2014

dk Hugbúnaður ehf.

Þorvaldur Hafdal Jónsson

# [ITEMS WEBSERVICE]

In this document we will go over many of the functionality of the  dkWSItemsCgi API has to offer.

## Overview

### WebService

Dk offers a webservice interface that other software can interact with to send receive data from the dkSystem. The interface is a standard CGI that uses SOAP (Simple Object Access Protocol)
http://www.w3schools.com/soap/default.asp

Since the service is a SOAP service almost every programing language can use the interface

as I write this I know of the following languages using the service PHP,ASP, ASP.NET, C#, VB#,Ruby

### System modules

We offer many of the dksystem modules in the interface today and there are always new features being added,
These modules are now in the service
Items(Products),Invoices,Quotes,Subscriptions,Customers,Vendors,Members,Projects

### Test service

To connect to a test service you can use the following information.

**Service URL**
**http://saga.dk.is/dkwsitemscgi.exe**

**WSDL**
**http://saga.dk.is/dkwsitemscgi.exe/wsdl/IItemService**

**SOAP**
**http://saga.dk.is/dkwsitemscgi.exe/soap/IItemService**

**[SOAP Header]**
**Username:** ws.test
**Password:** test

## SOAP Header

There are two ways to create a soap header

1. You can use WSE 2.0 to authenticate with the service.
The header will look something like this

```
<soap:Header>
        <wsse:Security soap:mustUnderstand="1">
                <wsse:UsernameToken xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
                wssecurity-utility-1.0.xsd" wsu:Id="SecurityToken-24ada6f8-4626-4269-b786-a22361bfde78">
                        <wsse:Username>ws.test</wsse:Username>
                        <wsse:Password Type="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
                username-token-profile-1.0#PasswordText">test</wsse:Password>
                </wsse:UsernameToken>
        </wsse:Security>
</soap:Header>
```

2. The service offers the use of a SoapHeader caller TBasicSecurity and you can use that is you are unable to use WSE 2.0

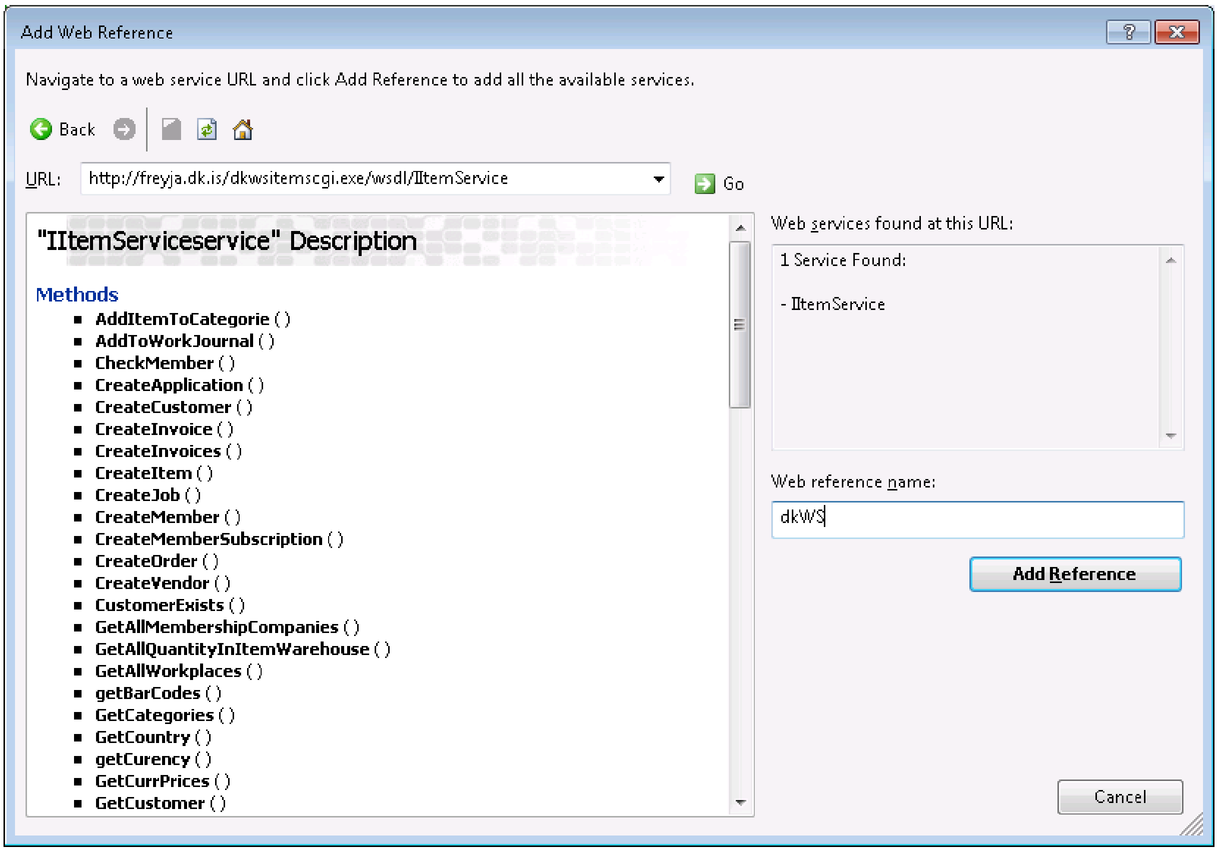
```
<soap:Header>
        <q1:BasicSecurity id="h_id1" xmlns:q1="urn:dkWSValueObjects">
                <Username xsi:type="xsd:string">ws.test</Username>
                <Password xsi:type="xsd:string">test</Password>
        </q1:BasicSecurity>
</soap:Header>
```

## Example usage

In these examples visual studio is used and the C# language

### Add a Web Reference to service

There are many programming languages and development tool available but in the examples I will use Visual Studio in the examples and reference.
Open up Visual Studio and open/create a project.
Click on Website and then on Add Web reference
enter the URL to the WDSL and enter a name for the service for example dkWS


Add Web Reference

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

URL: http://freyja.dk.is/dkwsitemscgi.exe/wsdl/IItemService

"IItemServiceservice" Description

Methods

- AddItemToCategorie ( )
- AddToWorkJournal ( )
- CheckMember ( )
- CreateApplication ( )
- CreateCustomer ( )
- CreateInvoice ( )
- CreateInvoices ( )
- CreateItem ( )
- CreateJob ( )
- CreateMember ( )
- CreateMemberSubscription ( )
- CreateOrder ( )
- CreateVendor ( )
- CustomerExists ( )
- GetAllMembershipCompanies ( )
- GetAllQuantityInItemWarehouse ( )
- GetAllWorkplaces ( )
- getBarCodes ( )
- GetCategories ( )
- GetCountry ( )
- getCurency ( )
- GetCurrPrices ( )
- GetCustomer ( )

Web services found at this URL:

1 Service Found:

- IItemService

Web reference name: dkWS

In the clause „USES“add then name of the service you added in before

```
using dkWS;
```

## Fetching departments

```
//Create an instance of the service
IItemServiceservice ws = new IItemServiceservice();
//Example of working with the url from settings
string wsURL = (string)Settings["dkwsurl"];
if (wsURL[wsURL.Length-1].ToString() != "/")
{
   wsURL = wsURL + "/";
}
//Add the URL and the service path
ws.Url = wsURL + "dkwsitemscgi.exe/soap/IItemService";
//Add soap header for authenticating
ws.BasicSecurityValue = new BasicSecurity();
ws.BasicSecurityValue.Username = (string)Settings["dkwsusername"];
ws.BasicSecurityValue.Password = (string)Settings["dkwspassword"];
//Fetch all departments created/changed after1.1.1900
ws.GetDim1(new DateTime(1900,1,1));
```

A good practice is to create a class that inherits `IItemServiceservice` that sets the `URL,Username,Password.`

## Example of Item/Product sync

```
IItemServiceservice ws = new IItemServiceservice();
ws.BasicSecurityValue = new BasicSecurity();
ws.BasicSecurityValue.Username = "ws.test";
ws.BasicSecurityValue.Password = "test";
TItemOptions itmOpt = new TItemOptions();
itmOpt.IncludeBarCodes = true;
DateTime dtt = new DateTime(2011, 8, 1);
List<TItem> items = new List<TItem>();
TItem[] tmpItems;
int estItmCount = ws.NumberOfModifiedItems(dtt);
if (estItmCount <= 500)
{
       foreach (TItem itm in ws.GetItems(dtt, itmOpt))
       {
              items.Add(itm);
       }
}
else
{
       int itmCount = 1;
       int nextRecId = 0;
       while (itmCount > 0)
       {
              tmpItems = ws.GetItemsEx(nextRecId, 500, itmOpt);
              foreach (TItem itm in tmpItems)
              {
                     items.Add(itm);
              }
              itmCount = tmpItems.Length;
              if (tmpItems.Length > 0)
              {
                     nextRecId = (tmpItems[tmpItems.Length - 1].RecordId + 1);
              }
       }
}
//Now do something with the items received
```

## Example Create Invoice

```
dkWS.IItemServiceservice serv = new dkWS.IItemServiceservice();
serv.SecurityValue = new dkWS.Security();
serv.SecurityValue.UsernameToken = new dkWS.TUsernameToken();
serv.SecurityValue.UsernameToken.Username = "ws.test";
serv.SecurityValue.UsernameToken.Password = "test";
dkWS.TInvoice inv = new dkWS.TInvoice();
inv.CustomerNumber = "abcd";
inv.SalePerson = "WEB";

List<dkWS.TInvoiceLine> lines = new List<dkWS.TInvoiceLine>();

dkWS.TInvoiceLine lin = new dkWS.TInvoiceLine();
lin.ItemCode = "000001";
lin.Quantity = 4;
```

```
lines.Add(lin);
//No UnitPrice is set so prices from dk will be used
dkWS.TInvoiceLine lin2 = new dkWS.TInvoiceLine();
lin2.ItemCode = "000002";
lin2.Quantity = 4;
lin2.UnitPrice = 200;
lin2.DiscountAmount = 50;
lines.Add(lin2);
//Price will be set as 190 and discount 50 for the whole invoice line so the total
amount of the line will be 750

inv.Lines = lines.ToArray();
serv.CreateInvoice(inv, false); //Send the invoice to the webservice and create
```

## Classes

### TCustomer
Customer, has many of the properties from the customer in the dkSystem

### TCustomerOptions
This class is used to define what extra information should be included in the response from the web service for example Item receivers, Contacts, Memos

### TItem
Item/Product: The Item contains many of the properties from the item in the dkSystem

### TJob
Job/Project: Information about the Job in the dkSystem

### TVendor
Vendor/supplier, Information about the vendor in the dkSystem

### TInvoice
Invoice, Object contains the Head of an invoice

### TInvoiceLine
Invoice line, is the line of an invoice that has an item defined quantity and so on.

### TOrder
Order, this is Object contains the head of an sales order

### TOrderLine
Order Line, contains the basic information for a line belonging to an order

### TMember
Member contains many attributes of a member from the dkSystem

### TmemberOptions
This class is used to define option and control information flow for the TMember Object.

## System Modules

### Customers

The service offers the ability to create and update a customer.

#### GetCustomer

This queries a customer from the dkSystem and returns all information about the customer and any additional information defined in the TCustomerOptions class.

*Params:*

1. Customer Number (Unique ID)
2. TCustomerOptions,  This class defines what extra information should be returned with the Tcustomer and how the request should be handled.(Contacts, Itemrecivers and more)

#### GetCustomers

Get all customers that have change from the date specified as a parameter in the function and return an array of Tcustomer.

*Params:*

1. Modified date from query
2. TCustomerOptions,  This class defines what extra information should be returned with the Tcustomer and how the request should be handled.(Contacts , Itemrecivers and more)

#### GetCustomersEx

This function is very useful to do a full/initial sync of customers in chunks with Record ID as a position in the dkSystem and number of objects to return.

*Params:*

1. Record ID
2. Number of objects
3. TCustomerOptions,  This class defines what extra information should be returned with the Tcustomer and how the request should be handled.(Contacts ,Itemrecivers and more)

#### NumberOfModifiedCustomers

This function returns the number of changed customers from the date/time specified.

*Params:*

1. Date Changes, modified  :  the minimum date time that should be used to return

*Return:*

- [INT] Number of customers changed in the table.

## GetCustomerFromPhoneNumber

Returns a customer from phone number.

*Params:*

1. Phone number to look for.

## UpdateCustomer

This function updates a customer in the dkSystem.

*Params:*

1. TCustomer

**NOTE:** When updating a customer have in mind if working with production data, you are overriding in the dkSystem and if you define an empty string in let say Address3 that will become an empty string in the dkSystem.

## CreateCustomer

This function created a customer in the dkSystem

*Params:*

1. TCustomer

## CustomerExists

This function checks if a customer exists in the dkSystem and return true if the customer is found else if return false.

*Params:*

1. Customer number

## GetCustomerPaymentTerms

This function return all payment term from the dkSystem

*Params:*

*NONE*

## GetCustomerPaymentModes

This function returns all payment modes from the dkSystem

*Params:*

*NONE*

## Inventory/Products

### GetItem

This function returns one product from the dkSystem.
The option class defines what extra information to return with the product for example barcodes, warehouse information

*Params:*

1. Item Code(Unique)
2. TItemOptions

### GetItemFromRecordID

This function returns one product from the dkSystem.
The option class defines what extra information to return with the product for example barcodes, warehouse information

*Params:*

1. RecordID
2. TItemoptions

### GetItems

This function returns all products from the dkSystem that have changed from a specified date/time.
The option class defines what extra information to return with the product for example barcodes, warehouse information

*Params:*

1. Modified Date/Time
2. TItemOptions

### GetItemsEx

This function is very useful to do a full/initial sync of products in chunks with Record ID as a position in the dkSystem and number of objects to return.

*Params:*

1. Record ID
2. Number of objects

This function is very useful if you need to do full/initial sync and return in chunks.

## NumberOfModifiedItems
This function returns the number of changed Items from the date/time specified.

*Params:*

1. Date Changes, modified : the minimum date time that should be used to return

*Return:*

- [INT] Number of Items changed in the table.

## CreateItem
This function creates a product in the dkSystem.

*Params:*

1. TItem

## UpdateItem
This function updates an existing item with the values supplied in the item class.

*Params:*

1. TItem

## GetBarCodes
This function returns barcodes changed after the date/time supplied.

*Params:*

1. Modified date/time

## GetItemCategories
This function returns categories for one product.

*Params:*

1. Item Code

## GetCategories

This function returns one or all categories.
If Param1 = 0 all categories are returned.

*Params:*

1. Record ID

## AddItemToCategorie

This action attaches the product to a category.

*Params:*

1. Titem
2. TCategory

## RemoveItemFromCategorie

This action detaches the product from a category.

*Params:*

1. Titem
2. TCategory

## GetItemVendors

This function returns all vendors for a product.

*Params:*

1. ItemCode
2. Boolean : OnlyWebshopItem

## GetCustomerItemDiscounts

This function takes a TInvoice object and calculates the best prices/Discounts for a customer with rules in the dkSystem.

*Params:*

1. TInvoice

## GetQuantityInItemWarehouse
This function returns item quantity in warehouse.

*Params:*

1. ItemCode
2. warehouse

## GetAllQuantityInItemWarehouse
This function returns all items/products belonging to a specific warehouse.

*Params:*

1. Modified Created date of the Item/product
2. Warehouse
3. Only items in webshop

## GetItemGroups
This function returns item/product groups.

*Params:*

- Date changed modified

Using 1.1.1900 will return all groups

## GetItemsByGroup
This function returns all items/products belonging to a specified group.

*Params:*

- Group ID

## GetItemsByGroupRecordID
This function returns all items/product belonging to a specific group.

*Params:*

- Group Record ID

## Invoice System

### CreateInvoice

This function creates an invoice in the dkSystem.
if the parameter doPost is set to true the invoice gets an invoice number and cannot be modified since it is now a legal invoice, if doPost is false the invoice is created but not printed and booked and can there for be changed.

*Params:*

- TInvoice
- doPost

### GetInvoice

This function returns one invoice from the dkSystem.

*Params:*

- Customer number
- Invoice number

### GetInvoices

This function return X invoices for a customer.

*Params:*

- Customer number
- Number of invoices to return

### GetInvoicesFromClaims

This function return invoices belonging to claimer.

*Params:*

- ArrayOfInvoiceClaims (Array of claims)

### CreateOrder

This function creates a quote/Order in the dkSystem.

*Params:*

- TOrder

## GetSalesPerson

This returns a salesperson.

*Params:*

- SalesPersonID

## GetSalesPersons

This function returns all salespersons.

*Params:*

1. Date changed, modified

## SalesPersonsExists

This function checks if a salesperson exists in the dkSystem Database

*Params:*

- SalepersonID

## getPaymentTypes

This function returns all payment types.

*Params:*

NONE

## GetCurency

This function returns all currency and rates for one or more currency´s.
If Param1 is empty then all currencies are returned

*Params:*

- CurrencyID

### GetCountry

This function returns countries.

If Param1 is empty then every country is returned

*Params:*

- Country ID

### GetCurrPrices

This function returns foreign prices for an item/Product

*Params:*

- ItemID

### GetEmployees

This function returns employees.

*Params:*

1. Date changed, modified

## Vendor System

### GetVendor

This function returns a vendor from the dkSystem.

*Params:*

1. Vendor ID

### GetVendors

This function returns vendors from the dkSystem.

*Params:*

1. Date Changed, Modified

### UpdateVendor

This function updates the vendor in the dkSystem with the values supplied in Param1.

*Params:*

1. TVendor

### CreateVendor

This function creates a vendor with the values supplied in param1.

*Params:*

1. TVendor

### VendorExists

This function checks if a specified vendor exists and return true if found.

*Params:*

1. Vendor ID

## Member System

### GetMember

This function returns a member.

*Params:*

1. MemberID
2. TMemberOptions

### GetMembers

This function returns members.

*Params:*

1. Date changed,modified
2. TMemberOptions

## CreateMember

This function created a member with the values supplied in param1.

*Params:*

1. TMemberV2

## UpdateMember

This function updates a member with the values supplied in param1.

*Params:*

1. TMemberV2

## UpdateMembers

This function takes in many members and updates the corresponding member in the dkSystem.

*Params:*

1. ArrayOfMembersV2

## GetMembersEx

This function returns members from the dkSystem in chunks.
Very useful function for fist sync and also if lots of changes have been made

*Params:*

1. MemberRecordID
2. NumberOfItems
3. TMemberOptions

## GetModifiedMembers

This function return members that have been created changed from a certain point in time.

*Params:*

1. Date Changed, Modified
2. TMemberOptions

### NumberOfModifiedMembers
This function returns how many members have changed.

*Params:*

1. Date Changed, modified

### MemberExists
This function checks if a member exists.

*Params:*

1. MemberID

### isMemberInGroup
This function checks if a member is a member of a group.

*Params:*

1. MemberID
2. GroupID
3. SubGroupID

### GetMemberGroups
This function returns groups.

*Params:*

1. Date changed, modified

### GetMemberSubGroups
This function returns subgroups.

*Params:*

1. Date changed, modified

### isGovermentEmployee
This function checks if the member is a government employee
This function is ONLY valid in Iceland

*Params:*

1. MemberID

### HasAnyMemberSubscription

This function check if a member has a subscription.

*Params:*

1. MemberID

### GetMemberSubscriptionGroupTypes

This function return subscription types.

*Params:*

NONE

### GetMemberSubscriptionGroups

This function returns subscription groups from a type.

*Params:*

1. TypeID

### CreateMemberSubscription

This function creates a subscription for a member.

*Params:*

1. TMemberSubscription

### GetMemberSubscription

This function returns a subscription for a member.

*Params:*

1. MemberID
2. TypeID
3. GroupID

## GetMemberSubscriptionByPassword

This function returns a subscription from a password.

*Params:*

1. TypeID
2. Password

## ValidMemberSubscription

This function validates if a subscription is active from a password.

*Params:*

1. TypeID
2. Password

## GetAllMembershipCompanies

This function returns membership companies, funds and grants that have been configured in the member system.

*Params:*

1. GetFunds : Defines if funds should be returned.
2. GetGrants: Defines if grants should be returned.

*Return:*

- ArrayOfMembershipCompanies: Array that contains all membership companies. Each membership company can contain many funds and grants.

## GetMembershipData

This function returns all membership companies that the member is associated to along with grants and funds.

*Params:*

1. MemberNumber :The unique ID of the member

*Return:*

- ArrayOfMembershipCompanies: Array that contains all membership companies. Each membership company can contain many funds and also grants.

### CheckMember
This function checks if a member is associated to a specific membership company and if the member is active.

*Params:*

1. MemberNumber :Unique ID of the member
2. MembershipCompany: Membership company ID

### CheckMember2
This function check if a specified member is active.

*Params:*

1. MemberNumber : Unique ID of the member

### CreateApplication
Creates a grant application for member and a specific grant in a specific fund.

*Params:*

1. Application : Grant Application

*Return:*

- TApplicationResult: contains the number of the application and the message to be displayed to the user.

### SendApplicationAttachment

Upload a file/attachment for the grant application.

*Params:*

- fileData : File content
- fileName: Name of file
- memberNumber: Unique ID of the member that the application belongs to.
- applicationNumber: Number of the application that the file should be attached to.

### GetApplicationAttachmentInfo
Get information about attachments that are assigned to an application.

*Params:*

1. fundCode : Number of fund that belongs to the application.
2. applicationNumber: Number of the application

*Returns:*

ArrayOfAttachment:  Array of attachment information that belong to the application.

## GetElections

Upload a file/attachment for the grant application.

*Params:*

- VoterType: Type of voter ( voterType for member is 11 )
- Number: Voter (member) number

*Returns:*

- ArrayOfElectionBallots: array of TElectionBallot items //Member can be on many ballots
    - RecordId : The ballot id
    - Title: The ballot title
    - Footer : The ballot footer
    - DateFrom : Starting date for the ballot
    - DateTo : End date for the ballot
    - Color : Color for the ballot in the HTML format of “#RRGGBB”
    - Terms: Terms text for the ballot
    - Instructions: Instructions for the ballot
    - BallotType : (0 = election, 1 = survey)
    - HasVoted: Has voter voted in this ballot
    - VotingDate: When did voter vote in this ballot
    - ElectionHeads : ArrayofElectionHeads, array of TElectionHead// Each ballot can have many questions
        - RecordId : Question ID
        - BallotId : The ballot id
        - Title: The question title
        - AnswerType: (0 = Boolean value, 1=float value, 2=string value)
        - Required: Is it required to answer this question
        - Multiselect: Can the voter select many answers for this question
        - MultiselectMinCount: Mininum answer count
        - MultiselectMaxCount: Maximum answer count
        - HasBlankVote: Can the voter return a blank vote for this question
        - Footer: Footer text for the question
        - Terms: Terms text for the question
        - Instructions: Instructions for the question
        - ElectionLines : ArrayofElectionLines, array of TElectionLine //Each question can have many defined answers
            - HeadId: The question ID
            - SeqNum: The answer ID

- Answer: The answer string value

### ReturnVotes

Sends selected votes for voter to the server

*Params:*

1. VoterNumber: Voter (member) number
2. VoterType: Type of voter ( voterType for member is 11 )
3. BallotId: The number of the ballot the returned votes belong to
4. arrVotes: array of votes for the questions on the selected ballot
    - ArrayOfElectionAnswers: array of TElectionAnswer
        - HeadId: The question ID
        - SeqNum: The answer ID
        - AnswerType : (0 = Boolean value, 1=float value, 2=string value)
        - StringAnswer**:** If answer is of type String (2) then the appropriate answer is returned in StringAnswer
        - BoolAnswer: If Answer is of type Boolean(0) then the appropriate answer is returned in BoolAnswer
        - FloatAnswer**:** If Answer is of type Float(1) then the appropriate answer is returned in FloatAnswer
        - IsBlankAnswer: If voter returns a blank vote, then this value should be true Special behaviour: If IsBlankAnswer is returned SeqNum must be Zero(0) else it is not registered (since BlankAnswer has no SeqNum). HeadId is returned as per usual.

*Returns:*

- TWSResult:
    - ResultCode : 0 = error, 1 = success
    - ErrorString: ErrorMessage

## Project/Job System

### GetJob

This function returns one job.

*Params:*

1. JobID

## GetJobs
This function returns jobs.

*Params:*

1. Date Changes, modified

## CreateJob
This function creates a job in the dkSystem.

*Params:*

1. TJob

## UpdateJob
This function updates a job in the dkSystem with the values supplied in Param1.

*Params:*

1. TJob

## AddToWorkJournal
This function creates or adds to an existing workjournal.

*Params:*

1. TworkJournalLine
2. Workjournal start date
3. Workjournal end date

## GetWorkItems
This function returns all matching the options supplied.

*Params:*

1. TWorkOptions

## AddWorkItem

This function adds a work item to a Work Journal.

*Params:*

1. TWorkLine
2. Workjournal start date
3. Workjournal end date

## AddWorkItems

This function adds work items to a Work Journal.

*Params:*

1. Array of TWorkLine
2. Workjournal start date
3. Workjournal end date

## UpdateWorkItem

This function updates a work item in a Work Journal.

*Params:*

1. TWorkLine

## UpdateWorkItems

This function updates work items in a Work Journal.

*Params:*

1. Array of TWorkLine

## DeleteWorkItem

This function deletes a work item from a Work Journal.

*Params:*

1. TWorkLine

## DeleteWorkItems

This function deletes work items from a Work Journal.

*Params:*

1. Array of TWorkLine

## GetWorkCostItems

This function returns all Cost Items matching the options supplied.

*Params:*

1. TWorkOptions

## AddWorkCostItem

This function adds a work cost item to a Work Journal.

*Params:*

1. TWorkCostLine
2. Workjournal start date
3. Workjournal end date

## AddWorkCostItems

This function adds work cost items to a Work Journal.

*Params:*

1. Array of TWorkCostLine
2. Workjournal start date
3. Workjournal end date

## UpdateWorkCostItem

This function updates a work cost item in a Work Journal.

*Params:*

1. TWorkCostLine

## UpdateWorkCostItems

This function updates work cost items in a Work Journal.

*Params:*

1. Array of TWorkCostLine

## DeleteWorkCostItem

This function deletes a work cost item from a Work Journal.

*Params:*

1. TWorkCostLine

## DeleteWorkCostItems

This function deletes work cost items from a Work Journal.

*Params:*

1. Array of TWorkCostLine

## Miscellaneous

### GetRecords

This function is used to extract records from any table in dk that have changed after a specified Date Time.

*Params:*

1. Date Changes, modified : the minimum date time that should be used to return
2. TableName : Name of the table that should be used
3. Options : Use options to specify what should be included in the result like field names

*Return:*

- ArrayOfRecord: Array of record that contains all fields that have been specified in the options.

### GetRecordEx

This function is very useful to do a full/initial sync of any table in chunks with Record ID as a position in the dkSystem and number of objects to return.

*Params:*

1. Record ID
2. Number of objects
3. TRecordOptions, This class defines what extra information should be returned with the TRecord Array

### GetRecordCount

This function returns the count of records that have changed from the specified table from the date/time specified.

*Params:*

2. TableName : The name of the dk table
3. Date Changes, modified : the minimum date time that should be used to return

*Return:*

- [INT] Number of records changed in the table.

## GetAttachment

This function returns the content of an attach file in dk.

*Params:*

1. AttachmentID : The unique ID of the attachment to be fetched

*Return:*

- [Byte array] the content of the file in binary format.

## GetAttachmentImage

This function returns the content of an image attached to dk in the defined size.

*Params:*

1. ImgaeID : The uniqe ID of the image to be fetched
2. MaxWidth : the maximum resize width
3. MaxHeight : the maximum resize height
4. Format : the format of the image to return

*Return:*

- [Byte array] the content of the image in binary format.

WORD of advice

Like with any API you need to be aware that you are connecting to another system a telling that system what to do.
This has it´s benefits but it also has it´s down side.
for example if all values are not stored in the system connecting to the web service or objects are not being accrued from the web service it is possible to override all values in the dkSystem

An example of this would be customer sync

```
Tcustomer cust = new TCutomer.Create();
cust.Number = „5885522“
cust.Name = „This is funny“
cust.Address1 = „somewhere“
ws.UpdateCustomer(cust)
```

If the customer cust.Number he has been updated with all the values from the cust object
the Tcustomer object contains many more properties than number,name.address1 and when the Tcustomer object was created all properties got default values.

The more correct way would to get the customer from the webservice and update only the values that you need to change

```
Tcustomer  Cust = Ws.getCustomer(cust,Null)
cust.Name = „This is funny“;
cust.Address1 = „somewhere“;
ws.UpdateCustomer(cust);
```

Now the only values that have changed are the name and the address in the dkSystem