
public final static String WEBSERVICE_LOGIN = "/Shops/[shop_name]/Users/[user_name]";
public final static String WEBSERVICE_PASSWORD = "[user_password]";
...
```

Recompile the files and launch the Web service:

```
%JAVA% -cp bin;lib/* org.junit.runner.JUnitCore
de.epages.WebServices.PageCacheService.PageCacheTestCase
```

The output will show you whether the connection has been successfully established.

For the Web services, you can also create a separate administration access. This means that the Web services are disconnected from the shop administrator. If the access data for the administrator is changed, these changes do not have to be applied for the Web services.

It is recommended to use a proxy to create the connection. This makes it easier to log and evaluate the connection data as well as ensuring easier communication in support cases.

Tests with example data

Once the connection has been successfully established, you should test the Web service examples provided.

This test phase helps you to easily familiarise yourself with the available functions. It gives you an overview of how data can be transferred between the Web service and the shop system and what functions are available.

We recommend that you create the ePages demo shop on the shop system because the examples are tailored to this shop.

The example data for the Web service clients is stored in code as constant strings.

Data mapping

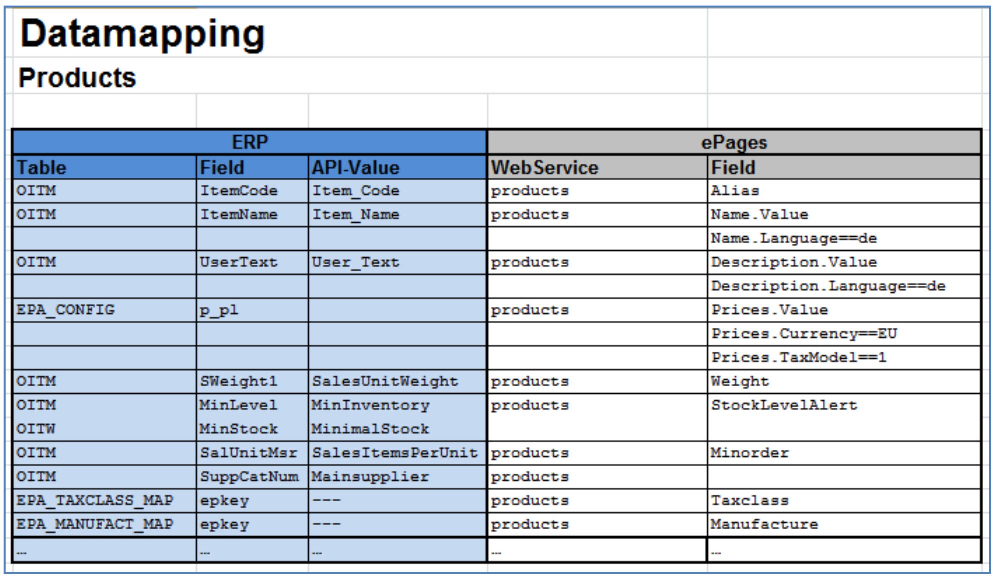
The examples describe data the exchange between the Web service client and ePages. The question of how the data is exchanged between the client and the ERP system database is open. The data model and data organisation in the ERP database are certainly different than in the ePages database.

A data mapping must therefore be carried out.

Which data is transferred to ePages, how and with what type, is described in the WSDL and XSD files for the Web services, see http://www.epages.com/WebService/WebServices.html.

The Web service client must read the data from the ERP system database, map it to the corresponding data structure defined in the WSDL and then transfer this data.

An example of the data mapping is shown in the graphic below:



Datamapping

Products

ERP			ePages	
Table	**Field**	**API-Value**	**WebService**	**Field**
OITM	ItemCode	Item_Code	products	Alias
OITM	ItemName	Item_Name	products	Name.Value
				Name.Language==de
OITM	UserText	User_Text	products	Description.Value
				Description.Language==de
EPA_CONFIG	p_pl		products	Prices.Value
				Prices.Currency==EU
				Prices.TaxModel==1
OITM	SWeight1	SalesUnitWeight	products	Weight
OITM	MinLevel	MinInventory	products	StockLevelAlert
OITW	MinStock	MinimalStock		
OITM	SalUnitMsr	SalesItemsPerUnit	products	Minorder
OITM	SuppCatNum	Mainsupplier	products	
EPA_TAXCLASS_MAP	epkey	---	products	Taxclass
EPA_MANUFACT_MAP	epkey	---	products	Manufacture
...	...	...	...	...

Definition of desired functions

On the basis of the requirements, we can now define which functions should be provided by the Web services. Practical experience has shown that the following basic functions are usually required:

Initial import to the shop system

The available basic data is transferred from the ERP system to the shop system.

Update in the case of data modifications

Data modifications must be reconciled between the systems. Since the ERP system is the leading system, the Web services must be able to send and collect data (push/pull).

Note: After changes have been made to content, the page cache must be reset to display the latest data. The "PageCacheService" Web service can be used for this.

Controlling the data exchange

The Web services must be able to transfer data at specified intervals and also on request.

You must also define the business objects for which data is exchanged. A comparison between the available ePages Web services and your requirements allows you to identify the new Web services that need to be implemented.

When defining and using or implementing the required functions, a phased procedure is recommended, e.g.:

- Phase 1: Transfer and update product and order data
- Phase 2: Transfer customer data
- Phase 3: Transfer warehouse and price information in real time

The real-time data exchange is above all performance critical, and in particular for the ERP system. For example, if customer-specific prices or warehouse stocks need to be calculated in the ERP system for a list display in the shop with multiple products for each individual product, this can result in long waiting times in the shop.

Data transfer example

Below is a practical example showing how the demo clients are used to establish a connection and create a product in the target shop.

Java versions of the clients are used in the example. The requirements for this are:

- Java knowledge
- Java environment to compile the classes
- Demo shop on the ePages target system

The tests can be carried out with a different shop. However, in this case, more changes than described will need to be carried out.

Connection test

The connection test is carried out as follows:

1. Download the package with the demo clients and unpack it e.g. to the directory *WS_DemoClients*:

   ftp://epages-software.de/pub/products/epages6/6.12/AddOns/WebServiceExamples.6.12.0/WS_DemoClients.zip

2. Amend the connection parameters:

   To do this, open the file
   WS_DemoClients\java\src\de\epages\WebServices\WebServiceConfiguration.java

   Amend the parameters as follows:

   ```
   // public final String WEBSERVICE_URL = getWebserviceURL();
   public final static String WEBSERVICE_URL =
   "http://whitelinux.epages.de/epages/Store.soap";
   ```

   The *whitelinux.epages.de* server is a public test system which can be used for the Web service tests.

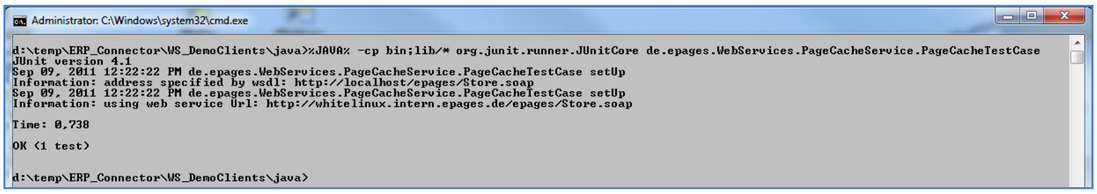
3. Recompile the file in the directory *WS_DemoClients\java*:

   ```
   %JAVAC% -cp bin;lib/* -d
   binsrc\de\epages\WebServices\WebServiceConfiguration.java
   ```

4. Test the connection

   ```
   %JAVA% -cp bin;lib/* org.junit.runner.JUnitCore
   de.epages.WebServices.PageCacheService.PageCacheTestCase
   ```

   The output for a successful test is as follows:



```
d:\temp\ERP_Connector\WS_DemoClients\java>%JAVA% -cp bin;lib/* org.junit.runner.JUnitCore de.epages.WebServices.PageCacheService.PageCacheTestCase
JUnit version 4.1
Sep 09, 2011 12:22:22 PM de.epages.WebServices.PageCacheService.PageCacheTestCase setUp
Information: address specified by wsdl: http://localhost/epages/Store.soap
Sep 09, 2011 12:22:22 PM de.epages.WebServices.PageCacheService.PageCacheTestCase setUp
Information: using web service Url: http://whitelinux.intern.epages.de/epages/Store.soap

Time: 0,738

OK (1 test)


d:\temp\ERP_Connector\WS_DemoClients\java>
```

Create product

To test the data transfer, create a product in the demo shop on the target system:

1. Amend the product service for the test:

   To do this, open the file
   WS_DemoClients\java\src\de\epages\WebServices\ProductService\ProductTestCase.java

2. Comment on the following lines in the bottom part of the file:

   ```
   //          testDelete();
   //          testExists(false);
   ```

   This is required to allow you to see the product in the shop as well. Otherwise, it is immediately deleted since deletion is part of the test case.

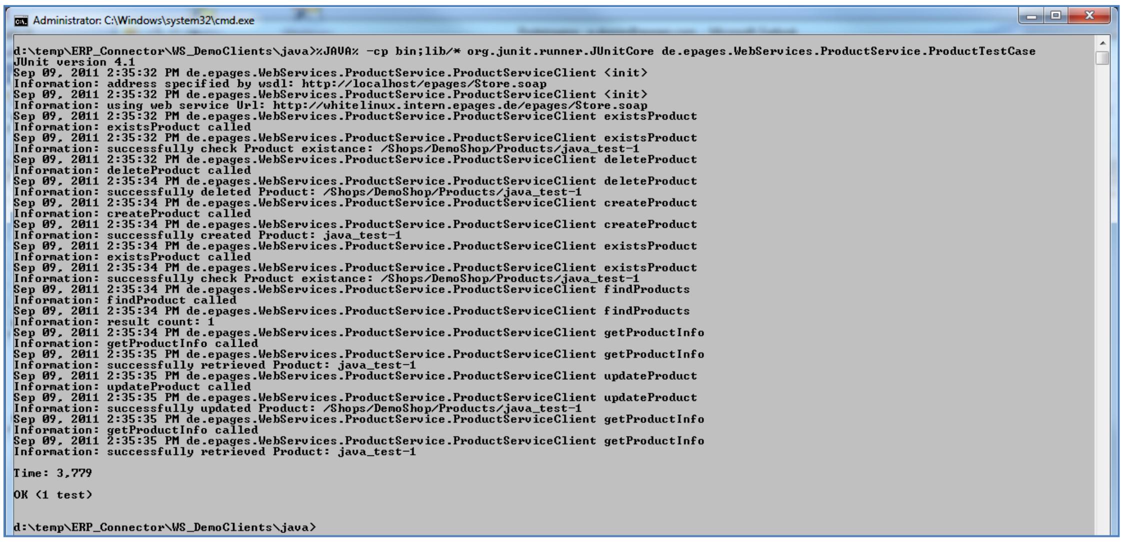
3. Compile the file in the directory *WS_DemoClients\java*:

   ```
   %JAVAC% -cp bin;lib/* -d bin
   src\de\epages\WebServices\ProductService\ProductTestCase.java
   ```

4. Start the test case:

   ```
   %JAVA% -cp bin;lib/* org.junit.runner.JUnitCore
   de.epages.WebServices.ProductService.ProductTestCase
   ```

5. Print the output on the command line:



```
d:\temp\ERP_Connector\WS_DemoClients\java>%JAVA% -cp bin;lib/* org.junit.runner.JUnitCore de.epages.WebServices.ProductService.ProductTestCase
JUnit version 4.1
Sep 09, 2011 2:35:32 PM de.epages.WebServices.ProductService.ProductServiceClient <init>
Information: address specified by wsdl: http://localhost/epages/Store.soap
Sep 09, 2011 2:35:32 PM de.epages.WebServices.ProductService.ProductServiceClient <init>
Information: using web service Url: http://whitelinux.intern.epages.de/epages/Store.soap
Sep 09, 2011 2:35:32 PM de.epages.WebServices.ProductService.ProductServiceClient existsProduct
Information: existsProduct called
Sep 09, 2011 2:35:32 PM de.epages.WebServices.ProductService.ProductServiceClient existsProduct
Information: successfully check Product existance: /Shops/DemoShop/Products/java_test-1
Sep 09, 2011 2:35:32 PM de.epages.WebServices.ProductService.ProductServiceClient deleteProduct
Information: deleteProduct called
Sep 09, 2011 2:35:34 PM de.epages.WebServices.ProductService.ProductServiceClient deleteProduct
Information: successfully deleted Product: /Shops/DemoShop/Products/java_test-1
Sep 09, 2011 2:35:34 PM de.epages.WebServices.ProductService.ProductServiceClient createProduct
Information: createProduct called
Sep 09, 2011 2:35:34 PM de.epages.WebServices.ProductService.ProductServiceClient createProduct
Information: successfully created Product: java_test-1
Sep 09, 2011 2:35:34 PM de.epages.WebServices.ProductService.ProductServiceClient existsProduct
Information: existsProduct called
Sep 09, 2011 2:35:34 PM de.epages.WebServices.ProductService.ProductServiceClient existsProduct
Information: successfully check Product existance: /Shops/DemoShop/Products/java_test-1
Sep 09, 2011 2:35:34 PM de.epages.WebServices.ProductService.ProductServiceClient findProducts
Information: findProduct called
Sep 09, 2011 2:35:34 PM de.epages.WebServices.ProductService.ProductServiceClient findProducts
Information: result count: 1
Sep 09, 2011 2:35:34 PM de.epages.WebServices.ProductService.ProductServiceClient getProductInfo
Information: getProductInfo called
Sep 09, 2011 2:35:35 PM de.epages.WebServices.ProductService.ProductServiceClient getProductInfo
Information: successfully retrieved Product: java_test-1
Sep 09, 2011 2:35:35 PM de.epages.WebServices.ProductService.ProductServiceClient updateProduct
Information: updateProduct called
Sep 09, 2011 2:35:35 PM de.epages.WebServices.ProductService.ProductServiceClient updateProduct
Information: successfully updated Product: /Shops/DemoShop/Products/java_test-1
Sep 09, 2011 2:35:35 PM de.epages.WebServices.ProductService.ProductServiceClient getProductInfo
Information: getProductInfo called
Sep 09, 2011 2:35:35 PM de.epages.WebServices.ProductService.ProductServiceClient getProductInfo
Information: successfully retrieved Product: java_test-1

Time: 3,779

OK (1 test)


d:\temp\ERP_Connector\WS_DemoClients\java>
```

6. Check in the demo shop to make sure that the product *java_test-1* is displayed in the product list.

In case of any problems, send your queries to: support@epages.com.