



**DirectSmile Online Interfaces
Documentation
March 2009**

BEFORE WE BEGIN	3
THE DIRECTSMILE STREAMIMAGE INTERFACE:	4
STREAMING IMAGE PARAMETERS:	5
PARAMETERS REQUIRED FOR AUTHENTICATION... ..	5
STANDARD PARAMETERS FOR SET RENDERING... ..	6
SPECIAL PARAMETERS FOR ADVANCED IMAGE PROCESSING... ..	7
TREATMENT OF SPECIAL CHARACTERS IN URLS:	10
USING THE STREAM IMAGE INTERFACE DYNAMICALLY	12
THE DIRECTSMILE STREAMDOC INTERFACE:	14
STREAMING DOCUMENT PARAMETERS:	15
PARAMETERS REQUIRED FOR AUTHENTICATION... ..	15
STANDARD PARAMETERS FOR DOCUMENT RENDERING... ..	16
SPECIAL PARAMETERS FOR ADVANCED DOCUMENT PROCESSING.....	17
JAVASCRIPT AND THE DIRECTSMILE WEBSERVICES:	20
WHAT WE NEED TO GET STARTED	20
CREATING DOCUMENTS USING THE DIRECTSMILE WEB SERVICES AND JAVA SCRIPT	24
THE DocSTATUS RETURN VALUE	26
PHP AND THE DIRECTSMILE WEBSERVICES:	29
CREATING DOCUMENTS USING THE DIRECTSMILE WEB SERVICES AND PHP5.....	30
THE DocSTATUS RETURN VALUE	33
THE MSGCOMXML	35
THE MSGCOMMAND OBJECT STRUCTURE	36
MY FIRST MSGCOMXML – AN EASY EXAMPLE	36
DATA STORAGE STRUCTURES.....	37
<PARAMETERS>.....	39
SUPPORTED MSGCOMXML PARAMETERS IN DIRECTSMILE ONLINE	39
PARAMETERS TO CONTROL DOCUMENT RENDERING... ..	39
<FANDV FIELDNAME="OUTPUT" FIELDVALUE="PDFPREVIEW"/>.....	39
<FANDV FIELDNAME="JPGCOMPRESSION" FIELDVALUE="80"/>	40
<FANDV FIELDNAME="CONVERTTO" FIELDVALUE="1"/>.....	40
<FANDV FIELDNAME="OUTPUTPAGEPIXWIDTH" FIELDVALUE="600"/>.....	40
<FANDV FIELDNAME="PAGERANGE" FIELDVALUE="1-10"/>	40
<FANDV FIELDNAME="IMPPREVIEW" FIELDVALUE="1"/>.....	40
<FANDV FIELDNAME="SUPPRESSETIMAGECROPPING" FIELDVALUE="1"/>.....	40
<FANDV FIELDNAME="WATERMARK" FIELDVALUE="SYS_IMAGE"/>	41
<STATIC FIELDS>	42
<TABLEROWS>	43
THE DIRECTSMILE WEB SERVICE IN MICROSOFT .NET APPLICATIONS AND ACTIVE SERVER PAGES	44
APPENDIX A THE DIRECTSMILE WEBSERVICE FUNCTIONS IN DETAIL:	49

Before we begin,

This Document is intended, to show you how to use the interfaces of your DirectSmile Online, to implement personalised images and documents on your Website.

It is build up as a step by step guide for Developers and Site builders. But no matter whether you are an experienced Web Developer or a Beginner, you will find useful Information in this guide to make you familiar with the DirectSmile Online Interfaces and their functionality.

DirectSmile Image personalisation can be divided in two major Parts, Images and Documents. Images are single Pictures, including personalisation while Documents can contain all sorts of personalisation like variable text, multiple personalised Images and even script logic, to change the behaviour of the document.

Even though we will not create Sets or Documents, one has to keep in mind, that personalizing Documents is more complex then Images and that of course affects the interface and its usage. This is why we will start with Images and then work our way up to the Document Interfaces.

Images and Documents are stored in the Repository on your DirectSmile Online Server. To access the Server, a valid Login is needed. Each account on a DirectSmile Online Server has its own repository. For the Examples here, we have created a login on our DirectSmile Online Server, so that you can use the example Images and Documents that are stored in the Help Account.

To connect to the Account, the Username is “Help” and Password “help”. The URL to the

Server is: <http://dsmo.directsmile.de/dsmo>

As we are constantly concerned to rise the quality of our Products and Documentations, please feel free to mail you suggestions and experiences to me:

Andreas.Spranger@DirectSmile.de

The DirectSmile StreamImage Interface:

The DirectSmile StreamImage Interface handles all Image requests. To invoke a personalisation, we need to build an URL that holds the necessary information to request the personalised Image from the Server. The Server will then render the Image and return the result either as a data stream or as an URL to the Image File that has been calculated.

The necessary information that is needed for DirectSmile Online to determine the image to render and the text on the image is included in the URL.

The minimum Information that is needed by DirectSmile Online is:

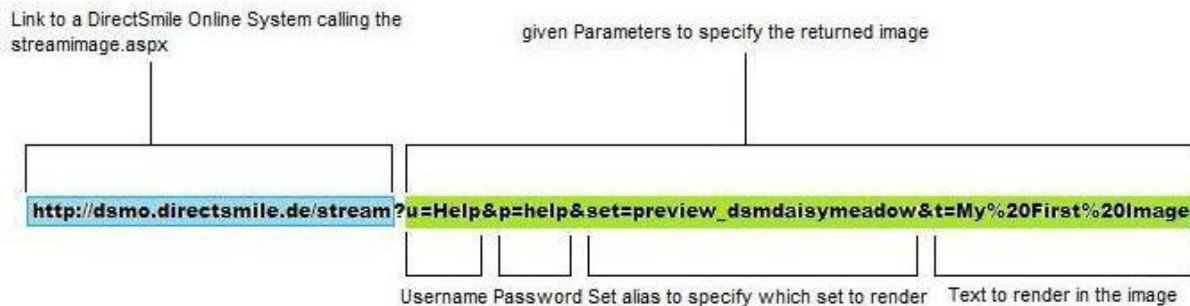
A valid login information to authenticate

A SetAlias to determine which set to render

Some Text to render

With this information, we are able to generate our first Picture with the following Link:

http://dsmo.directsmile.de/dsmo/stream?u=Help&p=help&set=preview_dsmdaisymeadow&t=My%20First%20Image



To include the rendered image into an html page, all you need to do is to define an image container `` on your page and set the image source to the DirectSmile StreamImage URL that matches you needs.

Here is an example of an easy webpage that includes the above image. Just copy the passage into a text file and name it "MyFirstImage.html".

```
<!DOCTYPE html>
<html>
  <head>
    <meta content="text/html; charset=ISO-8859-1" http-equiv="content-type">
  </head>
  <body>
    <label for="myDSMImage">My first DirectSmileOnline Image:
    
  </body>
</html>
```

Streaming Image Parameters:

To define the exact details of the output, the DirectSmile StreamingImage Interface supports a rich set of Parameters. Each of them can be added to the URL to alter the result. The syntax to add a parameter is “&” + Parameter + “=” + “Parameter Value”.

The following Parameters and Parameter Values are supported by the DirectSmile StreamImage Interface:

Parameters required for authentication...

u=UserName

Username to Authenticate e.g. u=Help

p=PassWord

Password to Authenticate e.g. p=help

You need to provide a valid login to authenticate to the DirectSmile Online Server. The login information also defines the account and repository that will be available. Each Account on a DirectSmile Online Server has its own specific repository.

http://dsmo.directsmile.de/dsmo/stream?u=Help&p=help&set=preview_dsmdaisymeadow&t=authentication

ac=Authenticationcode

A string Value that identifies a User by a unique Hash Value e.g

ac=BDF4A2A0956202CE6119B5EBBD8A66D

To avoid the usage of Usernames and Passwords in the URL, you can provide an AuthenticationCode, to access the DirectSmile Online Server. The AuthenticationCode can be retrieved in the Users Section of the DirectSmile Frontend. For the user “Help”, the Authentication Code is:

C6788A579A351018E8FAB48F619FBE9

http://dsmo.directsmile.de/dsmo/stream?ac=BDF4A2A0956202CE6119B5EBBD8A66D&set=preview_dsmdaisymeadow&t=secure%20authentication

Standard Parameters for Set rendering...

set=SetAlias

The name of the Set to render e.g. *set=preview_dsmdaisymeadow*

t=TextToRender

The text to render onto the Image e.g. *t=Joe%20Handsome* please refer to the “Treatment of special characters in URLs” section below for further advice.

pw=PixelWidth

The width of the output image in Pixel. e.g. *pw=1024* The maximum output width of a watermarked “preview Image” is 600. Rendering watermarked preview images costs no smileys. The calculation of images with a maximum width of 600 pixel costs 0.2 smileys. The maximum output width of a non preview image is the original width of the Set.

wt=0 / wt=1 / wt=2

Watermark type. E.g. *wt=1* The watermark type defines whether an image is imprinted with a watermark image. If a watermark is shown on the image, the rendering of the image is free of smiley charge. *0 = No Watermark, 1 = DirectSmile Watermark, 2 = Preview Watermark.*

co=20 - 100

The compression of an image alters the quality and the size of the output. A value of 100 equals highest quality and size with absolutely no compression. 20-99 alter the quality and compression. If this parameter is set to 0 or not provided, a value of 65 is set for the preview queue and a value of 95 for the print queue. For an explanation of the queue parameter, see the explanation below.

Special Parameters for advanced image processing...

queue=0 / queue=1

The DirectSmile Online Server offers a variety of load balancing and process priority features. The queue parameter of the StreamImage Interface is one measure to define the priority of the rendering process. If it is set to 0 or not provided, the job will be handled with high priority and will be preferred in the render queue. This setting is usually perfect for online requests, since the customer will then retrieve his request as fast as possible. This is also the reason, why the standard image compression value is set to 65 to speed up processing and reduce data size for this kind of jobs. If this parameter is set to 1, then the standard image compression is set to 95 to retrieve a high quality image. This second job queue is used to process print jobs, that need a high quality output but do not need to be rendered in a time critical process.

cache=off

Whenever the DirectSmile online Server processes an image or a document, the result is stored in a local cache repository. To prevent the Server to return a cached image but instead render a new image this parameter can be set to off.

http://dsmo.directsmile.de/dsmo/stream?ac=BDF4A2A0956202CE6119B5EBBD8A66D&set=dsm_clouds&co=100&t=caching%20enabled

You can test the effect of this behaviour by loading these links into your web browser. If you hit the reload button in your browser after the image showed up the first time, the upper link will instantly return the cached image, while the lower link will recalculate the image and return it after a view seconds.

http://dsmo.directsmile.de/dsmo/stream?ac=BDF4A2A0956202CE6119B5EBBD8A66D&set=dsm_clouds&co=100&t=caching%20disabled&cache=off

err=text

Usually, the result of a stream call will be a picture that will then be loaded into an image frame. This is reason why the StreamImage interface will return errors the same way as an image. Setting the err parameter to text will result in a text error message.

The following link does not provide a valid authentication code, which will result in an error message. In this case the error is returned as an image.

http://dsmo.directsmile.de/dsmo/stream?ac=NONE&set=dsm_clouds&t=err%20Parameter

While this link will return the error as plain text:

http://dsmo.directsmile.de/dsmo/stream?ac=NONE&set=dsm_clouds&err=txt&t=err%20Parameter

return=url

If you need the URL to the rendered image instead of a DataStream that includes the image, this parameter will result in a plain text URL that refers to the rendered Image.

http://dsmo.directsmile.de/dsmo/stream?ac=BDF4A2A0956202CE6119B5EBBD8A66D&set=dsm_clouds&t=image%20stream

The above link will return the image, while the below link will provide the plain text URL to the image.

http://dsmo.directsmile.de/dsmo/stream?ac=BDF4A2A0956202CE6119B5EBBD8A66D&set=dsm_clouds&t=image%20stream&return=url

If the return Parameter is set to URL, then all error messages will be returned as plain text as well.

!!! Please make sure, that you always request URLs at runtime since the garbage collector process on the DirectSmile Online Server may have removed the cached image which would render the URL invalid!!!

crop=on

The cropping parameter can be set to off, to provoke the rendering of only that part of the image, which is actually altered to render the text into it. The below link returns the cropped image.

Since the cropped image is always returned in the original size of the set, the pw (Pixel Width) Parameter is not allowed when you set Image Cropping to on.

http://dsmo.directsmile.de/dsmo/stream?ac=BDF4A2A0956202CE6119B5EBBD8A66D&set=dsm_clouds&t=cropping&crop=on

To retrieve not only the cropped image but also an XML file which describes the cropping, you can set the return parameter to URL.

http://dsmo.directsmile.de/dsmo/stream?ac=BDF4A2A0956202CE6119B5EBBD8A66D&set=dsm_clouds&t=cropping&crop=on&return=url

If you then replace the suffix of the provided filename from jpg to xml, you can access the corresponding cropping description.

<http://dsmo.directsmile.de/dsmimages/AC1/6F/FB/1095A5CF122A6318EF80FC069B7E.jpg>
<http://dsmo.directsmile.de/dsmimages/AC1/6F/FB/1095A5CF122A6318EF80FC069B7E.xml>

As you can see, the XML file includes all necessary information to position the cropped on the background image.

bg=1

To retrieve the blank background image of the set, you can use the bg parameter.

http://dsmo.directsmile.de/dsmo/stream?ac=BDF4A2A0956202CE6119B5EBBD8A66D&set=_dsm_clouds&bg=1

Parameters for animation rendering...

`aniframe=[n]`

It is as well possible, to render animations with directsmile as gif files. To render an animation works exactly as an image prozess. The only difference is that if you just want to render one frame of the animation as a still image, you can provide the aniframe parameter, to specify which picture frame you want to render. Just provide the Frame Number e.g.

`aniframe=32`

Treatment of Special Characters in URLs:

Due to the formatting of URLs, some characters have to be treated special.

Whenever you send an URL to a Server, the server reads the URL to determine, what to do with it. In the DirectSmile StreamImage URL, some characters have special meanings, such as the “&” and the “=” sign.

http://dsmo.directsmile.de/dsmo/stream?username=Help&password=help&setalias=preview_dsmdaisymeadow&textinimage=My%20First%20Image

These signs define the given Parameters. If we want to have those signs in the “Text to Render”, they have to be encoded. Each character is simply replaced with a set of characters, to let the server know, that this character is not meant to define a parameter.

The following table shows, the encoding that represents the most common special characters.

Character	Encoding	Name
	%20	SPACE
!	%21	EXCLAMATION MARK
"	%22	QUOTATION MARK
#	%23	NUMBER SIGN
\$	%24	DOLLAR SIGN
%	%25	PERCENT SIGN
&	%26	AMPERSAND
'	%27	APOSTROPHE
(%28	LEFT PARENTHESIS
)	%29	RIGHT PARENTHESIS
*	%2a	ASTERISK
+	%2b	PLUS SIGN
,	%2c	COMMA
-	%2d	HYPHEN-MINUS
.	%2e	FULL STOP
/	%2f	SOLIDUS
:	%3a	COLON
;	%3b	SEMICOLON
<	%3c	LESS-THAN SIGN
=	%3d	EQUALS SIGN
>	%3e	GREATER-THAN SIGN
?	%3f	QUESTION MARK
@	%40	COMMERCIAL AT

E.g. the Text: “he&she+us=them” equals the encoded string “he%26she%2bus%3dthem”.

A full Unicode Table can be found at: <http://www.utf8-chartable.de/>

This java script function receives a string, and replaces all special characters by their corresponding encoding.

```

function text_encode(rawtext)
{
  /*******
  /**      replace cr/lf with /lf      *
  /*******
  rawtext = rawtext.replace(/\r\n/g, "\n");
  var encodedtext = "";
  for (var n=0; n<rawtext.length; n++)
  {
  /*******
  /**      get unicode for current character      *
  /*******
  var c=rawtext.charCodeAt(n);
  /*******
  /**      all characters from 0-127 => 1byte      *
  /*******
  if (c<128)
  {
    encodedtext += String.fromCharCode(c);
  }
  /*******
  /**      all characters from 127 to 2047 => 2byte      *
  /*******
  else if((c>127) && (c<2048))
  {
    encodedtext += String.fromCharCode((c>>6)|192);
    encodedtext += String.fromCharCode((c&63)|128);
  }
  /*******
  /**      all characters from 2048 to 66536 => 3byte      *
  /*******
  else
  {
    encodedtext += String.fromCharCode((c>>12)|224);
    encodedtext += String.fromCharCode(((c>>6)&63)|128);
    encodedtext += String.fromCharCode((c&63)|128);
  }
  }
  /*******
  /**      escape remaining special characters      *
  /*******
  txt = escape(encodedtext);
  /*******
  /**      replace + signs by "%2B" since they must be treated seperately *
  /*******
  encodedtext = txt.replace(/\+/, "%2B");
  return encodedtext;
}

```

Using the Stream Image Interface dynamically

This example is going to show you, how to use a java function, to generate Images with variable Text.

Let's start with the example above, copy the following passage into a text file and name it "DirectSmile.html".

```
<!DOCTYPE html>
<html>
  <head>
    <meta content="text/html; charset=ISO-8859-1" http-equiv="content-type">
  </head>
  <body>
    <form method="post" name="form1" onsubmit="return Preview('');"><br/><br/>
      <H4>Choose the Text you want to render then click "Let's go"</H4>
      <label for="name">Text for the personalisation
      <input name="name" id="name" value="DirectSmile" type="text">
    </label><br/>
      <input name="button" onclick="Preview();" value="Let's go"
type="button"><br/>
    </form>

    <label for="myDSMImage">My first dynamic DirectSmileOnline Image:<br/>
    <img src="" name="preview" id="myDSMImage">
  </body>
  <script language="JavaScript" type="text/javascript">
function Preview()
{
<!--
    var txt = document.getElementById('name').value.substr(0, 50);
    var url =
"http://dsmo.directsmile.de/stream?ac=BDF4A2A0956202CE6119B5EBBD8A66D"
    + "&setalias=preview_dsmdaisymeadow&textinimage=" + txt;

    document.getElementById("myDSMImage ").src = url;
    return true;
}
//-->
</script>
</html>
```

Mainly we got a HTML page here with a Text field to enter the Text to render:

```
<label for="name">Text for the personalisation
<input name="name" id="name" value="DirectSmile" type="text">
```

And a button, to start the rendering:

```
<input name="button" onclick="Preview();" value="Let's go" type="button"><br/>
```

In the lower part of the HTML file, below the `</Body>` tag, is the java code, that creates the StreamingImage URL and enters it into the source value of the image Object on the page.

```
function Preview()
{
<!--
var txt = document.getElementById('name').value.substr(0, 50);
    var url =
"http://dsmo.directsmile.de/streamimage.aspx?ac=BDF4A2A0956202CE6119B5EBBD8A66D"
    + "&setalias=preview_dsmdaisymeadow&textinimage=" + txt;

    document.getElementById("preview").src = url;
    return true;
}
```

If you load the page into a browser, enter some Text and hit the “Let’s go” button, it will render and show the Image.

To avoid special characters, to mess up the link, we can now add the function mentioned above, to format the Text to render properly. Just copy the function below the Preview function into the HTML page and add the red line to the code.

To make the code more readable, I removed the comments in this section.

```
function Preview()
{
<!--
var txt = document.getElementById('name').value.substr(0, 50);
txt = text_encode(txt);
var url = "http://dsmo.directsmile.de/streamimage.aspx?ac=BDF4A2A0956202CE6119B5EBBD8A66D"
    + "&setalias=preview_dsmdaisymeadow&textinimage=" + txt;

    document.getElementById("preview").src = url;
    return true;
}

function text_encode(rawtext)
{
    rawtext = rawtext.replace(/\r\n/g, "\n");
    var encodedtext = "";
    for(var n=0; n<rawtext.length; n++)
    {
        var c=rawtext.charCodeAt(n);
        if (c<128)
        {
            encodedtext += String.fromCharCode(c);
        }
        else if((c>127) && (c<2048))
        {
            encodedtext += String.fromCharCode((c>>6)|192);
            encodedtext += String.fromCharCode((c&63)|128);
        }
        else
        {
            encodedtext += String.fromCharCode((c>>12)|224);
            encodedtext += String.fromCharCode(((c>>6)&63)|128);
            encodedtext += String.fromCharCode((c&63)|128);
        }
    }
    txt = escape(encodedtext);
    encodedtext = txt.replace(/\+/g, "%2B");
    return encodedtext;
}
```

<http://dsmo.directsmile.de/Examples/DirectSmile.html>

The DirectSmile StreamDoc Interface:

The DirectSmile StreamDocument Interface is very similar to the StramImage Interface. To invoke a personalisation, we need to build an URL that holds the necessary information to request the personalised Document from the Server. The Server will then render the Document and return the result either as a data stream or as an URL to the Image File that has been calculated.

The necessary information that is needed for DirectSmile Online to determine the Document to render and the text of the variables inside the Document is coded into the URL.

The minimum Information that is needed by DirectSmile Online is:

A valid login information to authenticate

A DocumentAlias to determine which set to render

A Variable Name and Some Text to render

With this information, we are able to generate our first Picture with the following Link:

<http://dsmo.directsmile.de/dsmo/streamdocument.aspx?u=Help&p=help&doc=calendarV3&-01NameInPicture=My%20First%20Document>

Link to the DirectSmile Online Server calling the streamdocument.aspx	given Parameters to specify the rendered Document			
<input type="text"/>	<input type="text"/>			
http://dsmo.directsmile.de/dsmo/streamdocument.aspx?u=Help&p=help&doc=CalendarV3&-01NameInPicture=My%20First%20Document				
<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>
Usenam	Password	Document Alias	Variable Name	Text to Render
Variable in the Document and the Text to render				

Besides the fact, that Documents need Variable Names to define which text goes where inside the Document and some more Parameters, that are supported by the StreamDocument interface, this interface works exactly like the StreamImage Interface.

Streaming Document Parameters:

To define the exact details of the output, the DirectSmile StreamingDocument Interface supports a rich set of Parameters. Each of them can be added to the URL to alter the result. The syntax to add a parameter is "&" + Parameter + "=" + "Parameter Value".

The only exception to this is are the Variable Names that are added to the URL to define the text to render inside the variables of the Document. The Variable Name is given as a Parameter with a leading "-" (Minus) sign to define it as a Variable Name and the Text to render is provided as the parameter Value.

The following Parameters and Parameter Values are supported by the DirectSmile StreamDocument Interface:

Parameters required for authentication...

u=UserName

Username to Authenticate e.g. u=Help

p=PassWord

Password to Authenticate e.g. p=help

You need to provide a valid login to authenticate to the DirectSmile Online Server. The login information also defines the account and repository that will be available. Each Account on a DirectSmile Online Server has its own specific repository.

<http://dsmo.directsmile.de/dsmo/streamdocument.aspx?u=Help&p=help&doc=CalendarV3&-01NameInPicture=authentication>

ac=Authenticationcode

A string Value that identifies a User by a unique Hash Value e.g

ac=BDF4A2A0956202CE6119B5EBBD8A66D

To avoid the usage of Usernames and Passwords in the URL, you can provide an AuthenticationCode, to access the DirectSmile Online Server. The AuthenticationCode can be retrieved in the Users Section of the DirectSmile Frontend. For the user "Help", the Authentication Code is:

C6788A579A351018E8FAB48F619FBE9

<http://dsmo.directsmile.de/dsmo/streamdocument.aspx?ac=BDF4A2A0956202CE6119B5EBBD8A66D&doc=CalendarV3&-01NameInPicture=secure%20authentication>

Standard Parameters for Document rendering...

doc=DocAlias

The name of the Set to render e.g. *doc=calendarV3*

A Variable and its Value (regard the leading minus)

-01NameInPicture=TextToRender

The text to render onto the Image e.g. *-01NameInPicture=Joe%20Handsome* please refer to the “Treatment of special characters in URLs” section in the StreamImage chapter for further advice.

page=x

The page you want the StreamDocument Interface to return. Documents usually have various pages. This Parameter can be used to return a specific page instead of the whole document. X equals the number of the page, that you want the system to return.

hires=0 / hires=1

Usually this parameter should be set to 0, to return a Web Preview rendering of the document, which has a lower resolution and image quality than a print job. Using a WebPreview results in a faster output and a smaller file size, this is optimised for the use in a Web Browser. If you set this parameter to 1, the document is returned in full print resolution. Just as an example, the FileSize of a hires rendering of the CalendarV3 is about 22 MegaByte while the Preview is about 2 MegaByte in size.

<http://dsmo.directsmile.de/dsmo/streamdocument.aspx?ac=BDF4A2A0956202CE6119B5EBBD8A66D&doc=CalendarV3&-01NameInPicture=PDFPreview&hires=0#>

<http://dsmo.directsmile.de/dsmo/streamdocument.aspx?ac=BDF4A2A0956202CE6119B5EBBD8A66D&doc=CalendarV3&-01NameInPicture=hires%20PDF&hires=1#>

impo=0 / impo=1

This parameter defines whether the document should include the impositions that are used to print the job on a press. A value of 1 means that the impositions are shown in the document.

<http://dsmo.directsmile.de/dsmo/streamdocument.aspx?ac=BDF4A2A0956202CE6119B5EBBD8A66D&doc=CalendarV3&-01NameInPicture=Imposition&impo=1#>

pw=x

The width in pixel that the **output page** is returned in. If this value is set to a number higher than 80, then the JPG file that is returned when using the page=x parameter has the specified width.

co=20 - 100

The compression of an image alters the quality and the size of the output. A value of 100 equals highest quality and size with absolutely no compression. 20-99 alter the quality and compression. If this parameter is set to 0 or not provided, a value of 65 is set for images in preview jobs.

Special Parameters for advanced document processing...

queue=0 | queue=1

The DirectSmile Online Server offers a variety of load balancing and process priority features. The queue parameter of the StreamImage Interface is one measure to define the priority of the rendering process. If it is set to 0 or not provided, the job will be handled with high priority and will be preferred in the render queue. This setting is usually perfect for online requests, since the customer will then retrieve his request as fast as possible. This is also the reason, why the standard image compression value is set to 65 to speed up processing and reduce data size for this kind of jobs. If this parameter is set to 1, then the standard image compression is set to 95 to retrieve a high quality image. This second job queue is used to process print jobs, that need a high quality output but do not need to be rendered in a time critical process.

cache=off

Whenever the DirectSmile online Server processes an image or a document, the result is stored in a local cache repository. To prevent the Server to return a cached document or image but instead render a new job this parameter can be set to off.

<http://dsmo.directsmile.de/dsmo/streamdocument.aspx?ac=BDF4A2A0956202CE6119B5EBBD8A66D&doc=CalendarV3&-01NameInPicture=caching\%20enabled>

You can test the effect of this behaviour by loading these links into your web browser. If you hit the reload button in your browser after the image showed up the first time, the upper link will instantly return the cached document, while the lower link will recalculate the job and return it after a view seconds.

<http://dsmo.directsmile.de/dsmo/streamdocument.aspx?ac=BDF4A2A0956202CE6119B5EBBD8A66D&doc=CalendarV3&-01NameInPicture=caching\%20disabled&cache=off>

err=text

Usually, the result of a stream call will be a picture or a PDF file that will then be loaded into an image frame or iframe object inside your webpage. This is reason why the StreamImage interface will return errors the same way, as an image. Setting the err parameter to text will result in a text error message.

The following link does not provide a valid authentication code, which will result in an error message. In this case the error is returned as an image.

<http://dsmo.directsmile.de/dsmo/streamdocument.aspx?ac=NONE&doc=CalendarV3&-01NameInPicture=err%20Parameter>

While this link will return the error as plain text:

<http://dsmo.directsmile.de/dsmo/streamdocument.aspx?ac=NONE&doc=CalendarV3&-01NameInPicture=err%20Parameter&err=txt>

return=url

If you need the URL to the rendered PDF instead of a DataStream that includes the image, this parameter will result in a plain text URL that refers to the rendered Document.

<http://dsmo.directsmile.de/dsmo/streamdocument.aspx?ac=BDF4A2A0956202CE6119B5EBBD8A66D&doc=CalendarV3&-01NameInPicture=image%20stream&return=url>

The above link will return the PDF, while the below link will provide the plain text URL to the PDF

<http://dsmo.directsmile.de/dsmo/streamdocument.aspx?ac=BDF4A2A0956202CE6119B5EBBD8A66D&doc=CalendarV3&-01NameInPicture=image%20stream&return=url>

If the return Parameter is set to URL, then all error messages will be returned as plain text as well.

!!! Please make sure, that you always request URLs at runtime since the garbage collector process on the DirectSmile Online Server may have removed the cached Document which would render the URL invalid!!!

Web Service Functions

To further enhance the usage of the DirectSmile Online Server, we need to take a look at another Interface of the Server. The webservice.

Before we start to use the webservice of your DirectSmile Online Server, we need to address some issues that arise when you call code from a Server.

JavaScript:

If you want to call the webservice using JavaScript, it is necessary, to have the HTML pages and the JavaScript on the same Domain, that runs the service. This means, when you create a Webpage, calling the webservice with Java, it needs to reside on the same server as the DirectSmile Online.

PHP:

If you are using PHP script to call the functions, then there is no need for these pages to reside on the DirectSmile Server. However, the server that hosts the Websites needs to have PHP 5 enabled and running. (And you have to make sure, that the PHP Soap Client functionality is enabled.)

We will address both cases separately in this Documentation.

To simplify the usage of the webservices, we will provide a basic class that includes all of the Webservice Functions, to simplify the calls.

For a detailed overview of all Functions that are supported by the DirectSmile webservice and the return values, you can call <http://dsmo.directsmile.de/dsmo/lb.asmx> or the corresponding page on your server <http://myDSMOServer/dsmo/lb.asmx>.

JavaScript and the DirectSmile webservice:

To work through these examples, please access the FileSystem on your DirectSmile Online Server and add a folder Examples, where we can store the Pages, which we will create. The Internet Information Server by default stores its files to “C:\Inetpub\wwwroot\”. The DirectSmile Online will be installed in the DSMO sub folder: “C:\Inetpub\wwwroot\DSMO”.

Please create a folder “C:\Inetpub\wwwroot\Examples”, to store the examples to.

Please refer to <http://dsmo.directsmile.de/examples> to try the Examples in this Document. All examples are free for downloading them.

Now save a copy of the DSMOHelper.js and the DocTest1.html files in this Folder. Once done, you can open the DocTest1.html with a WebBrowser.

<http://dsmo.directsmile.de/Examples/DocTest1.html>

The DSMOHelper.js holds all necessary Functions of the Webservice and handles the Ajax Calls and responses to simplify the usage of the Service. It is open source and experienced WebDevelopers might want to have a look into it and alter the functionality to their needs.

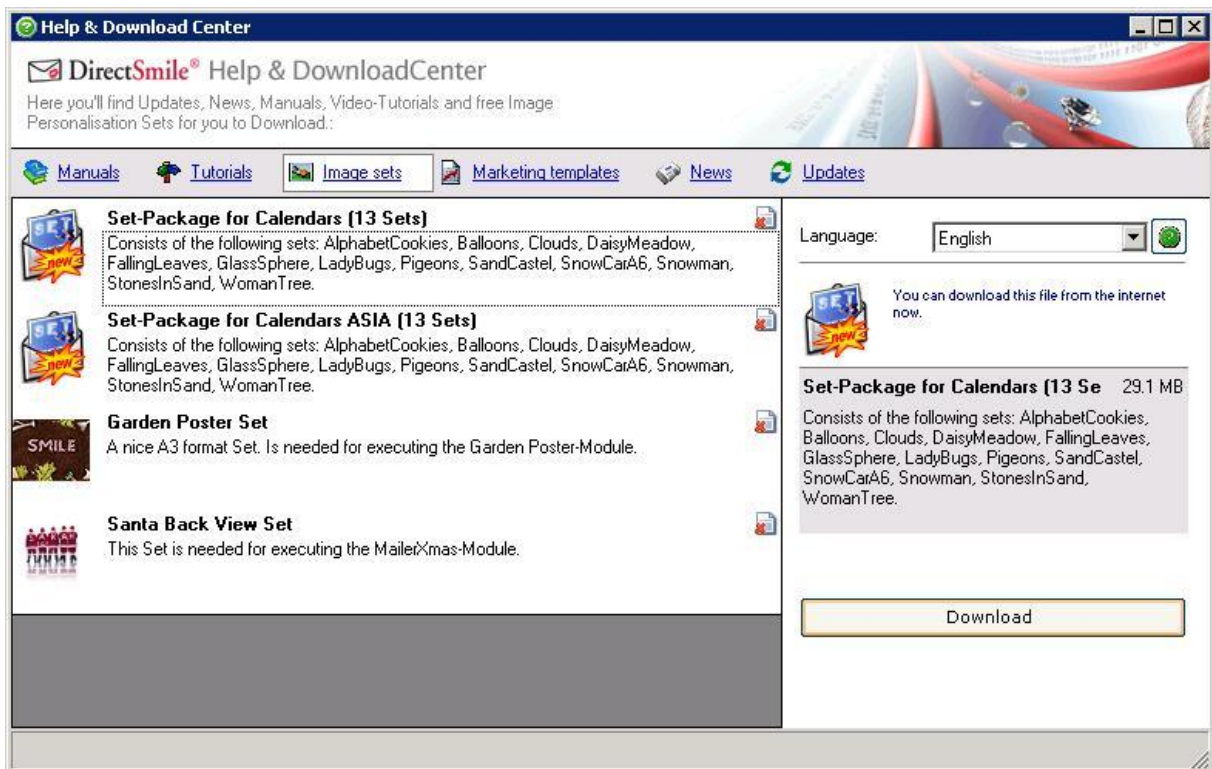
What we need to get started

For the Examples, we are going to use in this documentation, you need the CalendarV3 Document and the sets that are used in it. If you are using your own DirectSmile Online Server, you need to download and install the Document from the DirectSmile Help and Download Center.

To do so, open the DirectSmile Generator Window on your DirectSmile Online Server. Then choose “Open Help and DownloadCenter”:

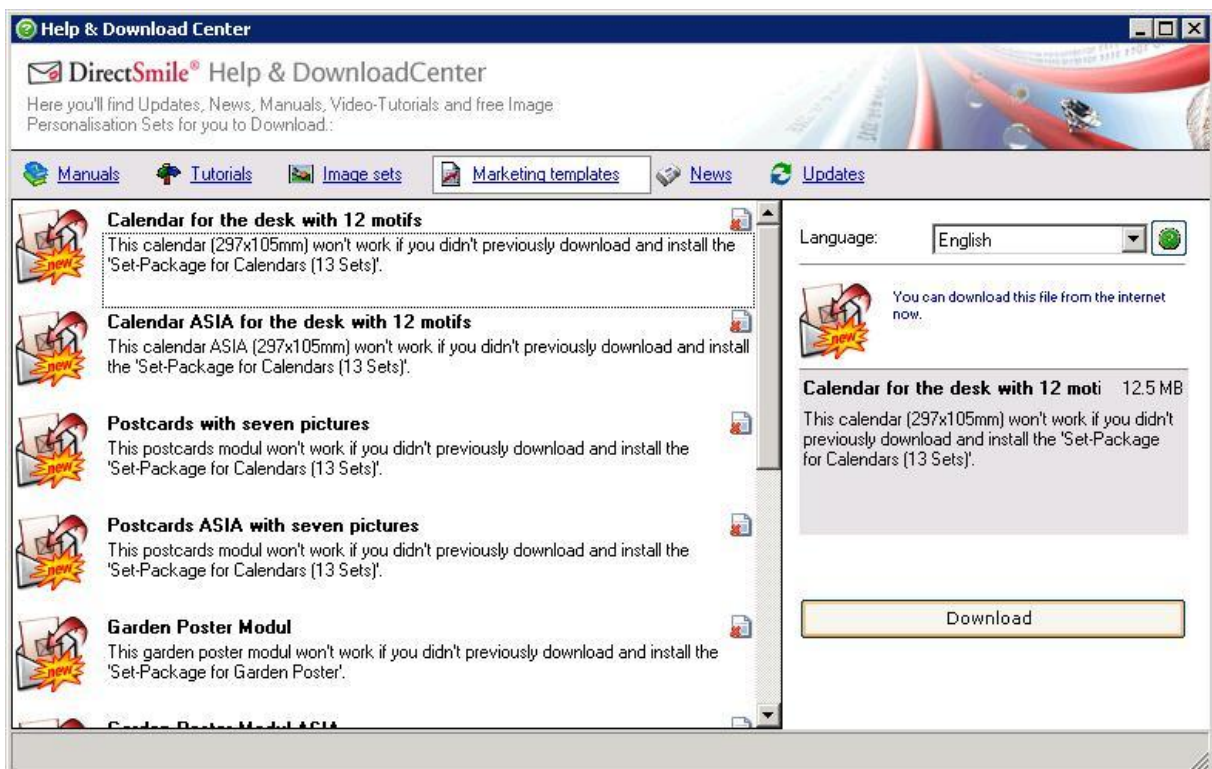


Once the Help and DownloadCenter opened, navigate to the ImageSets at the top bar:



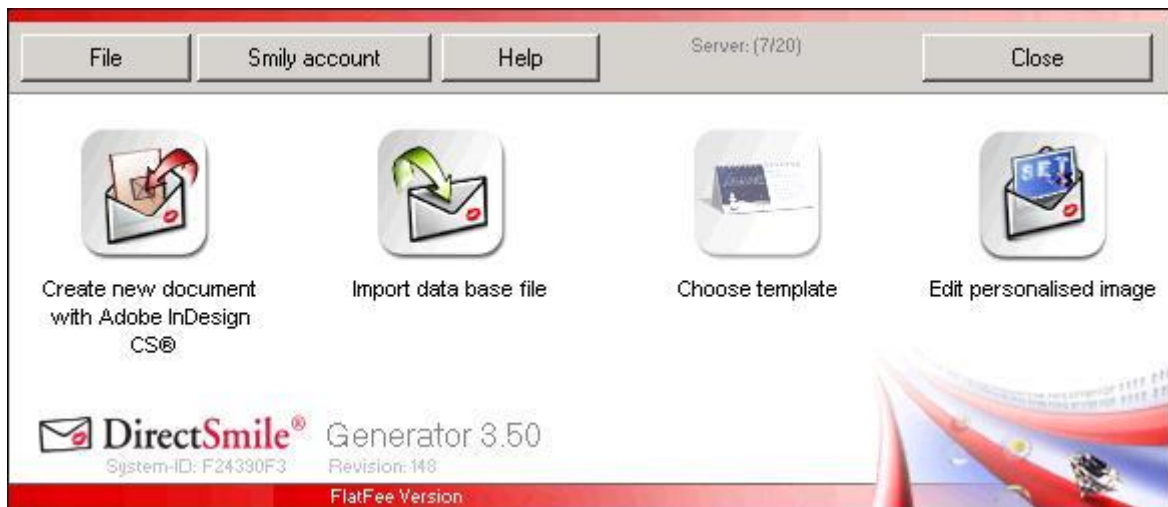
And choose the Set-Package for Calendars (13 Sets). Then hit the download button.

After we downloaded the Sets that are needed in the Document we will download the Document itself by choosing “Marketing Templates” on the top bar.

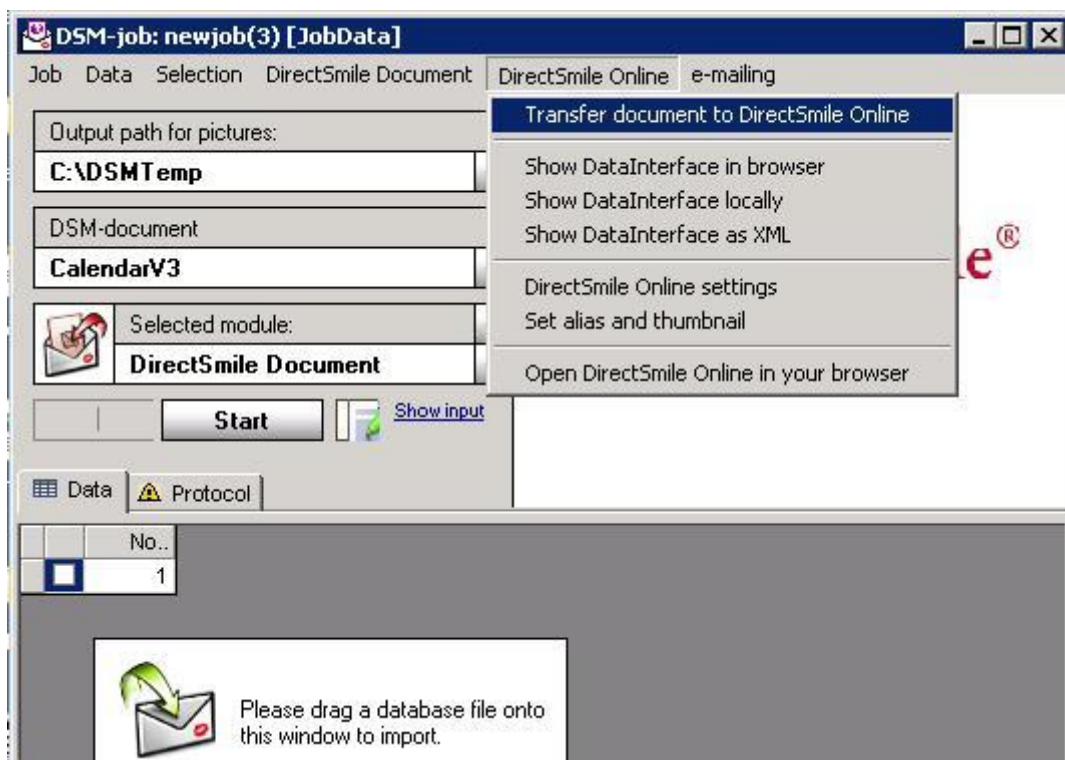


Choose “Calendar for the desk with 12 motifs” and hit the download button.

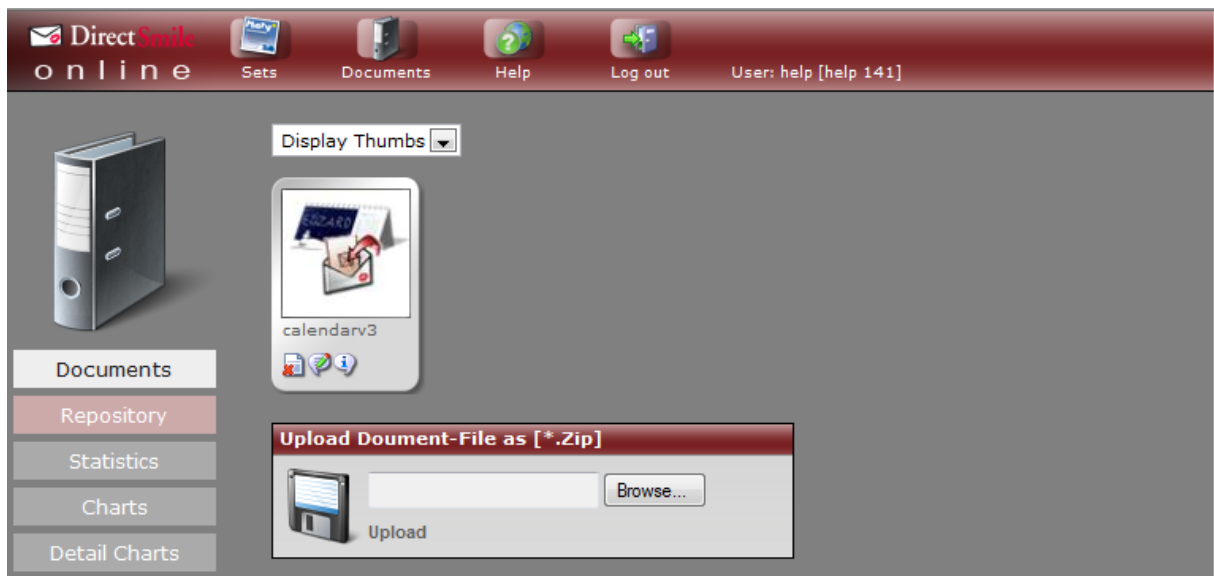
Now we need to transfer the Document into the DirectSmile Online System. To do so, choose the CalendarV3 template under Templates in your DirectSmile Generator:



And finally choose the Menu “DirectSmile Online” and here “Transfer document to DirectSmile Online”.



After that the Document will appear in the Document Section of your DirectSmile Online user.



Now that we have all that we need, let's get to work.....

Creating Documents using the DirectSmile Web Services and Java Script

Let's have a look at the content of the DocTest1.html:

```
<html>
  <meta content="text/html; charset=ISO-8859-1" http-equiv="content-type">
  <title>DocTest1.html</title>
</head>
  <script type="text/javascript" src="./DSMOHelper.js"></script>
  <script language="JavaScript" type="text/javascript">
<!--
var MyMsgComXML = '<?xml version="1.0" encoding="utf-8"?>'
+   '<COMMAND>'
+     '<Parameters>'
+       '<FandV FieldName="OUTPUT" FieldValue="PDFPreview"/>'
+     '</Parameters>'
+     '<StaticFields>'
+       '<FandV FieldName="01NameInPicture" FieldValue="&quot;my first
Document&quot;"/>'
+       '<FandV FieldName="StartMonth" FieldValue="&quot;1&quot;"/>'
+       '<FandV FieldName="mYear" FieldValue="&quot;2009&quot;"/>'
+       '<FandV FieldName="zHemisphere" FieldValue="&quot;North&quot;"/>'
+       '<FandV FieldName="Claim" FieldValue="&quot;Printing Emotions&quot;"/>'
+       '<FandV FieldName="Logo" FieldValue="&quot;DirectSmile_4c.eps&quot;"/>'
+     '</StaticFields>'
+   '</COMMAND>';

function GetDocument()
{
  var SessionID = '';
  var DocKey = "";
  var DocStatus = new Object();

  SetConstants(location.host);

  SessionID = Authenticate('Help','help');

  ResultValue = StartDoc(SessionID, "CalendarV3", MyMsgComXML);
  DocStatus = ParseDocStatusXML(ResultValue);
  DocKey = DocStatus.Key

  while (DocStatus.LastProgressPercent != 100)
  {
    DocStatus = ParseDocStatusXML(GetDocStatus(SessionID, DocKey));
  }
  if (DocStatus.PDFUrl != '') document.getElementById('iframe1').src = DocStatus.PDFUrl;
}
-->
</script>
<body>
<!-- <form method="post" name="form1" onsubmit="return GetDocument('');">DirectSmile Online
Examples.<br/><br/> -->
  <form method="post" name="form1" onsubmit="">DirectSmile Online Examples.<br/><br/>
  <input name="button" onclick="GetDocument();" value="Let's go"
type="button"><br/>
  </form>
  <iframe id="iframe1" src="" name="Doc in the box" width="640" height="480"
leftmargin="10" topmargin="10"/>
</body>
</html>
```

The Document itself mainly consists of a Button to start the rendering and an Iframe, that is supposed to show the output.

The Interesting part is the Java Script in-between.

```
var MyMsgComXML = '<?xml version="1.0" encoding="utf-8"?>'
+   '<COMMAND>'
+     '<Parameters>'
+       '<FandV FieldName="OUTPUT" FieldValue="PDFPreview"/>'
+     '</Parameters>'
+     '<StaticFields>'
+       '<FandV FieldName="01NameInPicture" FieldValue="&quot;my first
Document&quot;"/>'
+       '<FandV FieldName="StartMonth" FieldValue="&quot;1&quot;"/>'
+       '<FandV FieldName="mYear" FieldValue="&quot;2009&quot;"/>'
+       '<FandV FieldName="zHemisphere" FieldValue="&quot;North&quot;"/>'
+       '<FandV FieldName="Claim" FieldValue="&quot;Printing Emotions&quot;"/>'
+       '<FandV FieldName="Logo" FieldValue="&quot;DirectSmile_4c.eps&quot;"/>'
+     '</StaticFields>'
+   '</COMMAND>';

function GetDocument()
{
    var SessionID = '';
    var DocKey = "";
    var DocStatus = new Object();

    SetConstants(location.host);

    SessionID = Authenticate('Help','help');

    ResultValue = StartDoc(SessionID, "CalendarV3", MyMsgComXML);
    DocStatus = ParseDocStatusXML(ResultValue);
    DocKey = DocStatus.Key

    while (DocStatus.LastProgressPercent != 100)
    {
        DocStatus = ParseDocStatusXML(GetDocStatus(SessionID, DocKey));
    }
    if (DocStatus.PDFUrl != '') document.getElementById('iframe1').src = DocStatus.PDFUrl;
}
```

The first section of the JavaScript (blue) builds a Variable that is called MyMsgComXML. This structure is used to tell the DirectSmile Server what to do. In general it holds all Job Parameters to define the Output. For a detailed description of the MsgComXML, please refer to the MsgComXML Section of this documentation.

The rest of the source is a function that starts the rendering of a document, monitors the progress and returns the URL to the rendered PDF file (red).

The first thing we do is to tell the Service, which domain is invoking the WebService.

```
SetConstants(location.host);
```

After that, we use the Authenticate Function, to establish a connection to the DirectSmile WebService. The required Parameters are Username and Password, to authenticate. The return value is a session ID, that we will use further on to communicate with the Service.

```
    SessionID = Authenticate('Help','help');
```

Once we are authenticated and have a Session, we call the StartDoc Function. It requires three Parameters to run, a valid SessionID, the name of the Document, that we want to render and a valid MsgComXML to define the JobParameters. The Name of the Document can be retrieved from the Documents Section of the DirectSmile Server Frontend. It is referred to as DocumentAlias.

```
ResultValue = StartDoc(SessionID, "CalendarV3", MyMsgComXML);
```

In return, we get an XML that holds some Information about the Job we just started. Since the Result is in XML formatted, the DSMOHelper.js comes with a function, to parse the result and write its content into an Object, for easier handling.

```
DocStatus = ParseDocStatusXML(ResultValue);
```

The next line of code stores the DocumentKey, to identify the Document Rendering Job.

```
DocKey = DocStatus.Key
```

Once the Job is started, the DirectSmile Online Server needs a few seconds to process the job. We will invoke the GetDocStatus Function to retrieve the Document Rendering status. This Function requires two Parameters, the SessionID and the DocKey, to define which document we are referring to.

It as well returns a DocStatus which we again will send through the ParseDocStatusXML function, to renew the content of the DocStatus object.

```
while (DocStatus.LastProgressPercent != 100)
{
    DocStatus = ParseDocStatusXML(GetDocStatus(SessionID, DocKey));
}
```

Once our job is fully rendered, we can use the PDFUrl Value of the DocStatus, to refer to the rendered PDF.

```
if (DocStatus.PDFUrl != '') document.getElementById('iframe1').src = DocStatus.PDFUrl;
```

The DocStatus return Value

Now that we have rendered our first Document, let's have a closer look at the DocStatus, which we get in return, whenever we call the StartDoc or GetDocStatus Functions:

```
<?xml version="1.0" encoding="utf-8"?>
<DSMDocRenderStatus
  State="0"
  LastProgressMsg=""
  LastProgressPercent=""
  LastErrorMsg="Document request successfully queued"
  PDFUrl=""
  Key="697060567"
  Updated="11/20/2008 12:46:30 PM"
  Created="11/20/2008 12:46:30 PM"
  LastErrorXMLUrl=""
/>
```

This is the XML Answer that we get when we call StartDoc or GetDocStatus. The included Informations are:

Parameters	Values	Description
State	(Numeric) = 0: Idle 1: Running 2: Failed 3: OK 4: OK with ErrorXML 5: Job Accepted and transferred to the Backend	The Status of the Document Rendering Process
LastProgress Message	(Text)	Last Status change Message.
LastProgressPercent	(Numeric) 0-100	The percentage of the Job that has been calculated.
LastErrorMsg	(Text)	The last Error Message. Please note that not all Errors will halt the calculation.
PDFUrl	(Text)	The URL to the calculated PDF File. Only available after the Job successfully completed.
Key	(Numeric)	The Job Identifier Key
Updated	(Date) mm/dd/yyyy hh:mm:ss AM/PM	Timestamp of when the job was last updated.
Created	(Date) mm/dd/yyyy hh:mm:ss AM/PM	Timestamp of when the job has been created.
LastErrorXMLUrl	(Text) Url	URL to the ErrorXML that holds further details about job errors.
Jpeg Section		If you used the Parameter CONVERTTO, the following sections refers to the calculated JPEG images of the Document.
Page	(Numeric)	The Page No. of the PDF that this JPG is referred to.
URL	(Text) URL	The URL to the JPEG.

We can use this information dynamically since it is updated each time we call the GetJobStatus Function. E.g. we could create a progress bar on the Website to show the rendering status.

To do so, please copy the DocTest1.html and name it DocTest2.html.

Then we enter the following lines in the <body> section of the Document:

```
<body>
<!-- <form method="post" name="form1" onsubmit="return GetDocument('');">DirectSmile Online
Examples.<br/><br/> -->
  <form method="post" name="form1" onsubmit="">DirectSmile Online Examples.<br/><br/>
    <input name="button" onclick="GetDocument();" value="Let's go"
type="button"><br/>
  </form>
  <TABLE border="0" width="200px" bgcolor="#FFFFFF">
    <TR>
      <TD id="progressBar" width="1%" bgcolor="#FF0000">&#160;</TD>
      <TD width="*" bgcolor="#EEEEEE">&#160;</TD>
    </TR>
  </TABLE><br/>
  <iframe id="iframe1" src="" name="Doc in the box" width="640" height="480"
leftmargin="10" topmargin="10"/>
</body>
```

Now we add one line of code to the JavaScript section to control the progress bar:

```
while (DocStatus.LastProgressPercent != 100)
{
  DocStatus = ParseDocStatusXML(GetDocStatus(SessionID, DocKey));
  document.getElementById('progressBar').width = DocStatus.LastProgressPercent+"%";
}
```

This progress bar is a very simple HTML implementation. It consists of a Table with two rows and one line. The left Field has a red background color set, while the right field is white. Now we adjust the width of the left table field using the DocStatus.LastProgressPercent Value to make it move.

<http://dsmo.directsmile.de/Examples/DocTest2.html>

PHP and the DirectSmile Webservices:

To work through these examples, you need a WebServer with PHP5 running.

To enable Soap, you need to have the "add_extension=php_soap.dll" enabled in the PHP.INI file. Please also check the Soap Configuration Option in your PHP.INI.

Then copy the Example files to your WebSite, to run the Examples.

Please refer to <http://dsmo.directsmile.de/examples> to try the Examples in this Document. All examples are free for downloading them.

<http://dsmo.directsmile.de/Examples/DocTest1.html>

The interfacehelper.php.inc and the pdsmoservice.php.inc files hold all necessary Functions of the Webservice and handle the Ajax Calls and responses to simplify the usage of the Service. It is open source and experienced WebDevelopers might want to have a look into these files to alter the functionality to their needs.

Creating Documents using the DirectSmile Web Services and PHP5

Let's have a look at the content of the DocTest1.html:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml">
  <head>
    <title>DSMO Service Examples</title>
  </head>
  <body>
    <form action="DocTest1.php" method="post">DirectSmile Online
Examples.<br/><br/>
      <input name="submit" type="submit" value="Let's go" />
    </form><br />
<?php
    include_once("dsmoservice.php.inc");
//*****
// The following Variables define the domain, username and password, to access the Webservice
// with this informations we create the Webservice as $service
//*****
    $DSMOUrl = "http://dsmo.directsmile.de/dsmo/";
    $ServiceURL = $DSMOUrl."lb.asmx";
    $Username = "su";
    $Password = "su";
    $service = new DSMOServiceX();
    if (isset($_POST["submit"])) {
//*****
// This is the string holding the MsgComXML that we will use to create the document.
//*****
        $MyMsgComXML = '<?xml version="1.0" encoding="utf-8"?'>';
        $MyMsgComXML .= '<COMMAND>';
        $MyMsgComXML .= '<Parameters>';
        $MyMsgComXML .= '<FandV FieldName="OUTPUT"
FieldValue="PDFPreview"/>';
        $MyMsgComXML .= '</Parameters>';
        $MyMsgComXML .= '<StaticFields>';
        $MyMsgComXML .= '<FandV FieldName="01NameInPicture"
FieldValue="&quot;My first Document&quot;"/>';
        $MyMsgComXML .= '<FandV FieldName="StartMonth"
FieldValue="&quot;1&quot;"/>';
        $MyMsgComXML .= '<FandV FieldName="mYear"
FieldValue="&quot;2009&quot;"/>';
        $MyMsgComXML .= '<FandV FieldName="zHemisphere"
FieldValue="&quot;North&quot;"/>';
        $MyMsgComXML .= '<FandV FieldName="Claim"
FieldValue="&quot;&quot;"/>';
        $MyMsgComXML .= '<FandV FieldName="Logo"
FieldValue="&quot;&quot;"/>';
        $MyMsgComXML .= '</StaticFields>';
        $MyMsgComXML .= '</COMMAND>';
//*****
// This starts the creation of the document. Parameters are:
// DocAlias = the alias of the document we want to create (we just use the 1st
// entry in our AliasArray).
// MsgComXML = the MsgComXML that provides parameters and variables for the Job.
//*****
        $status = $service->StartDocArray("CalendarV3",
utf8_encode($MyMsgComXML));
//*****
// The Document Job gets a Key Value as Identifier. This Key is used to request the Status of
the render
// process in the following calls.
//*****
        $key = $status["Key"];
//*****
// While the Document is still rendering, We keep tracking the progress...
//*****
        while ($status["LastProgressPercent"] < 100) {
            sleep(1);
            $status = $service->GetDocStatusArray($key);
        }
    }
}
```

```

    }
    /*******
    *****
    // Once the Document is fully rendered, we use the PDFUrl to show the Document.
    /*******
    *****
        echo '<iframe id="iframe1" src="'. $status["PDFUrl"].'" name="Doc in
the box" width="640" height="480" leftmargin="10" topmargin="10"/>';
    }
    ?>
</body>
</html>

```

The Document itself mainly consists of a Button to start the rendering and an Iframe, that is supposed to show the output.

The Interesting part is the PHP Script in-between.

```

<?php

include_once("dsmoservice.php.inc");

$DSMOUrl = "http://dsmo.directsmile.de/dsmo/";
$ServiceURL = $DSMOUrl."lb.asmx";
$Username = "help";
$Password = "help";
$service = new DSMOServiceX();
if (isset($_POST["submit"])) {
    $MyMsgComXML = '<?xml version="1.0" encoding="utf-8"?>';
    $MyMsgComXML .= '<COMMAND>';
    $MyMsgComXML .= '<Parameters>';
    $MyMsgComXML .= '<FandV FieldName="OUTPUT"
FieldValue="PDFPreview"/>';
    $MyMsgComXML .= '</Parameters>';
    $MyMsgComXML .= '<StaticFields>';
    $MyMsgComXML .= '<FandV FieldName="01NameInPicture"
FieldValue="&quot;My first Document&quot;"/>';
    $MyMsgComXML .= '<FandV FieldName="StartMonth"
FieldValue="&quot;1&quot;"/>';
    $MyMsgComXML .= '<FandV FieldName="mYear"
FieldValue="&quot;2009&quot;"/>';
    $MyMsgComXML .= '<FandV FieldName="zHemisphere"
FieldValue="&quot;North&quot;"/>';
    $MyMsgComXML .= '<FandV FieldName="Claim" FieldValue="&quot;&quot;"/>';
    $MyMsgComXML .= '<FandV FieldName="Logo" FieldValue="&quot;&quot;"/>';
    $MyMsgComXML .= '</StaticFields>';
    $MyMsgComXML .= '</COMMAND>';

    $status = $service->StartDocArray("CalendarV3", utf8_encode($MyMsgComXML));
    $key = $status["Key"];
    while ($status["LastProgressPercent"] < 100) {

```

```

        sleep(1);
        $status = $service->GetDocStatusArray($key);
    }
    echo '<iframe id="iframe1" src="'. $status["PDFUrl"].'" name="Doc in the box"
width="640" height="480" leftmargin="10" topmargin="10"/>';
}
?>

```

The first section of the PHPScript (blue) builds a Variable that is called MyMsgComXML. This structure is used to tell the DirectSmile Server what to do. In general it holds all Job Parameters to define the Output. For a detailed description of the MsgComXML, please refer to the MsgComXML Section of this documentation.

The rest of the source is a function that starts the rendering of a document, monitors the progress and returns the URL to the rendered PDF file (red).

The first thing we do is to define some Parameters, we need to invoke the Webservice. This includes the Domain we connect to, the location of the Webservice on the Domain and a username and a password.

```

$DSMUrl = "http://dsmo.directsmile.de/dsmo/";
$ServiceURL = $DSMUrl."lb.asmx";
$username = "help";
$password = "help";

```

After that, you can create a new Service connection using the statement:

```
$service = new DSMServiceX();
```

Once we are authenticated and have a Session, we call the StartDoc Function. It requires two Parameters to run. The name of the Document, that we want to render and a valid MsgComXML to define the JobParameters. The Name of the Document can be retrieved from the Documents Section of the DirectSmile Server Frontend. It is referred to as DocumentAlias.

```
$status = $service->StartDocArray("CalendarV3", utf8_encode($MyMsgComXML));
```

In return, we get an XML that holds some Information about the Job we just started. In PHP we have the opportunity, to access this information directly. In this next line of code, e.g. we read the "Key" value of the construct to store it for further use. The Key Value identifies the Document Rendering Job.

```
$key = $status["Key"];
```

Once the Job is started, the DirectSmile Online Server needs a few seconds to process the job. We will invoke the GetDocStatus Function to retrieve the Document Rendering status. This Function requires the DocKey, to define which document we are referring to. It as well returns a DocStatus

```

while ($status["LastProgressPercent"] < 100) {
    sleep(1);
    $status = $service->GetDocStatusArray($key);
}

```

Once our job is fully rendered, we can use the PDFUrl Value of the DocStatus, to refer to the rendered PDF.

```
echo '<iframe id="iframe1" src="' . $status["PDFUrl"] . '" name="Doc in the box" width="640"
height="480" leftmargin="10" topmargin="10"/>';
```

The DocStatus return Value

Now that we have rendered our first Document, let's have a closer look at the DocStatus, which we get in return, whenever we call the StartDoc or GetDocStatus Functions:

```
<encoding="utf-8"?>
<DSMDocRenderStatus
State="3"
PageCount="4"
LastProgressMsg="OK"
LastProgressPercent="100"
LastErrorMsg=""
PDFUrl="http://dsmo.directsmile.de/dsmimages/AD13/98/C8/D84C482ABD00772A6D9ABA24B5D3.pdf"
Key="1726138735"
Updated="1/15/2009 11:37:36 AM"
Created="1/15/2009 11:37:36 AM"
LastErrorXMLUrl="">
<Jpeg Page="1"
Url="http://dsmo.directsmile.de/dsmimages/AD13/98/C8/D84C482ABD00772A6D9ABA24B5D3._001.jpg" />
<Jpeg Page="2"
Url="http://dsmo.directsmile.de/dsmimages/AD13/98/C8/D84C482ABD00772A6D9ABA24B5D3._002.jpg" />
<Jpeg Page="3"
Url="http://dsmo.directsmile.de/dsmimages/AD13/98/C8/D84C482ABD00772A6D9ABA24B5D3._003.jpg" />
<Jpeg Page="4"
Url="http://dsmo.directsmile.de/dsmimages/AD13/98/C8/D84C482ABD00772A6D9ABA24B5D3._004.jpg" />
</DSMDocRenderStatus>
```

This is the XML Answer that we get when we call StartDoc or GetDocStatus. The included Informations are:

Parameters	Values	Description
State	(Numeric) = 0: Idle 1: Running 2: Failed 3: OK 4: OK with ErrorXML 5: Job Accepted and transferred to the Backend	The Status of the Document Rendering Process
LastProgress Message	(Text)	Last Status change Message.
LastProgressPercent	(Numeric) 0-100	The percentage of the Job that has been calculated.
LastErrorMsg	(Text)	The last Error Message. Please note that not all Errors will halt the calculation.
PDFUrl	(Text)	The URL to the calculated PDF File. Only available after the Job successfully completed.
Key	(Numeric)	The Job Identifier Key
Updated	(Date) mm/dd/yyyy hh:mm:ss AM/PM	Timestamp of when the job was last updated.
Created	(Date) mm/dd/yyyy hh:mm:ss AM/PM	Timestamp of when the job has been created.
LastErrorXMLUrl	(Text) URL	URL to the ErrorXML that holds further details about job errors.
Jpeg Section		If you used the Parameter CONVERTTO, the following sections refers to the calculated JPEG images of the Document.
Page	(Numeric)	The Page No. of the PDF that this JPG is referred to.
URL	(Text) URL	The URL to the JPEG.

We can use this information dynamically since it is updated each time we call the GetJobStatus Function.

<http://dsmo.directsmile.de/Examples/DocTest2.html>

To further address the Document rendering in DirectSmile Online, we need to have a look at two Data Structures, which are needed to handle Document Jobs.

The MsgComXML

At first I would like to explain the DirectSmile MsgComXML. This XML formatted data structure describes a Document rendering job. It is divided in 3 sections. The first section stores job parameters. These are used to define the Job as one would do it in the Print Dialog of the Generator. In fact, all job parameters that are supported by DirectSmile can be set. The second section stores the static field values, these define the variable document fields and the corresponding database fields.

The third section stores the actual Data that is used to personalise the Document. It can hold one record set of data, up to full tables with information to render a stack of documents.

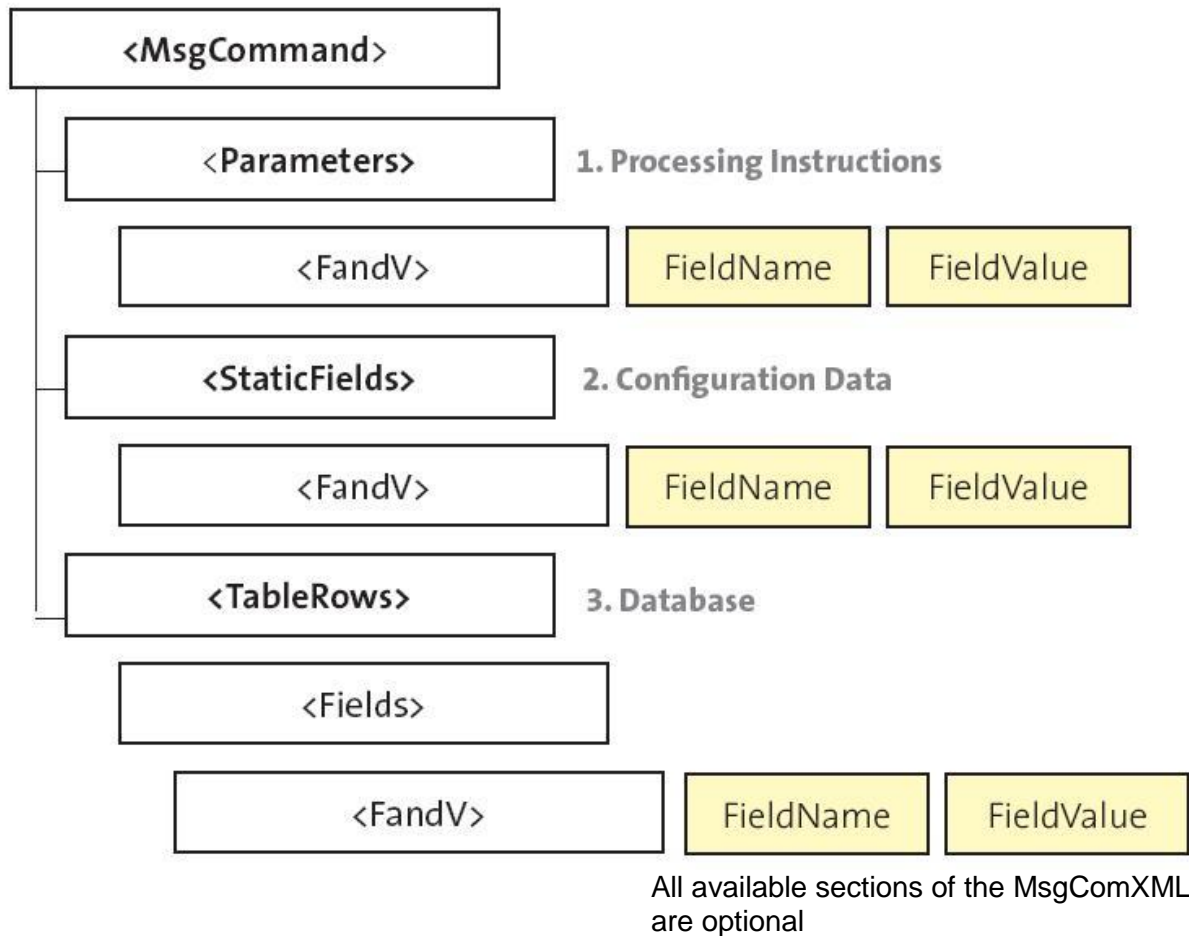
According to the DirectSmile Online WebServices, the MsgComXml is required as a String Variable in the StartDoc Function, to define the output. But it can as well be saved to a text file named .xmlprs and dropped into a DirectSmile Hotfolder, to render print Jobs.

The usual way to use this feature in DirectSmile is, to create it as a string and use it to create online Previews. Once the customer is satisfied and places an order, the Parameters in the MsgComXML can be adjusted to save them as XMLPRS file. This file can then be used with a DirectSmile Hotfolder to create the PrintReady Output of the Document.

It is important, to understand that an online preview is optimized for calculation speed and data size, to be able to quickly present an output to a customer, who is configuring his print product.

While once an order is placed, we need a high resolution output to feed the presses which in most cases is handled and calculated by a DirectSmile Production System to make sure, that the production side and the online preview rendering site do not interfere.

The MsgCommand Object Structure



The structure of the MsgComXML is build out of the Parameters Section, the Static Fields Section and the TableRows Section. Each section is optional and can be dropped. But most of the time you will at least need a Parameters and a StaticFields section. The Sections itself will be described on the following pages.

My First MsgComXML – An Easy Example

The example MsgComXML below shows how to create a preview PDF. We used it already in our examples in the Section “*Creating Documents using the DirectSmile Web Services*”.

All job relevant information is passed in the parameters section of the MsgComXML.

“OUTPUT” defines the output type and is set to “PreviewPDF”. For a Webservice Job, this will be sufficient in most cases.

(The FandV-Format (Field and Values) and all parameters are described on the next pages.)

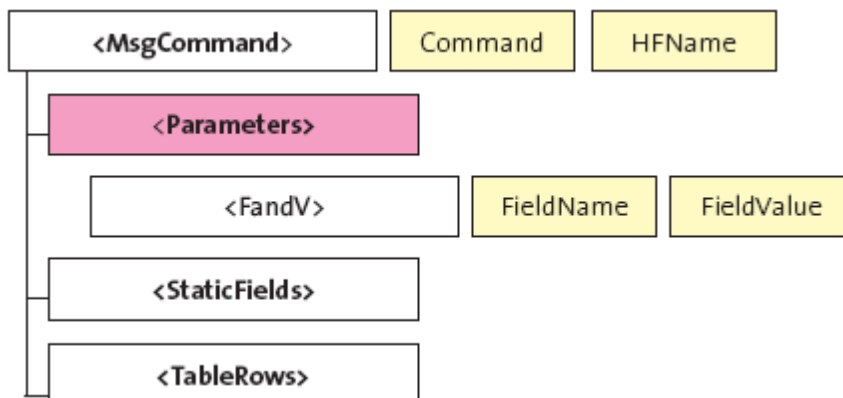
In the next section, the StaticFields you can pass content to the variable fields in the Document. In this example, we provide information for the variable fields in the CalendarV3 Document. These Fields are Document specific.

01NameinPicture	defines the Name to render in the Images on the Calendar.
StartMonth	defines the Month of the 1 st Calendar sheet.
mYear	defines the Year of the Calendar.
zHemisphere	defines the Hemisphere “North” or “South”.

Claim defines the Claim note in the Calendar footer.
Logo defines a picture to place on the Calendar as a Logo.

```
<?xml version="1.0" encoding="utf-8"?>
<COMMAND>
  <Parameters>
    <FandV FieldName="OUTPUT" FieldValue="PDFPreview"/>
  </Parameters>
  <StaticFields>
    <FandV FieldName="01NameInPicture" FieldValue="&quot;my first
Document&quot;"/>
    <FandV FieldName="StartMonth" FieldValue="&quot;1&quot;"/>
    <FandV FieldName="mYear" FieldValue="&quot;2009&quot;"/>
    <FandV FieldName="zHemisphere" FieldValue="&quot;North&quot;"/>
    <FandV FieldName="Claim" FieldValue="&quot;Printing
Emotions&quot;"/>
    <FandV FieldName="Logo"
FieldValue="&quot;DirectSmile_4c.eps&quot;"/>
  </StaticFields>
</COMMAND>
```

Data storage structures



All kinds of data are stored in Field-and-Value (FandV) objects. These are Very simple FieldName to FieldValue combinations. (FieldType doesn't matter here we are dealing with strings only in the connectivity server communication))

```
<FandV FieldName="Firstname" FieldValue="<Name>" />
```

A collection of **FandV** objects is called **FandVs**. In the example it's named "**StaticFields**" but It's still a **FandVs** collection

```
- <StaticFields>
  <FandV FieldName="Firstname" FieldValue="<Name>" />
  <FandV FieldName="Lastname" FieldValue="Marciniak" />
  <FandV FieldName="Adresse" FieldValue="Eigene Adresse" />
</StaticFields>
```

And finally you can provide a Table of variable Data if you want to render more than one Document in a job. In this case we add a <TableRows> Collection to the MsgCom XML. Like in this example, the

Firstname FandV above refers to a <TableRows> Field called “Name” (this is indicated by the brackets) the <TableRows> collection below represents the Information that will be used to Fill the Firstname Variable in the Documents.

```
- <TableRows>
  - <Fields>
    <FandV FieldName="Name" FieldValue="Erik" />
  </Fields>
  - <Fields>
    <FandV FieldName="Name" FieldValue="Peter" />
  </Fields>
</TableRows>
```

A MsgComXML for a Calendar Document with more than one Recordset (TableRow) would therefore look like this:

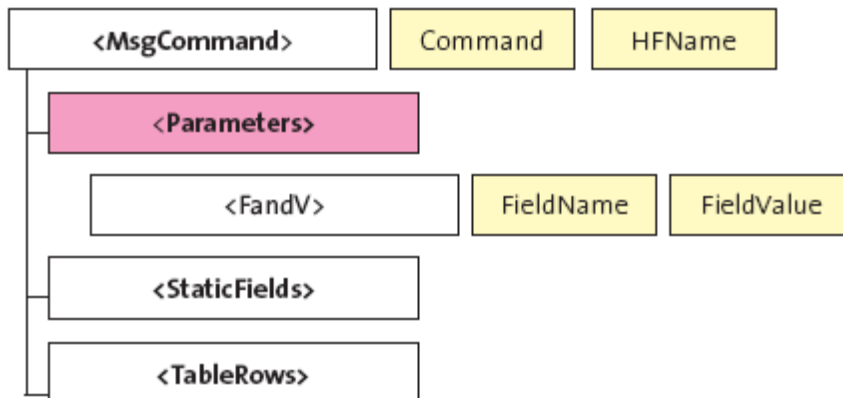
```
<?xml version="1.0" encoding="utf-8"?>
<COMMAND>
  <Parameters>
    <FandV FieldName="OUTPUT" FieldValue="PDFPreview"/>
  </Parameters>
  <StaticFields>
    <FandV FieldName="01NameInPicture" FieldValue="&lt;Name&gt;"/>
    <FandV FieldName="StartMonth" FieldValue="&quot;l&quot;"/>
    <FandV FieldName="mYear" FieldValue="&quot;2009&quot;"/>
    <FandV FieldName="zHemisphere" FieldValue="&quot;North&quot;"/>
    <FandV FieldName="Claim" FieldValue="&quot;Printing Emotions&quot;"/>
    <FandV FieldName="Logo" FieldValue="&quot;DirectSmile_4c.eps&quot;"/>
  </StaticFields>
  <TableRows>
    <Fields>
      <FandV FieldName="Name" FieldValue="Erik" />
    </Fields>
    <Fields>
      <FandV FieldName="Name" FieldValue="Peter" />
    </Fields>
  </TableRows>
</COMMAND>
```

!!! You might have noticed the encoding in the MsgComXML and the double quotes. It is important, to behold the UTF-8 encoding therefore we translate “<” to “<”, “>” to “>” and “”” to “"”.

It is also important, to understand the usage of Brackets and Quotas in the MsgComXML. Each Object in the collection is started with <ObjectName> and ends with </ObjectName> just like in a HTML Document. The FieldValues in the Static Fields Section are either included in quotas, to tell DirectSmile, that this is a String, or in “<>” brackets, to tell the System, that this Value refers to a Variable. In the example above, the Values are colored blue, to highlight this.

The Fields Collection can only hold String Values which thus does not need to be included in another quota to define its behavior.

<Parameters>



Parameters are the first section. Here you are able to transmit all kinds of processing instructions. All parameters are stored as FandV objects. Every left out parameter will be automatically replaced with the standard Value.

Supported MsgComXML Parameters in DirectSmile Online

Due to the fact, that the MsgComXML structure is not only used in the Online Interface, it includes all kinds of Parameters for the various Job Outputs. Since the focus of this Document is the DirectSmile Online Interface, I will explain those Parameters in detail, which are commonly used in online jobs.

Parameters to control Document Rendering...

<FandV FieldName="OUTPUT" FieldValue="PDFPreview"/>

The OUTPUT Parameter defines the output format of the rendered Document. For an Online Document, only PDF and PDFPREVIEW are used as OUTPUT Values. A Document that is created using the PDFPREVIEW option is calculated with standard parameters for Size and Quality to reduce the calculation time and the output size of the Document to values that can easily be transferred to the client PC. While a Document that is created using the PDF option, can be altered in size and quality using the corresponding parameters. The link to the generated PDF is returned in the DocumentRenderStatus after the completion of the Job.

Possible Options are:

PDFPREVIEW Generates a Document that is optimized in size and quality for Internet presentation.

PDF Generates a PDF Document according to the specified size and quality options.

<FandV FieldName="JPGCOMPRESSION" FieldValue="80"/>

The JPGCOMPRESSION Parameter defines the Compression of the Images that are included in the document. A value of 100 will result in no compression. The values 10-99 result in compressed images where 99 means almost no compression which will result in high quality images and 10 high compression and low quality. Compression will on the other hand reduce the size of the output tremendously.

<FandV FieldName="CONVERTTO" FieldValue="1"/>

The CONVERTTO Parameter, if set to 1 will convert the output PDF into a JPG File. If you use this parameter, you can define the width in Pixel of the JPG with the:

<FandV FieldName="OutputPagePixWidth" FieldValue="600"/>

This process then creates a PDF file and also stores each of the Pages (or the pages specified in the PAGERANGE) into JPGs. To retrieve the JPGs you need to alter the PDFUrl by cutting off the last 4 bytes and add “_nnn.jpg” where nnn is the page number (e.g. _001.jpg) the path can also be retrieved from the DocumentRenderStatus XML as explained in the “The DocStatus return Value” paragraph.

<FandV FieldName="PAGERANGE" FieldValue="1-10"/>

The PAGERANGE Parameter defines the Pages that are returned. It only works when CONVERTTO is set to 1. The Value of the Parameter can be provided in the PageFrom-PageTo form.

<FandV FieldName="IMPPREVIEW" FieldValue="1"/>

If set to 1, the IMPPREVIEW Parameter will result in an Imposition preview in the created PDF File.

<FandV FieldName="SUPPRESSETIMAGECROPPING" FieldValue="1"/>

If set to 1, the SUPRESSETIMAGECROPPING Parameter result in an output PDF that has no Image Cropping. This means that all pictures are rendered fully and not only the cropped part of the pictures. Surpresssetimagecropping can solve positioning issues that may occur on older RIPs, but it will slow down the ripping process tremendously and enlarge the output size of the returned File.

Please refer to the following example to try the Parameters for yourself:

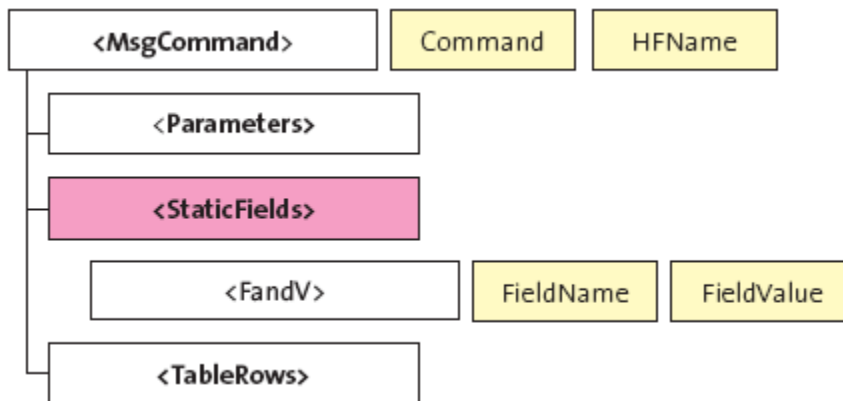
<http://dsmo.directsmile.de/Examples/MsgComXMLParam.html>

<FandV FieldName="WATERMARK" FieldValue="SYS_IMAGE"/>

If this Parameter is present in the MsgComXML and set to "" or "SYS_IMGE", a transparent GIF file named Watermark.gif, that is stored in the Generator subfolder System will be placed in the center of each document page.

Any other string value given in this Parameter will be shown as gray outline text in the center of each page.

<Static Fields>



Static Fields are directly passed to the DirectSmile Document. Each of it can contain either a String like "Paul" or a <TableRow> Data link like <Name>. If you like you can even transmit complex structures or VB-Scripts via this property.

FieldName contains the full name of a Variable in a Document. If the addressed variable is found in an embedded document then the full name is build up like this:

```
DocumentVariableName.VariableName
```

In this well known dot-logic you can address any variable of a DirectSmile document.

FieldValue can be:

a fixed string in quotations: "Peter"

a database column name in angle brackets: <Firstname>

an internal link to a variable of the DirectSmile Document

in brackets: <Name>

Or any VB-Script compatible term like :

```
"Dear" + <Name> + ", "
```

```
"$" + cStr(cLng(<Price>) * 0.90)
```

```
"Today is the " + now
```

<TableRows>

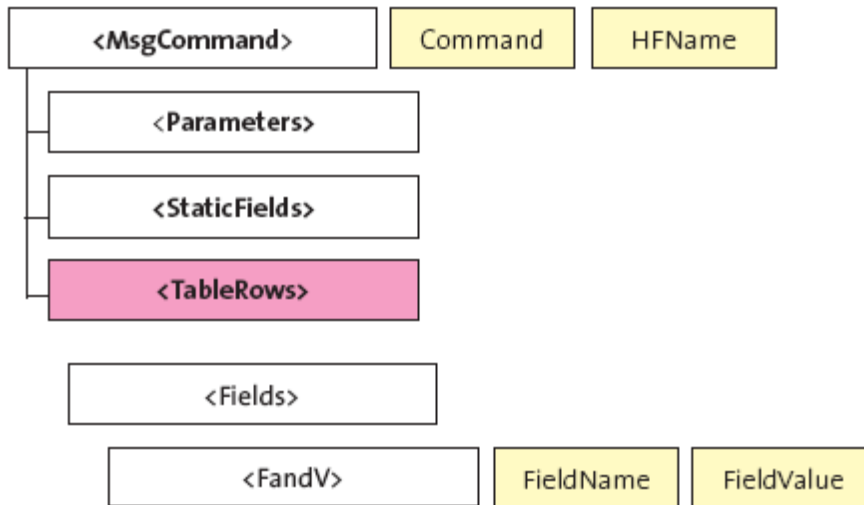
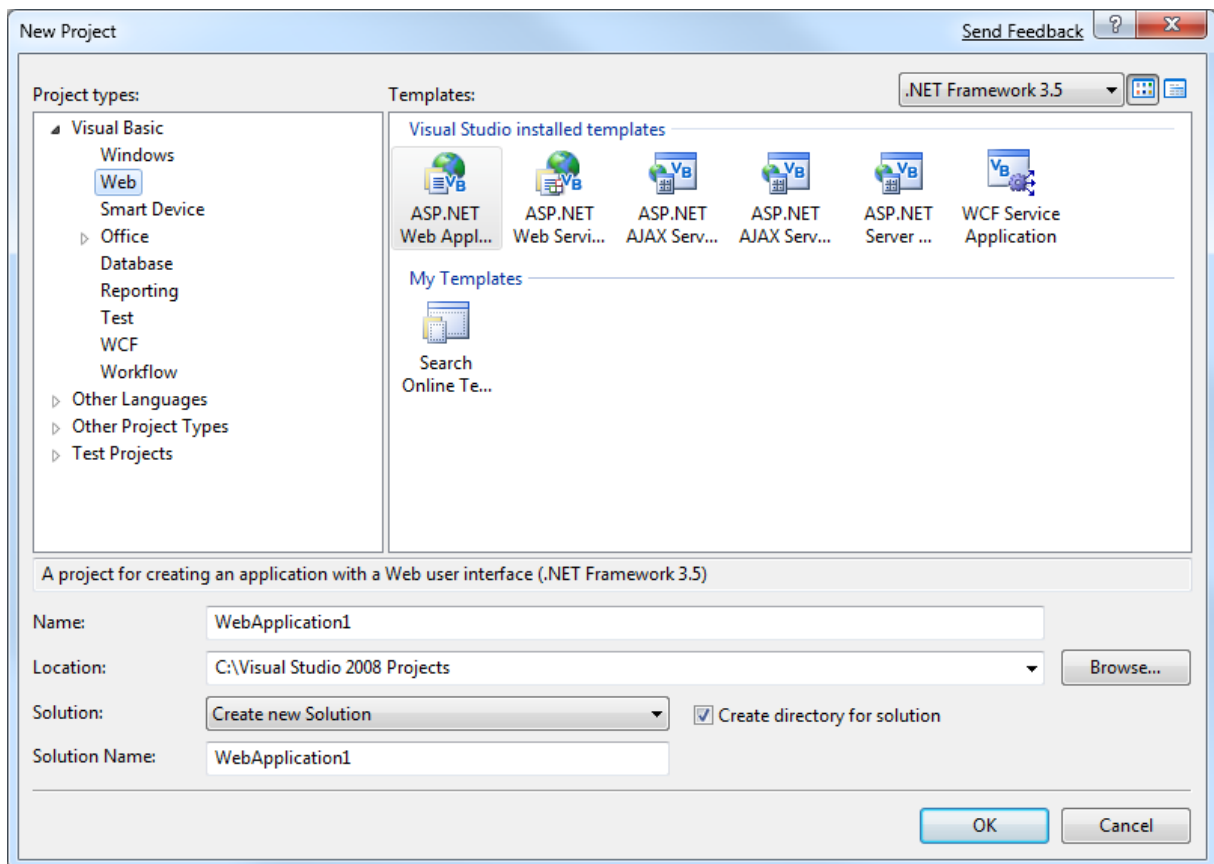


Table Rows are a simple way to define a database for the document rendering job.

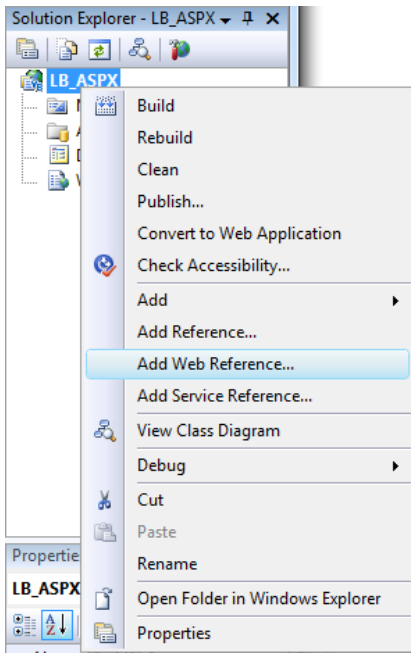
```
<TableRows>
  <Fields>
    <FandV FieldName="Name" FieldValue="Erik" />
    <FandV FieldName="Picture" FieldValue="c:\HPIM0121.JPG" />
  </Fields>
  <Fields>
    <FandV FieldName="Name" FieldValue="Peter" />
    <FandV FieldName="Picture" FieldValue="c:\HPIM01351.JPG" />
  </Fields>
</TableRows>
```

The DirectSmile Web Service in Microsoft .NET Applications and Active Server pages

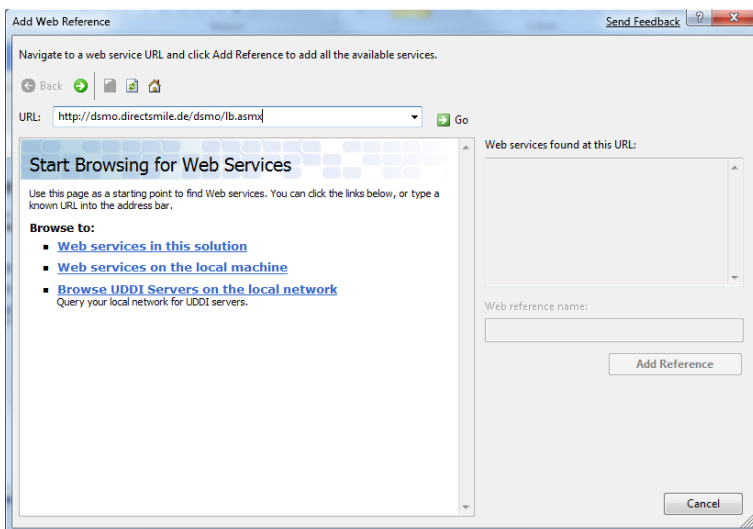
Using ASP.NET and Visual Studio 2008 is the easiest and most convenient way to use the WebService. All you have to do, is open an existing or a new asp.net Project:



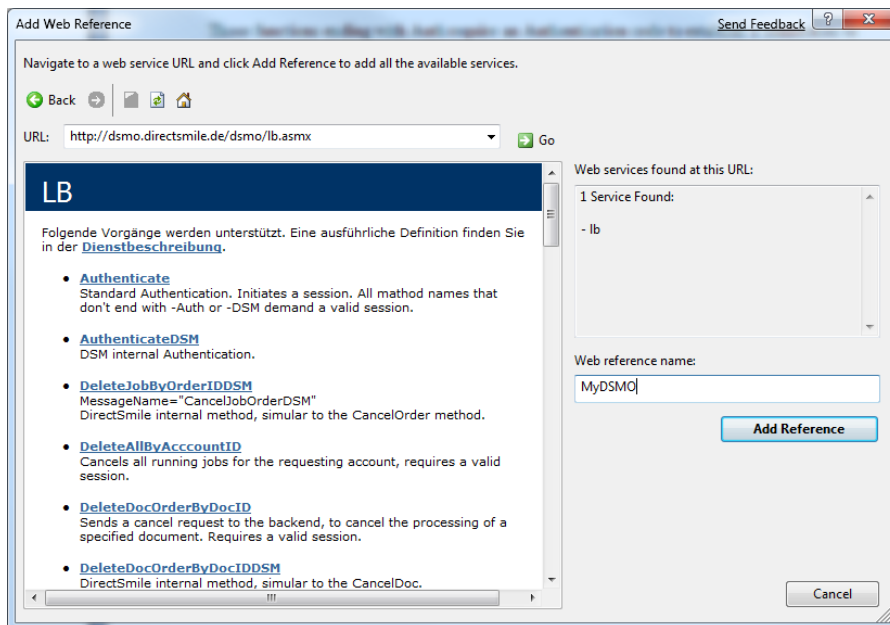
Once the Project is loaded, right click on the Project Entry in the Solution Explorer and choose “Add Web reference...”.



Now enter the URL to your Webservice (e.g. <http://dsmo.directsmile.de/dsmo/lb.asmx>) in the URL entry of the Form and choose “Web Services in this solution”.

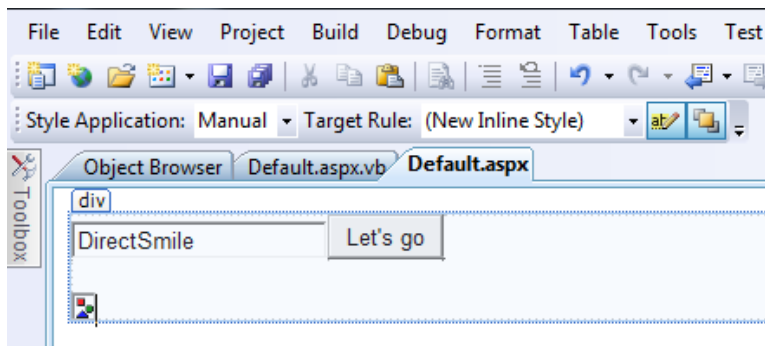


When the Web Service has been found, just choose a Name for the Reference and click “Add Reference”.



Now you can use the Web Service in your Project.

To create an easy Example page, just add a Textfield, a Button and an Image Container to your Web Page.



In the Visual Basic .aspx.vb file we create a Function that will be invoked by the Btn_Click event.

```
Private Sub BtnLG_Click(ByVal sender As Object, ByVal e As System.EventArgs) Handles
BtnLG.Click
    Dim imgUrl As String
    'Authentication Code of the Help user to logon to the DirectSmile System
    Dim AuthCode As String = "BDF4A2A0956202CE6119B5EBBD8A66D"
    'Reference to the Webservice
    Dim MyLB As New MyDSMO.LB

    'We retrieve an Image URL by invoking the GetImageUrlAuth function
    imgUrl = MyLB.GetImageUrlAuth(AuthCode, "preview_dsmdaisymeadow", Me.txtName.Text,
    640, "", 80, 1)

    'Setting the imageUrl Property of our Image Container to show the DirectSmile Image
    Me.ImgResult.imageUrl = imgUrl
End Sub
```

All we do here is we invoke the GetImageUrlAuth() function to retrieve an image URL and use this in the Image Container on the asp page to show the rendered Image. And this is what the result looks like:

ASPx & Webservice



A closer look at the .NET and more examples are coming soon.

Appendix A the DirectSmile WebService Functions in detail:

The DirectSmile Online WebService comes with a rich set of Functions, to address all kinds of needs for WebDevelopers and Personalisation Production. The following will describe the Functions in detail. To access a list of the supported function, just load the WebService site on your DirectSmile Online Server <http://myDSMOServer/DSMO/lb.asmx>.

If you do not yet run your own DirectSmile Online System, you can call the site on the DirectSmile Server:

<http://dsmo.directsmile.de/dsmo/lb.asmx>

However, calling this page from the localhost has the advantage, that one can try the provided functions directly from this page.

A WSDL description of the Service is available using the ?WSDL Parameter:

<http://dsmo.directsmile.de/dsmo/lb.asmx?WSDL>

Those functions, ending with DSM are special functions that are used from DirectSmile Online internally and thus not part of this description.

Those functions ending with Auth require an Authentication code to establish a connection to the DirectSmile Online Server. You can either use the authenticate function once and get a valid session ID to communicate with the server, or you use these functions which will authenticate whenever you call them using the provided Authentication code.

- [**Authenticate**](#)

Standard Authentication. Initiates a session and returns a SessionID. All method names that don't end with -Auth or demand a valid session.

Parameters:

Username A valid Username to login to DirectSmile Online

Password A valid Password to login to DirectSmile Online

Return Value:

SessionID A String that defines a valid session with the DirectSmile Online Server. The SessionID can then be used to invoke other functions using this session.

- [**DeleteAllByAccountID**](#)

Cancels all running jobs for the requesting account, requires a valid session.

Parameters:

SessionID A valid SessionID (see Authenticate)

Return Value:

Success A string containing either "OK" or an error message.

- [DeleteDocOrderByDocID](#)

Sends a cancel request to the backend, to cancel the processing of a specified document. Requires a valid session.

Parameters:

SessionID A valid SessionID (see Authenticate)
Key Key string that identifies a Document rendering Job

Return Value:

Success A string containing either “OK” or an error message.

- [DeleteJobByOrderID](#)

Cancels an image order, requires a valid session.

Parameters:

SessionID A valid SessionID (see Authenticate)
JobId JobID string that identifies an Image rendering Job

Return Value:

Success A string containing either “OK” or an error message.

- [GetAuthenticationCode](#)

Creates and returns an unique AuthenticationCode for a specific DirectSmile Online user. The returned Code can be passed to each method that method name ends with -Auth. You can call these methods directly without calling the Authenticate method upfront.

Parameters:

Username A valid DirectSmileOnline Username
Password The corresponding Password

Return Value:

AuthenticationCode An authenticationcode to logon to the DirectSmileOnline Server.

- [GetAvailableDhttList](#)

Generates a commaseperated list of available LandingPage templates. This function is deprecated, please use the GetAvailableDhttListXML instead. The SessionID is included to stay backward compatible, it is not used in the method.

Parameters:

SessionID This SessionID Parameter is no longer in use and may stay blank.

Return Value:

dHttList A colon separated string containing dHtt aliases.

- [GetAvailableDhttListAuth](#)

Method is similar to the GetAvailableDhttListAuth, but You do not need to call the Authenticate method before you call this Method. Please pass only the AuthenticationCode. You can get an AuthenticationCode by calling the GetAuthenticationCode Method.

Parameters:

AuthenticationCode An authenticationcode to logon to the DirectSmileOnline Server. It can be retrieved by using the GetAuthenticationCode Method.

Return Value:

dHttList A colon separated string containing dHtt aliases.

- [GetAvailableDocList](#)

Returns a xml based list of available documents, including urls to thumbnails and datainterfaces.

Parameters:

SessionID A valid SessionID (see Authenticate)

Return Value:

DocListXML A XML structure listing all available documents.

XML format:

```
<?xml version="1.0" encoding="utf-8" ?>
<string xmlns="http://www.directsmile.de/GenService/LB">
<?xml version="1.0" encoding="utf-8"?>
<DSMDocumentList>
<Document Alias=" calendarv3"
DisplayName=" CalendarV3"
Preview=http://dsmo.directsmile.de/DSMO/ThumbNail.aspx?ID=1&Type=Doc
DataInterfaceUrl=http://dsmo.directsmile.de/DSMO/users/help/DataInterfaces/CalendarV3/Datainte
rface.xml
/>
</DSMDocumentList>
</string>
```

- [GetAvailableDocListAuth](#)

Returns a xml based list of available documents, including urls to thumbnails and datainterfaces. Requires authenticationcode

Parameters:

AuthenticationCode An authenticationcode to logon to the DirectSmileOnline Server. It can be retrieved by using the GetAuthenticationCode Method.

Return Value:

DocListXML A XML structure listing all available documents.

XML format:

```
<?xml version="1.0" encoding="utf-8" ?>
<string xmlns="http://www.directsmile.de/GenService/LB">
<?xml version="1.0" encoding="utf-8"?>
<DSMDocumentList>
<Document Alias=" calendarv3"
DisplayName=" CalendarV3"
Preview=http://dsmo.directsmile.de/DSMO/ThumbNail.aspx?ID=1&Type=Doc
DataInterfaceUrl=http://dsmo.directsmile.de/DSMO/users/help/DataInterfaces/CalendarV3/Datainte
rface.xml
/>
</DSMDocumentList>
</string>
```

- [GetAvailableSetList](#)

Returns a comma separated list of available Sets. This function is deprecated, please use the GetAvailableSetListXML instead. Requires a valid session.

Parameters:

SessionID A valid SessionID (see Authenticate)

Return Value:

SetList A colon separated string containing Set aliases.

- [GetAvailableSetListAuth](#)

Method is similar to the GetAvailableSetList, requires an authenticationcode.

Parameters:

AuthenticationCode An authenticationcode to logon to the DirectSmileOnline Server. It can be retrieved by using the GetAuthenticationCode Method.

Return Value:

SetList A colon separated string containing Set aliases.

- [GetAvailableSetListXML](#)

Generates a xml based list of available DirectSmile Sets, requires a valid session.

Parameters:

SessionID A valid SessionID (see Authenticate)

Return Value:

SetListXML A XML formatted string containing Set details.

XML format:

```
<string xmlns="http://www.directsmile.de/GenService/LB">
  <?xml version="1.0" encoding="utf-8"?>
  <DSMSetList>
    <Set
      ID="30"
      Alias="artinstitute_tall"
      Name="ArtInstitute_Tall.dSet"
      Height="675"
      Width="1050"
      Thumbnail=http://dsmo.directsmile.de/DSMO/Thumbnail.aspx?ID=30&Type=Set
    />
  </DSMSetList></string>
```

- [GetDocAttributes](#)

Returns extended attributes of a DirectSmile Document.

Parameters:

SID A valid SessionID (see Authenticate)

DocumentAlias A valid DocumentAlias to identify the Document for which the information is returned.

- [GetDocDataInterfaceURL](#)

Returns an URL to the Interface description of a DirectSmile Document. Requires a valid session.

Parameters:

SessionID A valid SessionID (see Authenticate)

Return Value:

DataInterfaceURL A URL that is pointing to the Document Datainterface

- [GetDocStatus](#)

Returns a serialized DocStatusReport object, which contains progress information of the processing of document request. Requires a valid session.

Parameters:

SessionID A valid SessionID (see Authenticate)

Key Key string that identifies a Document rendering Job

Return Value:

DocStatusReport A XML formatted string containing the current Document render status

XML format:

- [GetImage](#)

Returns a byte array of an requested image. This is a synchronous call.

Parameters:

SessionID	A valid SessionID (see Authenticate)
SetAlias	A valid SetAlias that defines the DirectSmile Set to render.
TextToRender	Some Text to render into the Image
PixelWidth	The width of the output image in Pixels
Quality	The Jpeg Compression of the output image 100 = no compression.
Key	The Key value is no longer used and is ignored by the method.
WaterMarkType	0 = no watermark / 1 = DirectSmile Watermark / 2 = Preview
Watermark	

- [GetImageUrl](#)

Returns an URL to a rendered image. This is a synchronous call.

Parameters:

SessionID	A valid SessionID (see Authenticate)
SetAlias	A valid SetAlias that defines the DirectSmile Set to render.
TextToRender	Some Text to render into the Image
PixelWidth	The width of the output image in Pixels
Quality	The Jpeg Compression of the output image 100 = no compression.
Key	The Key value is no longer used and is ignored by the method.
WaterMarkType	0 = no watermark / 1 = DirectSmile Watermark / 2 = Preview
Watermark	

- [GetImageUrlAuth](#)

Method is similar to the GetImageUrl, but You do not need to call the Authenticate method before you call this Method. Please pass only the AuthenticationCode. You can get an AuthenticationCode by calling the GetAuthenticationCode Method.

Parameters:

AuthenticationCode	An authenticationcode to logon to the DirectSmileOnline Server. It can be retrieved by using the GetAuthenticationCode Method.
SetAlias	A valid SetAlias that defines the DirectSmile Set to render.
TextToRender	Some Text to render into the Image
PixelWidth	The width of the output image in Pixels
Quality	The Jpeg Compression of the output image 100 = no compression.
Key	The Key value is no longer used and is ignored by the method.
WaterMarkType	0 = no watermark / 1 = DirectSmile Watermark / 2 = Preview
Watermark	

Return Value:

ImageURL	A URL pointing to the rendered Image
----------	--------------------------------------

- [GetInterfaceVersion](#)

Returns the version of the DSMO webservice interface in Major.Minor.Revision format. This Method does not use any Paramters.

- [GetJobStatus](#)

Returns the status of a backend JobOrder, requires a valid session.

Parameters:

SessionID A valid SessionID (see Authenticate)
JobId JobID string that identifies an Image rendering Job

Return Value:

JobStatus Current Image rendering JobStatus

- [GetJobStatusAuth](#)

Returns the status of a backend JobOrder, requires an authenticationcode.

Parameters:

AuthenticationCode An authenticationcode to logon to the DirectSmileOnline Server. It can be retrieved by using the GetAuthenticationCode Method.
JobId JobID string that identifies an Image rendering Job

Return Value:

JobStatus Current Image rendering JobStatus

- [GetdHttpUrl](#)

Returns an URL to Html Document Template. The SessionID parameter is included to stay backward compatible, it is not used in the method.

Parameters:

SID A valid SessionID (see Authenticate)
dHttpAlias A valid dHttp Alias to identify the template to use.

Return Value:

dHttpURL An URL pointing to the Html Document Template.

- [PlaceOrder](#)

Send a JobOrder XML to the Backend, requires a valid session

Parameters:

SessionID A valid SessionID (see Authenticate)
ImagesXML A XML String that identifies the Job to render.

- [PlaceOrderAuth](#)

Send a JobOrder XML to the Backend, requires authenticationcode

Parameters:

AuthenticationCode An authenticationcode to logon to the DirectSmileOnline Server. It can be retrieved by using the GetAuthenticationCode Method.

ImagesXML A XML String that identifies the Job to render.

- [RestartJob](#)

Restarts a job stored in the database, requires a valid session.

Parameters:

Sid A valid SessionID (see Authenticate)

GUID A GUID is the identifier of a Job in the DirectSmile Database. These Identifiers are stored for 30 minutes after a job is done to be able to rerender the Job. After this period the GUID will be dropped from the Database table.

- [SendEmailTemplate](#)

Sends an Email containing a Template or html content. The SessionID parameter is included to stay backward compatible, it is not used in the method.

SessionID A valid SessionID (see Authenticate)

Recipients A List of Recipients for the E-Mail Template

ReplyTo A valid Reply Address for the E-Mail header

Subject A valid subject for the E-Mail header

TemplateAlias A valid dHtt Alias to identify the template to use.

MsgCom A XML structure to define the JobParameters

Html A String containing the HTML Page to use in the E-Mail

DocumentAlias A valid DocumentAlias to identify the Document for which the information is returned.

- [StartDoc](#)

Sends a document rendering request to the Backend. This an asynchronous call. The method returns a serialized DocumentStatusReport object.

SessionID A valid SessionID (see Authenticate)

DocumentAlias A valid DocumentAlias to identify the Document for which the information is returned.

MsgComXML A XML structured string that defines a Document Rendering Job. Please refer the the MsgComXML Paragraph of this documentation.