



DOCUMENTS 4

## SOAP Interface Documentation

WSDL DOCUMENTS-4.0.1870

© Copyright 2014 otrs software AG. All rights reserved.

No part of this publication may be reproduced or transmitted in any form or by any means without express written permission of otrs software AG. Any information contained in this publication is subject to change without notice.

All product names and logos contained in this publication are the property of their respective manufacturers.

EASY reserves the right to make changes to this software. The information contained in this manual in no way obligates the vendor.

## Table of Contents

<b>1. The DOCUMENTS SOAP Web Service .....</b>	<b>4</b>
<b>2. From Service to Project .....</b>	<b>9</b>
2.1 WSDL.....	9
2.2 WSDL and programming language .....	10
2.3 WSDL tool and SOAP session.....	12
2.4 Documents WSDL Version.....	13
2.5 Working in a Microsoft® Visual Studio 2005® project.....	13
2.6 Sample code console application VB.....	14
2.7 Creating a gSOAP Client with Visual C++ 2008® .....	16
2.8 Creating a WCF Client with Visual C# 2008® .....	21
<b>3. API.....</b>	<b>26</b>
3.1 DOCUMENTS methods .....	26
3.2 DOCUMENTS classes.....	68
3.3 Other functions.....	77
<b>4. HTTPS support (SSL/TLS).....</b>	<b>79</b>
4.1 Configurations in the docsoapprosy.ini.....	79
4.2 Creating a self-signed X.509 certificate .....	80
4.3 HTTPS/SSL capable gSOAP client with Visual C++ 2008® .....	81
4.4 Further HTTPS/SSL capable clients .....	82
<b>5. Appendix .....</b>	<b>84</b>
5.1 Archive file key .....	84
5.1.1 EE.i: .....	84
5.1.2 EE.x: .....	85
5.1.3 EAS: .....	87
5.2 Search resource id for an archive .....	88
5.3 Destination identifier for archive .....	88
5.4 Field labels for reading file attributes in reports/search.....	88
5.5 Label for personal default folders .....	89
5.6 Syntax description for filter expressions .....	89
5.7 Schema for getFilingPlanXML output .....	90
5.8 Client timeouts .....	91
<b>6. Index .....</b>	<b>93</b>
<b>7. Table of Figures.....</b>	<b>98</b>

## 1. The DOCUMENTS SOAP Web Service

DOCUMENTS provides a Web service that allows third-party applications to access DOCUMENTS files via a SOAP interface, and edit these.

### Function scope

Currently, the following operations are available at different object levels:

#### *User*

- Log on to/off the DOCUMENTS proxy.
- Log on a user under a different account.
- Determine the current user of the interface.
- Read user properties

#### *Folder*

- Read user's Inbox folder.
- Read user's Sent folder.
- Determine folders and subfolders (for default folders).
- Read public folders.

#### *File types*

- Determine file type.
- Read file type information on:
- Fields, archives, document tabs, workflows

#### *Files*

- Create, edit and delete.
- Send files to new recipients (parallel for information, sequential).
- Add, read and delete documents.
- Read tasks.
- Read monitor.
- Archive DOCUMENTS files.
- Set resubmission date
- Read properties and auto texts

#### *Search*

- Perform searches in file types and archives.

#### *Workflow*

- Determine workflows.
- Start and close workflows for DOCUMENTS files.
- Read and run user actions for DOCUMENTS files.
- Start scripts on DOCUMENTS files.

#### *Filing plans*

- Determine filing plans.
- Read filing plans in XML representation.

#### WSDL

- WSDL download.

#### Web service and proxy

The DOCUMENTS web service is not fully integrated in the DOCUMENTS server. It is made available by a special proxy server ("DOCUMENTS-Proxy"). The programmed application is a client of the proxy, which in turn operates as a client of the DOCUMENTS server. The proxy transforms incoming SOAP-requests to system-native function calls. The proxy also assists the server with the session-management of SOAP clients. To enable the application to reach the proxy, you may need to customize the address under which the communication takes place (see below URL ).

To be able to use this API, the following requirements must be met:

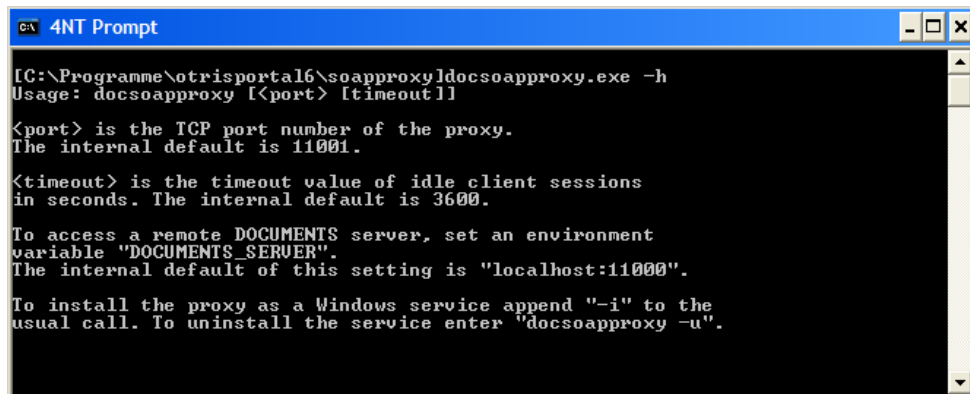
1. An installed DOCUMENTS server
2. A license file, which includes the DOCUMENTS-API license. With a differing license the web service may be limited to a small set of functions, or it may not operate at all.
3. Installing the [DOCUMENTS](#) proxy

As of [ELC 3.60](#) or [otris Portal 6.0](#), this comes as part of the [DOCUMENTS](#) installation, residing in the `soaproxy` subdirectory.

The *Soaproxy*'s installation directory also contains an actual service description file (WSDL file). This is the significant WSDL for the proxy and for client programming.

Calling the proxy by command line with the `-h` option results in a listing of further possible command line parameters.

Specifications of the port and of the timeout period can be made here. Also a (de-)installation as a Windows service is possible directly via the command line. The connection parameters of the DOCUMENTS server can be determined by an environment variable.



```
C:\ 4NT Prompt

[ C:\Programme\otrisportal6\soaproxy\docsoaproxy.exe -h
Usage: docsoaproxy [ <port> [timeout] ]

<port> is the TCP port number of the proxy.
The internal default is 11001.

<timeout> is the timeout value of idle client sessions
in seconds. The internal default is 3600.

To access a remote DOCUMENTS server, set an environment
variable "DOCUMENTS_SERVER".
The internal default of this setting is "localhost:11000".

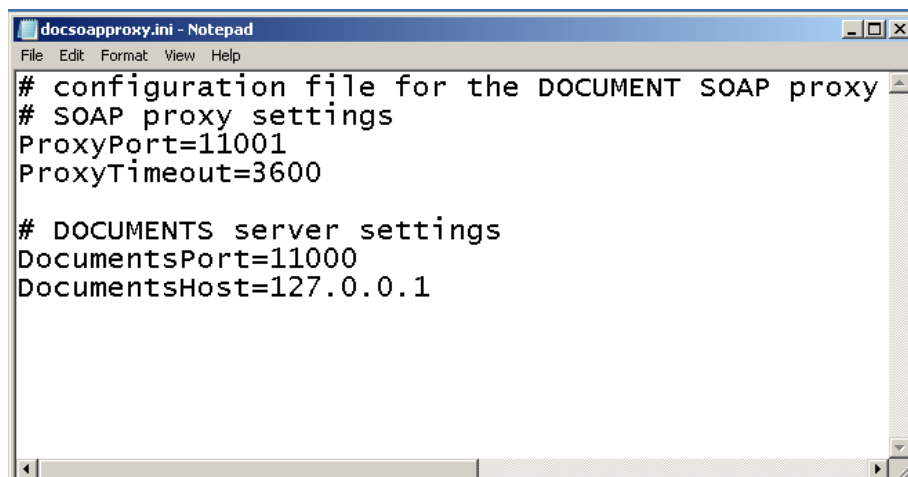
To install the proxy as a Windows service append "-i" to the
usual call. To uninstall the service enter "docsoaproxy -u".
```

*Fig. 1: Command line call docsoaproxy -h*

```
docsoaproxy 10123 3000
```

#### *Call via Port 10123 and timeout 3000*

The proxy configuration can also be controlled with an ini-file which must reside in the same directory as the proxy itself:



```
docsoaproxy.ini - Notepad
File Edit Format View Help

# configuration file for the DOCUMENT SOAP proxy
# SOAP proxy settings
ProxyPort=11001
ProxyTimeout=3600

# DOCUMENTS server settings
DocumentsPort=11000
DocumentsHost=127.0.0.1
```

*Fig. 2: docsoaproxy.ini as configuration file*

The settings in the configuration file overwrite their counterparts in the command line or in an environment variable.

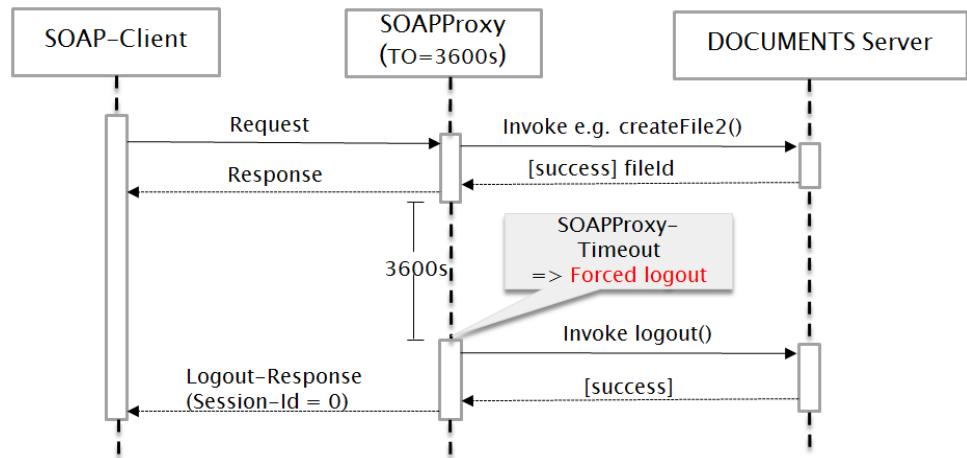
- ProxyPort:  
The port on which the proxy accepts the client requests. See also under: URL .

```
Default: 11001
```

- ProxyTimeout:
- The number of seconds after which a session expires, if it is not continued or closed via logout.

```
Default: 300
```

The sequence diagram from Fig. 3 shows the occurrence of a proxy timeout after 3600s (see also Client timeout).



*Fig. 3: Sequence diagram of a SOAP proxy timeout*

- DocumentsPort:

The port on which the DOCUMENTS server accepts the proxy requests.

Default: 11000

- DocumentsHost:

The host name on which the DOCUMENTS server was started.

Default: localhost

- SequenceSessionId:

As of version ELC 3.60g/otirs Portal 6.0g, the constantly changing allocation of *SessionIds* can be suppressed by the setting `SequenceSessionId=0` (see WSDL tool and SOAP session).

- `PortalServerEncoding`:  
Version [ELC 3.60g](#)/[otris Portal 6.0g](#) allows defining for the proxy the encoding containing the server data.  
Unless defined otherwise, the *system encoding* is used, and it is displayed on startup of the proxy, e.g.: `English_UnitedStates.1252`.  
For Windows, the encoding is built in this format:

`[Language]_[Country].[CodePage]`

For more information on this, see [Language Identifiers and Locales](#) in Microsoft® MSDN®.

When using an UTF-8 portal server, you need to set the following value regardless of country and language:

`PortalServerEncoding=UTF-8`

Be aware that, when using an encoding different from the system encoding, the SOAP client must consider this and customize its own encoding, if necessary. This also applies to UTF-8 encoding. See Fig. 6 below.

- `BlobBasePath`:  
Since WSDL DOCUMENTS-4.0.1870 it is possible to upload Blobs from the file system directly. This option specifies a shared directory, which contains the Blobs and which can be accessed by the SOAP proxy. The SOAP client does no longer need to pass the blob content encoded in base64 format. It can pass a file path instead, which is relative to the specified shared directory. From there the SOAP proxy takes the file and uploads it to the DOCUMENTS server.
- `SSL`:  
`SSL=1` enables SSL or TLS encrypted communication between clients and the SOAPProxy. This option is available since WSDL DOCUMENTS-4.0.1870.
- `keyfile`:  
Path to a certificate/key file for the SSL encryption. The path can be either absolute or relative to the installation path. The pem-file contains at least the private RSA key and the certificate of SOAP proxy. Intermediate certificates of a certificate chain may follow, if necessary.
- `keypasswd`:  
Password to read the private key in the key file. The password has to be specified in plain text.

As of version [ELC 3.60g](#) or [otris Portal 6.0g](#), an additional version of the proxy named *docsoaproxy\_log.exe* is included in the setup, which logs SOAP communication. This proxy makes debugging easier at communication level. See also section [WSDL tool and SOAP session](#).



## 2. From Service to Project

### 2.1 WSDL

The offered Web service is defined via the [Web Service Definition Language \(WSDL\)](#), which as metalanguage in XML format specifies exchange protocols that can be used to make server methods accessible from the outside.

```
<!-- operation request element -->
  <element name="getFileTypes">
    <complexType>
      <sequence>
        <element name="ignoreRight" type="xsd:boolean"
minOccurs="1" maxOccurs="1"/>
      </sequence>
    </complexType>
  </element>

<!-- operation response element -->
  <element name="getFileTypesResponse">
    <complexType>
      <sequence>
        <element name="filetype"
type="DOCUMENTS:FileTypeShortDescr" minOccurs="0"
maxOccurs="unbounded"/>
      </sequence>
    </complexType>
  </element>
```

#### *Excerpt from the WSDL for the "getFileTypes" method*

The above WSDL excerpt describes how to initiate a request to the server. You need to pass exactly one "ignoreRight" parameter to the "getFileTypes" method, where the server sends an indeterminate number of DOCUMENTS:FileTypeShortDescr objects as a response. However well-defined and precise the WSDL may be, it does not allow direct programming, or makes it very difficult.

## 2.2 WSDL and programming language

The presence of a service description in WSDL format is therefore only the first step in programming. To be able to use the deployed service, the matching structures and classes for the respective programming language and IDE must be generated from the WSDL. The current WSDL is deployed in the `soaproxy` directory. It can also be loaded via <http://soaproxy:11001/?wsdl> for an active proxy (see WSDL) and via <https://soaproxy:11001/?wsdl> for SSL support since WSDL DOCUMENTS-4.0.1870 respectively.

In Microsoft® Visual Studio 2005® this generation is performed by the "wsdl.exe" program, which enables generation in the target programming language via the `/language` option. Starting with Microsoft® Visual Studio® 2008 you find the file stored in the Windows SDK directory (`%WindowsSdkDir%` eg. `C:\Program Files\Microsoft SDKs\Windows\V6.0A\bin`).

```
"C:\Program Files\Microsoft Visual Studio 8\SDK\v2.0\Bin
\wsdl" /nologo /language:VB /out:.\DOCUMENTS.vb
/protocol:SOAP .\DOCUMENTS.wsdl
```

*Call to generate the "DOCUMENTS.vb" in Visual Basic from the DOS shell in the WSDL directory – `genWrapper.cmd`*

The "DOCUMENTS.vb" now allows both generating interface objects and calling object methods in Visual Basic using the classes derived from the WSDL.

Owing to the existing dependency, changes to the WSDL imply changes in the generated code of the "DOCUMENTS.vb".

You need to be aware here that, depending on the programming language and WSDL tool used, the generated code may turn out differently. Specific generation tools are used to provide object accumulations in an array, while other tools manage these in a separate list. Some tools are not aware of a distinguished `ReturnValue`, while others automatically generate the first output parameter as the return value of the function.

Of course, the offered data types are also dependent on the programming language. What has been defined as "String" or "Integer" in the WSDL is usually also offered in the destination language as "String" or "Integer". But this may be entirely different with a "ByteArray" in Base64 encoding. In the case of deviations in data types and signatures, the WSDL and the `DOCUMENTS` data file generated from it should be compared with each other.

Thus, for the above WSDL sample "*getFileTypes*", the following implementation results for the generated `DOCUMENTS.vb`:

```
Public Function getFileTypes(...ByVal ignoreRight As Boolean)>
    As FileTypeShortDescr()
        Dim results() As Object = Me.Invoke("getFileTypes",
        New Object() {ignoreRight})
        Return CType(results(0), FileTypeShortDescr())
    End Function
```

*Excerpt from the Documents.vb "getFileTypes" – Some attributes were omitted for the sake of clarity.*

Implementing this method is, however, not the only important issue here; calling it from the program, which is really quite trivial, is equally important.

```
Dim doc as New DOCUMENTS

Dim ftsd() As FileTypeShortDescr
ftsd = doc.getFileTypes(False)
```

*Calling the Documents.vb "getFileTypes" method in a program*

As illustrated in the sample, the `FileTypeShortDescr` class is generated including all its members defined in the WSDL.

For programming, therefore, this means convenient working on the basis of native classes and methods. This eliminates the need for directly working at the **SOAP** level below.

If the **DOCUMENTS** proxy is not running on the computer where the application is developed, you must - after instantiating the **DOCUMENTS** object, specify the URL on which the proxy accepts the requests of the programmed application.

The default setting from the WSDL "`http://localhost:11001`" applies when proxy and application are running on the same local computer.

The URL is composed of the computer name under which it can be reached on the net or its IP address, and the port.

```
Dim doc as New DOCUMENTS

Set 'proxy address
doc.Url ="http://proxyComputerOrIpAddress:11001"
```

*URL under which the DOCUMENTS proxy can be reached by the application.*

## 2.3 WSDL tool and SOAP session

Communication between client and proxy is performed via the SOAP protocol. In doing so, a unique `SessionId` that is threaded in the SOAP header ensuring that the proxy returns the responses to the correct client is set for each individual connection.

The communication process appears as follows:

```
Client A --> SessionId 0 --> Proxy
Client A logs in to the proxy using login

Proxy -->SessionId 8186860391163 --> Client A
and is returned a SessionId in the SOAP header by the proxy

Client A -->SessionId 8186860391163 -->Proxy
Client A sends another request to the proxy, including the
previously contained SessionId in the SOAP header.

Proxy -->SessionId 8186860397652 -->Client A
Proxy identifies the request to be from client A; it
responds to it, also sending a new (!) SessionId.
```

### *Session handling via SOAP*

For each new request of a client, the `SessionId` previously contained in the response must also be sent. Only this string of `SessionIds` indicates and identifies exactly one (1) session between client, proxy, and the **DOCUMENTS server**. In doing so, the proxy can return a new `SessionId` on responding to the request. But it does not necessarily have to change!

This mechanism is primarily used to prevent session hijacking, where a user draws the session of an already authenticated user. Below login is performed and the `Userinfo` function is started multiple times in succession before `logout` occurs:



```
file:///C:/Dokumente und Einstellungen/dotnetsoap/Eigene Dateien/Visual Studio 2005/Proj...
Session
Login SessionId: 81868603911635
Current Session: 0 - 81868603911635
Userinfo Name: Schreiber, Willi
Current Session: 1 - 8186860397652
Userinfo Name: Schreiber, Willi
Current Session: 2 - 8186860397805
Userinfo Name: Schreiber, Willi
Current Session: 3 - 8186860397805
Userinfo Name: Schreiber, Willi
Current Session: 4 - 8186860397854
Userinfo Name: Schreiber, Willi
Current Session: 5 - 8186860397907
Userinfo Name: Schreiber, Willi
Current Session: 6 - 8186860397959
Userinfo Name: Schreiber, Willi
Current Session: 7 - 8186860398008
Userinfo Name: Schreiber, Willi
Current Session: 8 - 8186860398060
Userinfo Name: Schreiber, Willi
Current Session: 9 - 8186860398112
Userinfo Name: Schreiber, Willi
Current Session: 10 - 8186860398161
Userinfo Name: Schreiber, Willi
END - Session
```

Fig. 4: Changing the SessionIds as part of login/logout

Depending on the WSDL tool used, the `SessionId` must be explicitly added to the SOAP header during implementation. This restriction does not apply to Microsoft® Visual Studio® projects: here the `SessionId` is automatically set in the background.

Allocating changing `SessionIds` can be prevented in the proxy's ini file through the `SequenceSessionId=0` directive (see `SequenceSessionId`).

## 2.4 Documents WSDL Version

The version number of the WSDL used in the Documents resides in the `targetNamespace` of the WSDL:

```
<?xml version="1.0" encoding="UTF-8"?>
<definitions name="DOCUMENTS"
targetNamespace="http://xml.otris.de/ws/DOCUMENTS-
4.0.1870.wsdl"
```

Version of the WSDL used: **DOCUMENTS-4.0.1870.wsdl**

## 2.5 Working in a Microsoft® Visual Studio 2005® project

The following applies to setting up a Microsoft® Visual Studio 2005® project:

1. The Microsoft 3.0 Web service enhancements must be installed:
  - Web Service Enhancements (WSE) 3.0 for Microsoft .NET
2. You need to add the following project references:
  - System.Web.Services
  - System.Xml
3. You should also add the WSDL file to the project:

- Project File Explorer -> Context menu -> Add -> Existing Element -> Select WSDL file

#### 4. The classes from the WSDL must be:

- generated in the target programming language, e.g.  
DOCUMENTS.vb see above
- and added to the project:
- Project File Explorer -> Context menu -> Add -> Existing Element -> Select file

## 2.6 Sample code console application VB

Below we assume that in Microsoft® Visual Studio 2005® Visual Basic a project named "docsample" was created for a console application and that the above references were added to the project.

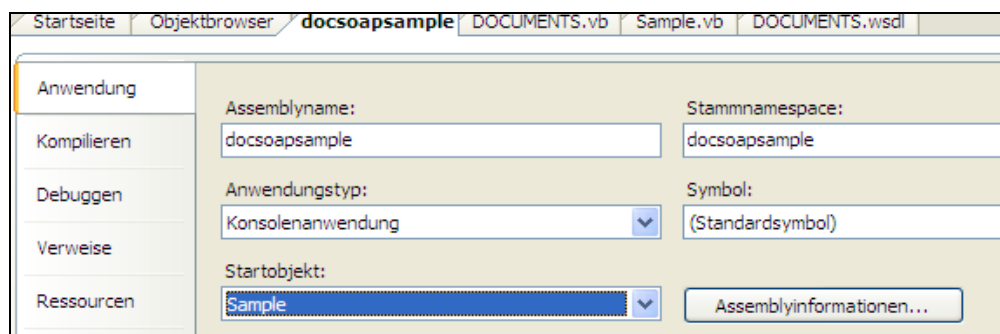
Add a class named "Sample" to the project:

```
Public Class Sample
End Class
```

Moreover, add a "Shared Sub Main" method to the "Sample" class:

```
Public Class Sample
    Public Shared Sub Main()
    End Sub
End Class
```

In the project properties the "Sample" class should now be proposed. In this startup object, the "Main" method is automatically started for application startup.



*Fig. 5: Selecting the startup object for the project*

In the Main method you create a DOCUMENTS object and specify the URL under which the proxy accepts your requests. Following this, login is performed, and the number of file types visible to the user is determined. Finally, logout of the proxy is performed.

```
Public Shared Sub Main()
```

```

' Instantiating the interface
Dim doc as New DOCUMENTS
' The URL under which the proxy can be reached by this
application.
doc.Url = "http://localhost:11001"
Try
    ' Login to proxy
    ' You may not be logged on to the server via the same
user
    doc.login("schreiber", "peachit","willi","", "de")

    ' Which file types are available for this user on the
server?
    Dim ftSD() As FileTypeShortDescr
    ftSD = doc.getFileTypes(False)
    If Not ftSD Is Nothing Then
        Console.WriteLine("Number of files: {0}",
ftSD.Length)
    End If
Catch ex As Exception
    Console.WriteLine("Exception: {0}", ex.Message)
Finally
    'Logout
    doc.logout
End Try
End Sub

```

You may need to customize the `codepage` for the displaying console prior to starting the "docsoapsample.exe" file; this particularly applies when the server supports UTF-8 and the proxy was started in its ini file using the "PortalServerEncoding=UTF-8" option.

```
PortalServerEncoding=UTF-8
```

#### *Encoding setting in the docsoaproxy.ini*

The "chcp" command, followed by *a single Code Page Identifier (MSDN)*, then allows toggling the console's codepage. The result for UTF-8 is the codepage number 65001.

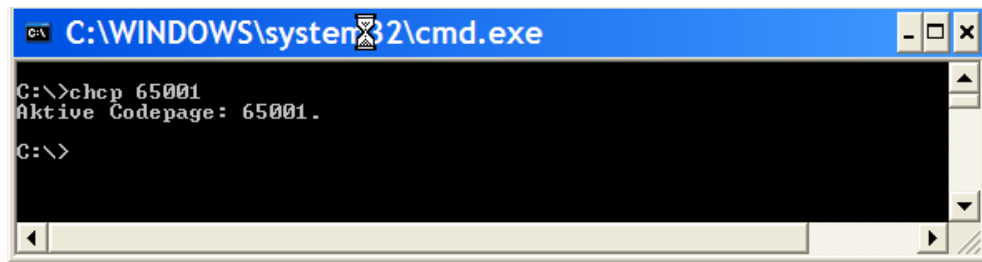


Fig. 6: Setting the Unicode utf-8 codepage to 65001

In order for the text (e.g. umlauts) in the current codepage to be correctly displayed on the current codepage, you must customize the console font, as necessary. This can be carried out via the "Title bar of window -> Context menu -> Properties".

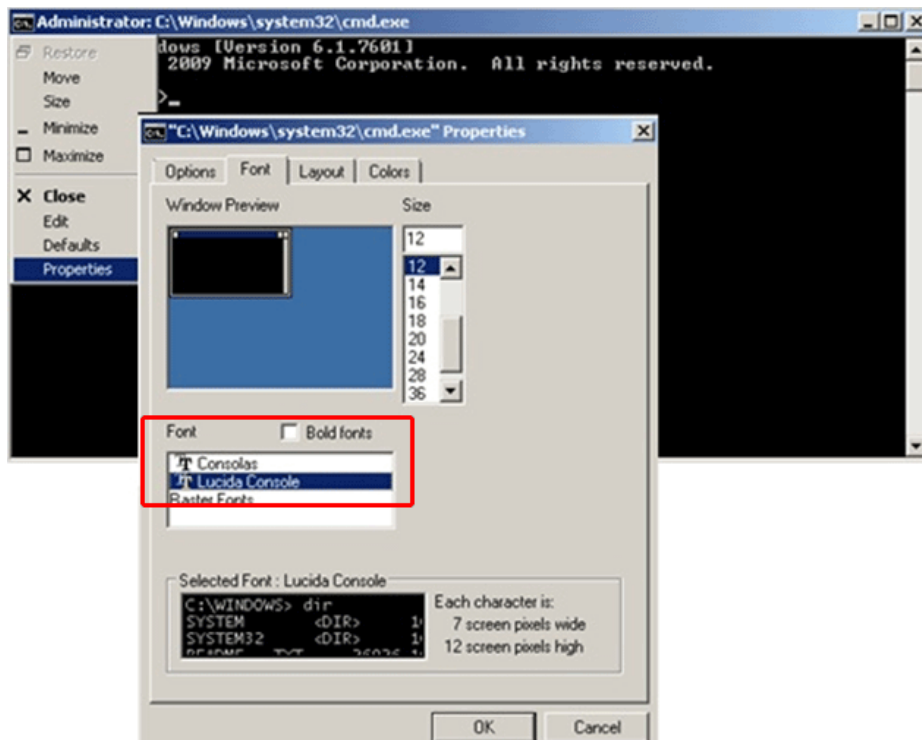


Fig. 7: Setting the console font Lucida Console

## 2.7 Creating a gSOAP Client with Visual C++ 2008®

The gSOAP toolkit provides an automated SOAP and XML data binding for C/C++. It simplifies the development of SOAP Web Services and clients. To create a gSOAP client the following tools are needed, which are included in the gsoap\bin\win32 directory by the gSOAP available from [SourceForge](http://sourceforge.net/projects/gsoap2/):

- The WSDL importer `wsdl2h`.
- The stub compiler `soapcpp2`.



The gSOAP importer `wsdl2h` imports the WSDL of a Web Service and generates a header file that contains the C/C++ declarations of the service components. This Header file will then be processed by the gSOAP compiler `soapcpp2` to generate the stubs for the client. Now we can begin to create a gSOAP client interacts with the web service.

1. Download gSOAP from [SourceForge](#).
2. Importing the WSDL by running `wsdl2h` from the command line:

```
wsdl2h -I C:\gsoap-2.8\gsoap\WS -f -o doc4.h
"C:\Program Files\Documents4\soapproxy\DOCUMENTS.wsdl"
```

Option `-I` tells the importer where to find the required gSOAP include files. To generate C++ code, use option `-f`. Option `-o` specifies the output file. The last parameter specifies the SOAP API WSDL file to be imported. In case of successful importing the output file `doc4.h` will be produced.

3. Generating C++ stubs by running `soapcpp2` from the command line:

```
soapcpp2 -C -L -I C:\gsoap-2.8\gsoap\import -w -x
doc4.h
```

Option `-C` indicates client-side code only. `-L` is needed if you do not want to generate `soapClientLib.cpp`. Option `-I` specifies the include file path. `-w` tells the generator not to produce schema files. `-x` tells it not to produce XML message files.

4. Creating a console application with Visual C++ 2008®

For the sake of simplicity, we create a C++ Win32 Console Application project and accept thereby all of the default settings. The further steps are described below.

- 1) Add the following files generated by gSOAP to the project under

Project > Add Existing Item:

- DOCUMENTS.nsmap
- soapC.cpp
- soapClient.cpp
- soapDOCUMENTSProxy.h
- soapH.h
- soapStub.h

- 2) Add the following files from the gSOAP directory (C:\gsoap-2.8\gsoap) to the project in the same way above:

- stdsoap2.cpp
- stdsoap2.h

- 3) Disable the use of precompiled headers for the following files:

- soapC.cpp
- soapClient.cpp
- Stdsoap2.cpp

From the context menu of the files in the Solution Explorer select Properties > C/C++ > Precompiled Headers, change the option for Create/Use Precompiled Header to Not Using Precompiled Headers.

4) Add include path to gSOAP (C:\gsoap-2.8\gsoap) under Project > Properties > C/C++ > General > Additional Include Directories.

5) Open the console application C++ file. Add the following #includes to the top of the file just after the stdafx.h include:

- #include "soapDOCUMENTSProxy.h"
- #include "DOCUMENTS.nsmmap"
- #include "stdsoap2.h"

## 5. Example code for the gSOAP client

```
#include "stdafx.h"
#include "soapDOCUMENTSProxy.h"
#include "DOCUMENTS.nsmmap"
#include "stdsoap2.h"
#include <iostream>
#include <fstream>
using namespace std;
```

```
int _tmain(int argc, _TCHAR* argv[])
{
    // Create a gSOAP client object.
    DOCUMENTS doc;

    // Endpoint URL of soapproxy (change as needed)
    doc.endpoint = "http://localhost:11001";

    // Sample for login
    _ns2__login req;
    req.locale = "de";
    req.user = "schreiber";
    req.principal = "peachit";
    req.code = "";
    req.passwd = "willi";

    _ns2__loginResponse res;
    int ret = doc.__ns1__login(&req, &res);
    if (ret == SOAP_OK)
        cout << "\nLogin succeeded: " << res.session << endl;
```

```

else

    cerr << "Login failed with the error message: \n"
          << doc.soap->fault->faultstring;

// Sample for getInbox
_ns2__getInbox req2;
_ns2__getInboxResponse res2;
ret = doc.__ns1__getInbox(&req2, &res2);
if (ret == SOAP_OK)
{
    cout << "\nGetInbox succeeded: " << res2.idFolder << endl;
    ns2__FileStatusList* fileStatuses = res2.fileStatuses;
    unsigned int size = fileStatuses->fileStatus.size();
    if (fileStatuses && size > 0)
    {
        // Sample for getFileInfo: We iterate through
        // all files in the inbox.
        _ns2__getFileInfo req3;
        _ns2__getFileInfoResponse res3;
        req3.allFields = true;
        req3.allAttributes = true;
        for (unsigned int i = 0; i < size; ++i)
        {
            req3.fileId = fileStatuses->fileStatus[i]->fileId;
            ret = doc.__ns1__getFileInfo(&req3, &res3);
            if (ret == SOAP_OK)
            {
                cout << "\n===== \n";
                cout << "Information for file with id '"
                      << req3.fileId << "': \n";
                cout << "filetype id: " << res3.filetypeId << endl;
                cout << "filetype name: " << res3.filetypeName << endl;
                cout << "filetype label: " << res3.filetypeLabel << endl;

                if (res3.fieldvalues)
                {
                    cout << "\nField values: \n";
                    vector<ns2__FieldData* >* pFieldData =
                                                                &(res3.fieldvalues->field);

```

```

        vector<ns2__FieldData* >::iterator it;
        for (it = pFieldData->begin(); it != pFieldData->end();
              ++it)
        {
            cout << (*it)->name << ": " << (*it)->value << endl;
        }
    }

    if (res3.documents)
    {
        cout << "\nDoc infos: \n";
        vector<ns2__DocInfo* >* pDocInfo =
            &(res3.documents->document);
        vector<ns2__DocInfo* >::iterator it;
        for (it = pDocInfo->begin(); it != pDocInfo->end(); ++it)
        {
            cout << "Id: " << (*it)->id << ", Name: " << (*it)->name
                << ", Size: " << (*it)->size << endl;

            // Sample for getDocument
            _ns2__getDocument req4;
            _ns2__getDocumentResponse res4;
            req4.fileId = req3.fileId;
            req4.docId = (*it)->id;
            ret = doc.__ns1__getDocument(&req4, &res4);
            if (ret == SOAP_OK)
            {
                string fileName = "Sample_" + res4.name;
                ofstream outfile(fileName.c_str(), ofstream::binary);

                // Write the document content to file named 'fileName'
                // in the current workspace
                outfile.write((const char*)res4.data->__ptr, res4.size);
                outfile.close();
            }
            else
        }
    }

```

```

        cerr << "GetDocument failed with the error message: \n"
              << doc.soap->fault->faultstring;
    }
}
}
}
else
{
    cerr << "GetFileInfo failed with the error message: \n"
          << doc.soap->fault->faultstring;
}
}
}
else
{
    cerr << "GetInbox failed with the error message: \n"
          << doc.soap->fault->faultstring;
}

// Sample for logout
_ns2__logout req5;
_ns2__logoutResponse res5;
ret = doc.__ns1__logout(&req5, &res5);
if (ret != SOAP_OK)
    cerr << "Logout failed with the error message: \n"
          << doc.soap->fault->faultstring;
return 0;
}

```

## 2.8 Creating a WCF Client with Visual C# 2008®

Windows Communication Foundation (WCF) is a tool in the .NET Framework (since the version 3.0) for building service-oriented applications. A WCF client communicates with a WCF service via an endpoint of the service, which consists of the following properties:

- An Address that is a URL and indicates where the endpoint can be accessed.
- A binding that specifies how the data will be transferred.

- A contract that defines the operations available.
- A set of behaviors to customize the local behavior of the service endpoint.

How to create a WCF client with Visual C# 2008® using .NET Framework 3.5 will be described step by step below.

1. Create a new C# console application project named docsoapcs.
2. Open the Add Service Reference dialog under Project > Add Service Reference.
3. In the dialog enter the SOAP API WSDL file (such as C:\Program Files\Documents4\soapproxy\DOCUMENTS.wsdl) into the entry field Address and specify a namespace (being default ServiceReference1) for the generated code and click Ok.
4. Expand Service References in the Solution Explorer and double click the namespace specified above (ServiceReference1). In the Object Browser you can see the generated client proxy class DOCUMENTSPortTypeClient under the namespace docsoapcs.ServiceReference1.
5. Go to the console application C# file (Program.cs) and add the using directive using docsoapcs.ServiceReference1 to the file.
6. Now you can create an instance of the WCF client and then call its methods to invoke the service.
7. Example code for the WCF client

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.IO;
using docsoapcs.ServiceReference1;

namespace docsoapcs
{
    class Program
    {
        static void Main(string[] args)
        {
            // Create a WCF client object with the given client
            // endpoint configuration to invoke the service.
            // Make sure that the URL of the soapproxy is correct.
            // The default is localhost. If the proxy is running
            // on a different machine you have to correct this.

            DOCUMENTSPortTypeClient myDoc = new
            DOCUMENTSPortTypeClient("DOCUMENTS", "http://localhost:11001");

            string currOperation = null;
```

```

string sessionId = null;
try
{
    // Sample for login
    currOperation = "Login";
    myDoc.login("schreiber", "peachit", "willi", "", "de",
               out sessionId);

    Console.WriteLine("\nLogin succeeded: " + sessionId);

    //Sample for getInbox
    currOperation = "GetInbox";
    FileStatus[] fileStatuses;

    string idFolder = myDoc.getInbox(ref sessionId,
                                     out fileStatuses);

    Console.WriteLine("\nGetInbox succeeded: " + idFolder);

    // Sample for getFileInfo: We iterate through
    // all files in the inbox
    currOperation = "GetFileInfo";
    foreach (FileStatus fileStatus in fileStatuses)
    {
        DocInfo[] documents;
        string filetypeId;
        string filetypeName;
        string filetypeLabel;
        FieldData[] fieldvalues = myDoc.getFileInfo(ref sessionId,
                                                    fileStatus.fileId, true, true, null, out documents,
                                                    out filetypeId, out filetypeName, out filetypeLabel);

        Console.WriteLine("\n=====
                           =====\n");

        Console.WriteLine("Information for file with id '"
                           + fileStatus.fileId + "':\n");

        Console.WriteLine("filetype id: " + filetypeId + "\n");
        Console.WriteLine("filetype name: " + filetypeName + "\n");
        Console.WriteLine("filetype label: " + filetypeLabel
                           + "\n");

        Console.WriteLine("\nField values: \n");
        foreach (FieldData fieldData in fieldvalues)
        {

```

```

        Console.WriteLine(fieldData.name + ": "
            + fieldData.value + "\n");
    }

    Console.WriteLine("\nDoc infos: \n");
    foreach (DocInfo docInfo in documents)
    {
        Console.WriteLine("Id: " + docInfo.id + ", Name: " +
            docInfo.name + ", Size: " + docInfo.size + "\n");

        // Sample for getDocument
        currOperation = "GetDocument";
        int size;
        string mime;
        byte[] data;
        string name = myDoc.getDocument(ref sessionId,
            fileStatus.fileId, docInfo.id,
            out size, out mime, out data);

        // Write the document content to file
        // named 'fileName' in the current workspace.
        string fileName = "Sample_" + name;
        FileStream outfile = new FileStream(fileName,
            FileMode.Create);

        outfile.Write(data, 0, data.Length);
        outfile.Close();
    }
}

catch (System.SystemException e)
{
    Console.WriteLine("\n" + currOperation + " failed with
        the error message:\n" + e.Message);
}

finally
{
    // Sample for logout
    if (sessionId != null)
        myDoc.logout(ref sessionId);

    // Close the client
    myDoc.Close();
}

```



```
    }  
  }  
}
```

## 3. API

### 3.1 DOCUMENTS methods

#### Documents.archiveFiles

Description	Archive DOCUMENTS files.
Input parameter	<p><b>fileIds:</b> 0..n string List of technical names of the DOCUMENTS files to be archived.</p> <p><b>deleteOnSuccess:</b> 1 Boolean optional Specifies whether successfully archived DOCUMENTS files should be deleted from DOCUMENTS afterwards. If this parameter is missing, this will be interpreted as <b>false</b>.</p>
Output parameter	<p><b>statusList:</b> 0..n <b>ArchiveStatus</b> Contains messages on history for each archive operation.</p>
Sample VB	<pre>' Array with status information about the files which ' are currently in the users inbox Dim fileStatuses() As FileStatus = Nothing  Dim idFolder As New String(String.Empty) ' get the inbox files idFolder = doc.getInbox(fileStatuses)  ' Now pick the first file If fileStatuses.Length &gt; 0 Then      Dim archiveStatus() As ArchiveStatus = Nothing     Dim deleteOnSuccess As Boolean = False      Dim fileIds() As String = {fileStatuses(0).fileId}     ' archive     archiveStatus = doc.archiveFiles(fileIds, deleteOnSuccess)      ' and check for the status information     For Each ars As ArchiveStatus In archiveStatus         Console.WriteLine("fileId: {0}", ars.fileId)         Console.WriteLine("messages: {0}", ars.messages)     Next Else     Console.WriteLine("No file to archive") End If</pre>
Explanation	We get the first DOCUMENTS file from the Inbox, and then determine its ID. This DOCUMENTS file is then transferred to the archive assigned to it via its file type. The returning ArchiveStatus object whose messages are written to the console will inform about successful archive transfer.
Comment	The <b>ArchiveStatus</b> lists error messages for transfer.

## Documents.browseFolder

Description	List DOCUMENTS files of a folder.
Input parameter	<p><b>folderName:</b> 0..1 string, optional Name of a public folder.</p> <p><b>folderId:</b> 0..1 string, optional The technical identifier of a folder can alternately be specified for <b>folderName</b> or <b>folderType</b>.</p> <p><b>folderType:</b> 0..1 String, optional The type label of a personal default folder can alternately be specified for <b>folderName</b> or <b>folderId</b>. A list of allowed labels can be found in the Appendix <b>Label for personal default folders</b>.</p> <p><b>startIndex:</b> 1 Integer Index of first file to be output. Counting starts with number 0. When <b>startIndex</b>=0 and <b>count</b> = -2, all DOCUMENTS files of the folder will be listed at once. This may take quite a long time, though.</p> <p><b>count:</b> 0..1 Integer, optional The desired maximum number of files to be listed must always be specified along with <b>startIndex</b>. If the value is set to -1, the function will use the number of files preset by the user. If the value is set to -2, all files as of the <b>startIndex</b> will be listed.</p> <p><b>preview:</b> 0..1 Boolean Specifies whether field contents should be output for preview.</p>
Output parameter	<p><b>fieldNames:</b> 1 StringList List of field names for preview. The same labels as described in the Appendix are used as field names for file beschreiben (see field labels in 5.2). The parameter will become obsolete if no preview is requested. This parameter is a simple output parameter; you cannot pass a list of field names here.</p> <p><b>files:</b> 0..n FolderFile File list of folder or its currently requested section.</p> <p><b>previousIndex:</b> 1 Integer If going backwards in the list is allowed, then the starting index for the first DOCUMENTS file of the previous page will be output here along with a negative value.</p> <p><b>nextIndex:</b> 1 Integer If going forward in the list is allowed, then the starting index for the first DOCUMENTS file of the next page will be output here along with a negative value.</p>
Return value	<p><b>headline:</b> 1 string The heading for the folder.</p>
Sample VB	<pre>Dim folderType As New String("Favourites") ' we don't need name and id Dim folderName As New String(String.Empty) Dim folderId As New String(String.Empty)  Dim files() As FolderFile = Nothing</pre>

	<pre> Dim headline As String Dim startIndex As Integer = 0 Dim previousIndex As Integer Dim nextIndex As Integer  Dim count As Nullable(Of Integer) Dim countSpecified As Boolean = count.HasValue Dim preview As Boolean = False Dim fieldNames As New List(Of String)  ' browse headline = doc.browseFolder(folderName, folderId, folderType, _     startIndex, _     count, countSpecified, _     preview, _     fieldNames.ToArray, files, _     previousIndex, nextIndex)  Console.WriteLine("headline: {headline}") If Not files Is Nothing Then     Console.WriteLine("Files found: {0}", files.Length())     ' files in folder     For Each ff As FolderFile In files         Console.WriteLine("Id:{0}", ff.id)     Next End If </pre>
Explanation	<p>We determine the file list in the "Favorites" folder, identifying these folders via the FolderType. Because we do not want to output a field list, we will switch preview to false. Moreover, the entire list should be requested, where the nullable count parameter is not to be given a value, so countSpecified will become automatically false. After starting the browse function we will iterate through the returned array of FolderFiles objects, and print the file ID to the console.</p>
Comment	<p>Error strings begin often with a technical abbreviation which may be followed by additional specifications separated by pipe symbols. The names always begin with the "DlcErr" character string".</p> <p>Some personal default folders may now contain other subfolders which do not consider this function on specifying <b>folderType</b>. To make their contents also reachable, the Documents.getFolderStructure function has been included.</p>

## Documents.cancelWorkflow

Description	Finishes the workflow of a DOCUMENTS file.
Input parameter	<b>fileId</b> : 1 string ID of the DOCUMENTS file for which a running workflow should be stopped.
Output parameter	None
Return value	None
Sample VB	<pre>Dim fileStatuses() As FileStatus = Nothing  doc.getInbox(fileStatuses)  If fileStatuses.Length &gt; 0 Then     For Each fs As FileStatus In fileStatuses         Console.WriteLine("fileId : {0}", fs.fileId)         doc.cancelWorkflow(fs.fileId)     Next Else     Console.WriteLine("No files found") End If</pre>
Explanation	All DOCUMENTS files in the logged-in user's Inbox are determined and, based on the fileId the cancelWorkflow method of a DOCUMENTS object, are started for each of these DOCUMENTS files.
Comment	

## Documents.createFile

Description	Create a new DOCUMENTS file.
Input parameter	<p><b>filetype</b>: 1 string The name of the <b>file type</b> for which a new DOCUMENTS file should be created. In order for a new file to be created, the DOCUMENTS file type must be enabled.</p> <p>To create a new <b>archive file</b>, you must specify the Destination identifier for archive.</p> <p><b>fields</b>: 0..n FieldData The number of FieldData objects of the file fields to be written, whereat their element names equal the field names and their element values equal the field values. All values must be specified as a string.</p> <p><b>addDocs</b>: 0..n DocUploadData A list including the documents to be added and the related data.</p>
Output parameter	None
Return value	<b>fileId</b> : 1 string A technical identifier of the new file. When an archive file has been created, the Archive file key will be returned.
Sample VB	<pre>' the filetype of the new file Dim filetype As New String("ftRecord")  ' Building the fields Dim fds As New List(Of FieldData)  Dim remark As New FieldData</pre>

	<pre> remark.name = "hrRemarks" remark.value = "The remark" fds.Add(remark)  Dim no As New FieldData no.name = "hrEmployeeNo" no.value = "00001" fds.Add(no)  ' Create Attachements Dim newFile As New DocUploadData Dim register As New String("Documents") newFile.register = register  Dim path As New String("test.txt") newFile.name = path  Dim newByteArray() As Byte = System.IO.File.ReadAllBytes(path) newFile.data = newByteArray  addDocs.Add(newFile)  Dim newFileId As New String(String.Empty)  newFileId = doc.createFile filetype, fds.ToArray, addDocs.ToArray) </pre>
Explanation	<p>A new DOCUMENTS file is created for the "ftRecord" file type. To do this, a field list is created which accepts the individual FieldData objects whose field values should then be set for the DOCUMENTS file. In addition, a document is attached. For this, the "test.txt" file, which must reside in the directory where the application was started, is read as ByteArray. The created DocUploadData object is added to the list, which is then included as an array with the createFile call.</p>
Comment	<p>To create an EAS archive file directly the function Documents.createFile2 is recommended.</p>

#### Documents.createFile2

Description	<p>Creates a new file similar to Documents.createFile. However this function is also able to create an EAS archive file directly (without generating a DOCUMENTS file) via setting the directEAS parameter to true.</p>
Input parameter	<p><b>filetype:</b> 1 String The name of the <b>file type</b> for which a new DOCUMENTS/archive file should be created. In order for a new file to be created, the file type must be released.</p> <p>To create a new <b>archive file</b>, you must specify the Destination identifier for archive.</p> <p><b>directEAS:</b> 1 Boolean For an EAS archive the file creation can take place directly via setting this parameter to true. Otherwise a file is at first created within DOCUMENTS whereat the default values are set and (if applicable) the scripts are executed.</p> <p><b>fields:</b> 0..n FieldData A list of FieldData objects of the file's fields to be written, whereat their element names equal the field names and their element values equal the field values. All values must be specified as a string.</p> <p><b>addDocs:</b> 0..n DocUploadData A list including the documents to be added and the related data.</p>

Output parameter	None
Return value	<b>fileId</b> : 1 String A technical identifier of the new file. When an archive file has been created, the Archive file key will be returned.
Sample VB	<pre>' the filetype of the new file Dim filetype As New String("ftRecord@peachitStore1")  ' Building the fields Dim fds As New List(Of FieldData)  Dim remark As New FieldData remark.name = "hrRemarks" remark.value = "The remark" fds.Add(remark)  Dim directEAS as Boolean directEAS = True  Dim newFileId As New String(String.Empty) newFileId = doc.createFile2(filetype, fds.ToArray, addDocs.ToArray)</pre>
Explanation	An EAS archive file is created for the "ftRecord" file type. To do this, a field list is created which accepts the individual FieldData objects whose field values should then be set for the archive file. The archive file is generated directly (without creating a DOCUMENTS file).
Comment	This method is more efficient than Documents.createFile in case of creating an EAS archive file directly.

### Documents.createFile3

Description	Creates a new file similar to Documents.createFile2. However this function is able to upload a blob from the file system directly. The blobs must be located in a shared directory specified by the option <code>BlobBasePath</code> in <code>docsoaproxy.ini</code> , which can be accessed by the SOAPProxy. Instead of the blob content encoded in base64 format only the relative file path has to be passed.
Input parameter	<p><b>filetype</b>: 1 String The name of the <b>file type</b> for which a new DOCUMENTS/archive file should be created. In order for a new file to be created, the file type must be released.</p> <p>To create a new <b>archive file</b>, you must specify the Destination identifier for archive.</p> <p><b>directEAS</b>: 1 Boolean For an EAS archive the file creation can take place directly via setting this parameter to true. Otherwise a file is at first created within DOCUMENTS whereat the default values are set and (if applicable) the scripts are executed.</p> <p><b>fields</b>: 0..n FieldData A list of FieldData objects of the file's fields to be written, whereat their element names equal the field names and their element values equal the field values. All values must be specified as a string.</p> <p><b>addDocs</b>: 0..n DocUploadData3 A list including the documents to be added and the related data.</p>
Output	None

parameter	
Return value	<b>fileId:</b> 1 String A technical identifier of the new file. When an archive file has been created, the Archive file key will be returned.
Sample VB	<pre> ' the filetype of the new file Dim filetype As New String("ftRecord@peachitStore1")  ' Building the fields Dim fds As New List(Of FieldData)  Dim remark As New FieldData remark.name = "hrRemarks" remark.value = "The remark" fds.Add(remark)  Dim directEAS as Boolean directEAS = False  ' Add a document to the register "Documents" Dim addDocs As New List(Of DocUploadData3)  Dim sameDoc As New DocUploadData3 sameDoc.register = "Documents" sameDoc.name = "test.txt" sameDoc.path = "test.txt" 'located in BlobBasePath sameDoc.replaceSpecified = False sameDoc.versioningSpecified = False sameDoc.deleteBlobSpecified = True sameDoc.deleteBlob = False  addDocs.Add(sameDoc)  Dim newFileId As New String(String.Empty) newFileId = doc.createFile3(filetype, fds.ToArray, addDocs.ToArray) </pre>
Explanation	A file is created for the "ftRecord" file type. To do this, a field list is created which accepts the individual FieldData objects whose field values should then be set for the file. We will also build a list of DocUploadData3 objects with a document located in a shared directory specified by the option BlobBasePath. These lists will then be passed as arrays to the editFile3 method.
Comment	This method is available since WSDL DOCUMENTS-4.0.1870.

#### Documents.DeleteDocuments

Description	Delete documents from a DOCUMENTS file.
Input parameter	<b>fileId:</b> 1 string Technical name of file. <b>docIds:</b> 0..n string The IDs of the documents to be deleted. These can be determined via Documents.getFileInfo.
Output parameter	None
Return value	<b>status:</b> 0..n DeleteStatus An indeterminate number of DeleteStatus objects that provide information about a successful/unsuccessful delete operation.



Sample VB	<pre> Dim fileStatuses() As FileStatus = Nothing  Dim idFolder As New String(String.Empty) ' Get all files from the inbox idFolder = doc.getInbox(fileStatuses) If fileStatuses.Length &lt;= 0 Then     Console.WriteLine("No file in inbox")     Return True End If ' and check each file for documents For Each fs As FileStatus In fileStatuses     Dim fileId As New String(fs.fileId)     Dim allFields As Boolean = False     Dim allAttributes As Boolean = False     Dim wishedFieldNames As New List(Of String)      Dim docinfo() As DocInfo = Nothing     Dim fileTypeId As New String(String.Empty)     Dim fileTypeNames As New String(String.Empty)     Dim fileTypeLabel As New String(String.Empty)     Dim fds() As FieldData = Nothing      ' getFileInfo for file     fds = doc.getFileInfo(fileId, allFields, allAttributes, wishedFieldNames.ToArray, docinfo, fileTypeId, fileTypeNames, fileTypeLabel)     If Not docinfo Is Nothing Then ' there could be documents         Dim delDocs As New List(Of String)         ' get all ids of the attached documents         For Each di As DocInfo In docinfo             delDocs.Add(di.id)         Next         If delDocs.Count &gt; 0 Then             Dim statuses() As DeleteStatus = Nothing             ' delete the documents             statuses = doc.deleteDocuments(fileId, delDocs.ToArray)             If Not statuses Is Nothing Then                 For Each st As DeleteStatus In statuses                     Console.WriteLine("Deleted Document Id: {0} Deleted: {1} Message: {2}", _ st.id, st.deleted, st.messages)                 Next             End If         Else             Console.WriteLine("No documents found")         End If     End If Next </pre>
-----------	--

Explanation	We determine all DOCUMENTS files residing in the logged-in user's Inbox. For each of these DOCUMENTS files we call the getFileInfo method using the fileId to check on the DocInfo array whether the DOCUMENTS file contains documents. We will then collect the IDs of these documents in a deletion list, and then perform deletion via deleteDocuments. Following this, a check will be made using the DeleteStatus objects on whether deletion was successful by outputting the status to the console.
Comment	

#### Documents.deleteFiles

Description	Allows deleting DOCUMENTS files.
Input parameter	<b>fileIds:</b> 0..n string The technical names of the DOCUMENTS files to be deleted. You can also use the Archive file keys.
Output parameter	None
Return value	<b>statusList:</b> 0..n DeleteStatus An indeterminate number of DeleteStatus objects that provide information about successful/unsuccessful delete operation.
Sample VB	<pre> Dim fileStatuses() As FileStatus = Nothing  Dim idFolder As New String(String.Empty) ' Get all files from the inbox idFolder = doc.getInbox(fileStatuses)  ' Now iterate through that array and the files If fileStatuses.Length &gt; 0 Then     For Each fs As FileStatus In fileStatuses         Console.WriteLine("fileId : {0}", fs.fileId)         Dim fileIds() As String = {fs.fileId}         ' delete the file         Dim deleteStatus() As DeleteStatus = Nothing         deleteStatus = doc.deleteFiles(fileIds)          For Each ds As DeleteStatus In deleteStatus             Console.WriteLine("Id : {0} deleted: {1} messages: {2}", ds.id, ds.deleted, ds.messages)         Next     Next Else         Console.WriteLine("No files found")     End If </pre>
Explanation	We get the logged-in user's DOCUMENTS files residing in his or her Inbox. We will then iterate through these DOCUMENTS files and, after determining the file ID and inserting it into the delete array, call the DeleteFiles method. The delete status for this operation will then be printed via the console.
Comment	

#### Documents.describeFileType

Description	Displays detailed information on a file type. This may be archive, field, tab, or workflow information.
-------------	---

Input parameter	<p><b>name:</b> String The name of the file type for which the information should be determined. Alternatively, you can also use the <b>id</b>. In this case, you need to pass an empty string for <b>name</b>.</p> <p><b>id:</b> String ID of file type to be examined. Alternatively, you can use the <b>name</b>. In this case, you need to pass an empty string for the <b>id</b>.</p> <p><b>categories:</b> 1 string The string is used to specify the data to be requested. In its dependency, the corresponding data structures are populated with <b>FileTypeDescription</b>. The following options are available:</p> <ul style="list-style-type: none"> <li>• <b>"fields"</b> <ul style="list-style-type: none"> <li>○ <b>"enum"</b> If the field is of the "enum" type and its enumeration values are also to be output, then "enum" must be specified, together with "fields", in the categories string: "fields, enum". The individual values for the languages are displayed in a single string (1), e.g. "year; de:Jahr; en:Year".</li> <li>○ <b>"fieldlabel" / "fieldlabel.locale"</b> Including "fields,fieldlabel" / "fields,fieldlabel.locale" to the categories string to get the entire labels of the fields and their ergonomic names in the current user's language, respectively. The returned values are available for both in the member 'fieldlabel' of <b>FieldDescription</b>, e.g. the value "de:Betreff; en:Subject" for "fieldlabel" and "Subject" for "fieldlabel.locale" in case that English is the current user's language, respectively. It has to be mentioned, if the categories string contains both ("fieldlabel" and "fieldlabel.locale"), "fieldlabel" will be ignored.</li> <li>○ <b>"initialValue"</b> If specification for field values should be additionally displayed, the categories string must contain the value "initialValue" in addition to "fields".</li> </ul> </li> <li>• <b>"docregisters"</b> Inserts information on the document tabs in the FileTypeDescription.</li> <li>• <b>"archiveinfo"</b> Inserts archive information into the FileTypeDescription.</li> <li>• <b>"workflowinfo"</b> Inserts workflow information into the FileTypeDescription.</li> </ul> <p>The individual options can also be assembled comma-separated into a single string, e.g. "fields, enum, docregisters, archiveinfo".</p>
Output parameter	None
Return value	<b>description:</b> 1 <b>FileTypeDescription</b>

Sample VB	<pre> Try   Dim ftsd() As FileTypeShortDescr   ftsd = doc.getFileTypes(False)   ' and iterate   For Each fsd As FileTypeShortDescr In ftsd     If Not fsd Is Nothing Then        'describe this filetype by passing his "name"       Dim name As String       name = (fsd.name)        ' here we want to learn something about the fields in this       filetype, their name and the filetype       Dim cat As New String("fields")       Dim ftd As FileTypeDescription       ' the call of the documents method at last       ftd = doc.describeFileType(name, "", cat)       Dim fields() As FieldDescription       fields = ftd.fields       Console.WriteLine("Fields count: {0} ", fields.Length())       For Each f As FieldDescription In fields         Console.WriteLine("Fields Name: {0} Type: {1} ", f.name, f.type)       Next       End If 'if not fsd     Next ' for each   Catch ex AS Exception     Console.WriteLine("DescribeFileType Message: {0}", ex.Message)   Return False End Try </pre>
Explanation	<p>Information on the fields of the file types should be output, particularly name and field type. This request is controlled via the category string cat= "fields", which ensures that the fields array is populated in the returned FileTypeDescription. Next, we need information saying for which file type this should be performed. For this, the name of the file type is passed to the describeFileType method. In the file type description, we then iterate through the fields and print name and type to the console.</p>
Comment	<p>File type description is essentially controlled through the "categories" string. If this is not really given a value, the resultant data structures will then not be populated with values either.</p> <p>The <b>archiveinfo</b> output is only supported for EAS since WSDL Documents-4.0.1827. If the label of the file type should be retrieved, you can use the function <b>Documents.describeFileType2</b> below.</p>

## Documents.describeFileType2

Description	Displays detailed information on a file type similar to <b>Documents.describeFileType</b> , however this function can also determine the file type label in the current user's language.
Input parameter	<p><b>name:</b> String The name of the file type for which the information should be determined. Alternatively, you can also use the <b>id</b>. In this case, you need to pass an empty string for <b>name</b>.</p> <p><b>id:</b> String ID of file type to be examined. Alternatively, you can use the <b>name</b>. In this case, you need to pass an empty string for the <b>id</b>.</p> <p><b>categories:</b> 1 string The string is used to specify the data to be requested. In its dependency, the corresponding data structures are populated with <b>FileTypeDescription</b>. The following options are available:</p> <ul style="list-style-type: none"> <li>• <b>"fields"</b> <ul style="list-style-type: none"> <li>○ <b>"enum"</b> If the field is of the "enum" type and its enumeration values are also to be output, then "enum" must be specified, together with "fields", in the categories string: "fields, enum". The individual values for the languages are displayed in a single string (1), e.g. "year; de:Jahr; en:Year".</li> <li>○ <b>"fieldlabel" / "fieldlabel.locale"</b> Including "fields,fieldlabel" / "fields,fieldlabel.locale" to the categories string to get the entire labels of the fields and their ergonomic names in the current user's language, respectively. The returned values are available for both in the member 'fieldlabel' of <b>FieldDescription</b>, e.g. the value "de:Betreff; en:Subject" for "fieldlabel" and "Subject" for "fieldlabel.locale" in case that English is the current user's language, respectively. It has to be mentioned, if the categories string contains both ("fieldlabel" and "fieldlabel.locale"), "fieldlabel" will be ignored.</li> <li>○ <b>"initialValue"</b> If specification for field values should be additionally displayed, the categories string must contain the value "initialValue" in addition to "fields".</li> </ul> </li> <li>• <b>"docregisters"</b> Inserts information on the document tabs in the FileTypeDescription.</li> <li>• <b>"archiveinfo"</b> Inserts archive information into the FileTypeDescription.</li> <li>• <b>"workflowinfo"</b> Inserts workflow information into the FileTypeDescription.</li> </ul>

	The individual options can also be assembled comma-separated into a single string, e.g. "fields, enum, docregisters, archiveinfo".
Output parameter	None
Return value	<b>description: 1 FileTypeDescription2</b>
Sample VB	<pre> Try     sampleLogin(doc) ' First we get the filetypes Dim ftsd() As FileTypeShortDescr ftsd = doc.getFileTypes(False) ' and iterate through all returned FileTypeShortDescr  For Each fsd As FileTypeShortDescr In ftsd     If Not fsd Is Nothing Then         Console.WriteLine(New String("-", 10))         Console.WriteLine("FileType Name: {0}", fsd.name)         Dim id As New String(String.Empty)         id = fsd.id         ' the name is optional if we pass the id         Dim name As New String(String.Empty)         Dim cat As New String(String.Empty)         Dim ftd As FileTypeDescription2          ftd = doc.describeFileType2(name, id, cat)         If Not ftd Is Nothing Then             Console.WriteLine("Filetype name: {0} label: {1}", ftd.name, ftd.label)         End If     End If Next Catch ex As Exception     Console.WriteLine("DescribeFileType2: {0}", ex.Message) Return False End Try </pre>
Explanation	The label of a file type should be displayed. This request is controlled via the empty category string cat="", which ensures that the label is populated in the returned FileTypeDescription2. Next, we need information saying for which file type this should be performed. For this, the id of the file type is passed to the FileTypeDescription2 method. In the FileTypeDescription2, we print the name and the label to the console.
Comment	<p>File type description is essentially controlled through the "categories" string. If this is not really given a value, the resultant data structures will then not be populated with values either.</p> <p>The <b>archiveinfo</b> output is only supported for EAS since WSDL Documents-4.0.1827. The same applies for this method itself.</p>

#### Documents.editFile

Description	Edit a DOCUMENTS file.
Input parameter	<b>fileid</b> : 0..1 string, optional The technical identifier of the file must be specified unless <b>keyfield</b> , <b>keyvalue</b> and <b>filetype</b> are specified instead. An Archive file key can also be used as a fileid.

	<p><b>filetype:</b> 0..1 string, optional File type or Destination identifier for archive of the wanted DOCUMENTS and archive file respectively. This parameter will be redundant when specifying <b>fileId</b>; however, it must be used together with <b>keyfield</b> and <b>keyvalue</b>.</p> <p><b>keyfield:</b> 0..1 string, optional Name of a field used as a key field to identify the DOCUMENTS file. This parameter will be redundant when specifying <b>fileId</b>; however it must be used together with <b>keyvalue</b> and <b>filetype</b>.</p> <p><b>keyvalue:</b> 0..1 string, optional The value of a key field to identify the DOCUMENTS file. This parameter will be redundant when specifying <b>fileId</b>; however, it must then be used together with <b>keyfield</b> and <b>filetype</b>.</p> <p><b>fields:</b> 0..n FieldData The file fields to be overwritten as the number of field data objects, where their element names equal the field names and their element value equal the field values. All values must be specified as a string.</p> <p><b>addDocs:</b> 0..n DocUploadData A list including the documents to be added and the data required for it.</p>
Output parameter	None
Return value	<p><b>newId</b></p> <p>The fileId of the edited file is returned. This may change with archive files in case the file has been versioned.</p>
Sample VB	<pre>Dim filetype As New String(String.Empty) Dim keyField As New String(String.Empty) Dim keyValue As New String(String.Empty)  ' we add no documents Dim addDocs As New List(Of DocUploadData)  ' the fielddata list with the fielddata to change Dim fds As New List(Of FieldData) Dim rm As New FieldData  rm.name = "hrRemarks" rm.value = "The new remark"  fds.Add(subject) doc.editFile(fileId, filetype, keyField, keyValue, fds.ToArray, addDocs.ToArray)</pre>
Explanation	The fileId of a DOCUMENTS file belonging to the "crmNote" file type is known. We will build a list of FieldData objects that includes the name ".name" of the "hrRemarks" field and the new value ".value" for the "The new remark" field. This list will then be passed as an array to the editFile method.
Comment	Make sure that the file is identified either via the "fileId" or via "filetype + keyfield + keyvalue" in a unique manner. However, through WSDL DOCUMENTS-4.0.1827 an archive file can be only identified via the "fileId", in other words the parameter "fileId" must be specified.

	<p>The field values will also be overwritten if the new and old values are equal.</p> <p>When using an archive file key, consider the different format for the respective EE.i, EE.x and EAS archives.</p> <p>Through WSDL DOCUMENTS-4.0.1827 a new file with the default values will be created automatically, if no file is found.</p> <p>However, since WSDL DOCUMENTS-4.0.1870 an error should be thrown in this case for archive files.</p> <p>For versioning or replacing documents you can use the method <b>Documents.editFile2</b>.</p>
--	--

#### Documents.editFile2

Descrip tion	<p>Edit a DOCUMENTS/archive file similar to <b>Documents.editFile</b>. However, this method makes it possible that a document in an active file can be versioned or replaced by a same-named document.</p> <p>For archive files this method works in the same way as <b>Documents.editFile</b>.</p>
Input parame ter	<p><b>fileId</b>: 0..1 string, optional The technical identifier of the file must be specified unless <b>keyfield</b>, <b>keyvalue</b> and <b>filetype</b> are specified instead. An Archive file key can also be used as a fileId.</p> <p><b>filetype</b>: 0..1 string, optional File type or Destination identifier for archive of the wanted DOCUMENTS and archive file respectively. This parameter will be redundant when specifying <b>fileId</b>; however, it must be used together with <b>keyfield</b> and <b>keyvalue</b>.</p> <p><b>keyfield</b>: 0..1 string, optional Name of a field used as a key field to identify the DOCUMENTS file. This parameter will be redundant when specifying <b>fileId</b>; however it must be used together with <b>keyvalue</b> and <b>filetype</b>.</p> <p><b>keyvalue</b>: 0..1 string, optional The value of a key field to identify the DOCUMENTS file. This parameter will be redundant when specifying <b>fileId</b>; however, it must then be used together with <b>keyfield</b> and <b>filetype</b>.</p> <p><b>fields</b>: 0..n FieldData The file fields to be overwritten as the number of field data objects, where their element names equal the field names and their element value equal the field values. All values must be specified as a string.</p> <p><b>addDocs</b>: 0..n DocUploadData2 A list including the documents to be added and the data required for it.</p>
Output parame ter	None
Return value	<p><b>newId</b></p> <p>The fileId of the edited file is returned. This may change with archive files in case the file has been versioned.</p>
Sample VB	<pre>Dim filetype As New String(String.Empty) Dim keyField As New String(String.Empty) Dim keyValue As New String(String.Empty)</pre>



	<pre> ' the fielddata list with the fielddata to change Dim fds As New List(Of FieldData) Dim rm As New FieldData rm.name = "hrRemarks" rm.value = "The new remark" fds.Add(rm)  ' Add a document with the same name as the document "test.txt" ' already existing in the register "Documents" Dim addDocs As New List(Of DocUploadData2) Dim path As New String("test.txt") Dim newByteArray() As Byte = System.IO.File.ReadAllBytes(path)  Dim sameDoc As New DocUploadData2 sameDoc.register = "Documents" sameDoc.name = "test.txt" sameDoc.data = newByteArray sameDoc.replaceSpecified = True sameDoc.replace = True sameDoc.versioningSpecified = True sameDoc.versioning = True  addDocs.Add(sameDoc)  doc.editFile2(fileId, filetype, keyField, keyValue, fds.ToArray, addDocs.ToArray) </pre>
Explan ation	<p>The fileId of a DOCUMENTS file belonging to the "ftRecord" file type is known. We will build a list of FieldData objects that includes a field with the name "hrRemarks" and the new value "The new remark". We will also build a list of DocUploadData2 objects that contains a document with the same name as a document already existing in the file. These lists will then be passed as arrays to the editFile2 method.</p>
Comme nt	<p>This method makes versioning or replacing of a document in an active file possible.</p> <p>Make sure that the file is identified either via the "fileId" or via the combination "filetype + keyfield + keyvalue" in a unique manner. Examples for specification of the parameter "filetype" for an archive file (see Destination identifier for archive):  EAS: Filetype7@myStore  EE.i: \$(#STANDARD)\REGTEST@eei1  EE.x:  Unit=Default/Instance=Default/View=REGTEST*Unit=Default/Instance=Default/DocumentSchema=REGTEST@eex1</p> <p>The field values will also be overwritten if the new and old values are equal.</p> <p>When using an archive file key, consider the different format for the respective EE.i, EE.x and EAS archives.</p> <p>If no active file is found, a new file with the default values will be created automatically. However, in this case for archive files an error will be thrown.</p> <p>This method is available since WSDL DOCUMENTS-4.0.1 870.</p>

## Documents.editFile3

Description	<p>Edit a DOCUMENTS/archive file similar to <b>Documents.editFile2</b>. However this function is able to upload a blob from the file system directly. The blobs must be located in a shared directory specified by the option <code>BlobBasePath</code> in <code>docsoaproxy.ini</code>, which can be accessed by the SOAPProxy. Instead of the blob content encoded in base64 format only the relative file path has to be passed.</p>
Input parameter	<p><b>fileId</b>: 0..1 string, optional The technical identifier of the file must be specified unless <b>keyfield</b>, <b>keyvalue</b> and <b>filetype</b> is specified instead. An Archive file key can also be used as a fileId.</p> <p><b>filetype</b>: 0..1 string, optional File type or Destination identifier for archive of the wanted DOCUMENTS and archive file respectively. This parameter will be redundant when specifying <b>fileId</b>; however, it must be used together with <b>keyfield</b> and <b>keyvalue</b>.</p> <p><b>keyfield</b>: 0..1 string, optional Name of a field used as a key field to identify the DOCUMENTS file. This parameter will be redundant when specifying <b>fileId</b>; however it must be used together with <b>keyvalue</b> and <b>filetype</b>.</p> <p><b>keyvalue</b>: 0..1 string, optional The value of a key field to identify the DOCUMENTS file. This parameter will be redundant when specifying <b>fileId</b>; however, it must then be used together with <b>keyfield</b> and <b>filetype</b>.</p> <p><b>fields</b>: 0..n FieldData The file fields to be overwritten as the number of field data objects, where their element names equal the field names and their element value equal the field values. All values must be specified as a string.</p> <p><b>addDocs</b>: 0..n DocUploadData3 A list including the documents to be added and the data required for it.</p>
Output parameter	None
Return value	<p><b>newId</b> The fileId of the edited file is returned. This may change with archive files in case the file has been versioned.</p>
Sample VB	<pre>Dim filetype As New String(String.Empty) Dim keyField As New String(String.Empty) Dim keyValue As New String(String.Empty)  ' the fielddata list with the fielddata to change Dim fds As New List(Of FieldData) Dim rm As New FieldData rm.name = "hrRemarks" rm.value = "The new remark" fds.Add(rm)  ' Add a document with the same name as the document "test.txt" ' already existing in the register "Documents" Dim addDocs As New List(Of DocUploadData3)  Dim sameDoc As New DocUploadData3</pre>

	<pre> sameDoc.register = "Documents" sameDoc.name = "test.txt" sameDoc.path = "test_new.txt" 'located in BlobBasePath sameDoc.replaceSpecified = True sameDoc.replace = True sameDoc.versioningSpecified = True sameDoc.versioning = True sameDoc.deleteBlobSpecified = True sameDoc.deleteBlob = False  addDocs.Add(sameDoc)  doc.editFile3(fileId, filetype, keyField, keyValue, fds.ToArray, addDocs.ToArray) </pre>
Explanation	The fileId of a DOCUMENTS file belonging to the "ftRecord" file type is known. We will build a list of FieldData objects that includes a field with the name "hrRemarks" and the new value "The new remark". We will also build a list of DocUploadData3 objects that contains a document with the same name as a document already existing in the file. These lists will then be passed as arrays to the editFile3 method.
Comment	<p>See the comment from Documents.editFile2.</p> <p>This method is available since WSDL DOCUMENTS-4.0.1870.</p>

#### Documents.followUp

Description	Move a DOCUMENTS file from the user's Inbox to the resubmission folder.
Input parameter	<p><b>fileId</b>: 1 string File ID.</p> <p><b>timeAbsolute</b>: 0..1 DateTime.iso8601 Time at which the DOCUMENTS file is to reappear in the Inbox. If the time is in the past, this function will fail with an error. Only times with 15, 30, 45 and 00 minutes should be used.</p> <p><b>timeAbsoluteSpecified</b>: 1 Boolean Only tells the SOAP client that timeAbsolute was given a value.</p> <p><b>isUTC</b>: 1 Boolean Information on whether the time refers to international standard time (UTC=Coordinated Universal Time) and must therefore be converted to the Portal server's local time first.</p> <p><b>comment</b>: 1 string Comment or reason for returning the DOCUMENTS file.</p>
Output parameter	None
Return value	None
Sample VB	<pre> Dim newTime As New System.DateTime newTime = System.DateTime.Now newTime = newTime.AddMinutes(50)  Dim followUpTime As Nullable(Of System.DateTime) followUpTime = newTime  ' the followUpTime is in local time Dim isUTC As Boolean = False Dim comment As New String("New FollowUp") </pre>

	<pre>' move the file in the follow up folder doc.followUp(fileId, followUpTime, followUpTime.HasValue, isUTC, comment)</pre>
Explanation	<p>We create a DateTime object with the current time, adding 50 minutes. This time will then be used for the resubmission time, where the time value is in local time, so UTC has been set to false. Because followUpTime is, according to its type, nullable, the call in VB requires the followUpTime.HasValue parameter, which tells the interface whether a value has actually been allocated or not. After allocating a comment, the followUp method of a DOCUMENTS object will be started and the data file will be moved to the resubmission folder.</p>
Comment	<ul style="list-style-type: none"> <li>Depending on how the classes are generated from the wsdl, the method's signature may vary.</li> <li>If the DOCUMENTS file already has a resubmission time and a new time should be set, the time interval must be at least 15 minutes.</li> <li>Only absolute times are allowed; whereas relative times such as "+1 hour" are not.</li> </ul>

#### Documents.GetAutoText

Description	Determines auto texts (e.g. %createdAt%) for a specific DOCUMENTS file.
Input parameter	<p><b>fileId</b>: 1 string DOCUMENTS file for which auto texts are determined. An empty string can also be used.</p> <p><b>autoTextNames</b>: 0..n string</p>
Output parameter	None
Return value	<p><b>autoTextValues</b>: 0..n string Stringarray including the determined auto texts.</p>
Sample VB	<pre>'we have access to a fileId  Dim autoTextNames As New List(Of String) Dim autoTextValues As String()  autoTextNames.Add("%fileOwner%") autoTextNames.Add("%title%") autoTextNames.Add("%fileType%")  autoTextValues = doc.GetAutoText(fileId, autoTextNames.ToArray())  Dim i As Integer = 0 For Each s As String In autoTextValues     Console.WriteLine("{0} - {1}", autoTextNames.Item(i), s)     i = i + 1 Next</pre>
Explanation	Using a fileId, the auto texts are read and output for the file title, file type and file owner.
Comment	An overview of auto texts can be found in separate documentation: " <i>DOCUMENTS AutoTexts</i> ". Some of the auto texts cannot be used in Soap access; this particularly applies to section 3 "AutoTexts for Enumeration Fields"

	If AutoTexts are to be determined from global information and system variables, such as %currentDate%, %currentWeekday% oder %runscript:Scriptname% , an empty string "" can be used as fileld because the determined values do not depend on the values of an actual DOCUMENTS file.
--	---

### Documents.getDocument

Description	Get data and document information on a document.
Input parameter	<b>fileld</b> : 1 string The unique technical name of the file. You can also use an Archive file key. The Docid documents can be determined via Documents.getFileInfo. <b>docld</b> : String The unique technical name of the document.
Output parameter	<b>docld</b> : 1 string Insofar as no error has occurred, the parameter with the same name will be returned by the call. <b>size</b> : 1 Integer File size in bytes. <b>mime</b> : 1 string Mime type of file, if known. <b>data</b> : 1 Base64 In VB Array of Byte Document content.
Return value	<b>name</b> : 1 string File name including extension.
Sample VB	<pre>Dim theFileName As New String(String.Empty)  Dim size As New Integer Dim mime As New String(String.Empty) Dim data() As Byte = Nothing  ' get the document theFileName = doc.getDocument(fileld, docid, size, mime, data) Console.WriteLine("Name: {size: {mime: {2}}", theName, size, mime)  Dim path As New String(theFileName) fs = System.IO.File.Create(path) fs.Write(data, 0, data.Length) fs.Close()</pre>
Explanation	It is assumed that the fileld of the DOCUMENTS file and the docld of the document to be extracted are known. The byte array will be filled with the document data on calling the getDocument method. File.Create is used to create a new data file on the file system and to write the bytes from the data array to the data file.
Comment	

### Documents.getFileId

Description	Determines the file id of a temporary search hit for an archive file.
Input parameter	<b>Key</b> 1 String The Key of an archive file
Output parameter	None
Return value	<b>fileld</b> : 1 String The file id of the correspondent temporary search hit.

Sample VB	<pre> Try     Dim archives As New List(Of String)     archives.Add("ftRecord@peachitStore1")      ' additonal filetypes to be searched     Dim filetypes As New List(Of String)     filetypes.Add("ftRecord")      Dim columns As New List(Of String)     columns.Add("Hit_ArchiveKey")      ' the filter for a fulltext search     Dim filter As New String(String.Empty)     ' all files     filter = "Search_Fulltext~'fa"     Dim sort As New String(String.Empty)     ' Search archive     Dim hitl As New HitList     hitl = doc.report3(filetypes.ToArray, archives.ToArray,         columns.ToArray, filter, sort, "Standard")      Dim rows As Integer = hitl.rows     Dim cols As Integer = hitl.columns     If Not hitl Is Nothing Then         Dim hds() As HitData = hitl.hit         For i As Integer = 0 To rows - 1             Dim hd As HitData = hds(i)             For j As Integer = 0 To cols - 1                 If Not String.IsNullOrEmpty(hd.column(j)) Then                     Console.WriteLine(New String("-", 10))                     Console.WriteLine("Archived file {0} ", hd.column(j))                     Dim fileId As String                     fileId = doc.getFileId(hd.column(j))                     Console.WriteLine("File id {0} ", fileId)                 End If             Next         Next i     End If     Catch ex As Exception         Console.WriteLine("Message: {0}", ex.Message)     End Try </pre>
Explanation	An EAS archive search will be performed and the archive file key of a search hit will be determined. For this, the file id of the temporary search hit in DOCUMENTS will be determined and printed on the console.
Comment	This method is available since WSDL DOCUMENTS-4.0.1827.

#### Documents.getFileInfo

Description	Determine file data to field values, attached documents and related file types.
Input parameter	<b>fileId</b> : 1 string The unique label of the file to be analyzed. An Archive file key can also be used. <b>allFields</b> : 1 Boolean

	<p>Indicates whether all fields should be fetched.</p> <p><b>allAttributes:</b> 1 Boolean If true, extended fields such as "DlcFile_Owner" will also be fetched. See Appendix 5.2 for more information.</p> <p><b>wishedFieldNames:</b> 0..n string The names of the fields to be output. The list will be ignored if allFields has the value "true".</p>
Output parameter	<p><b>fileTypeId:</b> 1 string ID of file type to which the DOCUMENTS file belongs.</p> <p><b>fileTypeName:</b> 1 string Name of file type for this DOCUMENTS file.</p> <p><b>fileTypeLabel:</b> 1 string GUI-label of file type on the Web interface.</p> <p><b>documents:</b> 0..n DocInfo The documents of the DOCUMENTS file including size, name and ID.</p>
Return value	<p><b>fieldvalues:</b> 0..n FieldData Contains the name/value pairs for the individual file fields.</p>
Sample VB	<pre> Dim fileStatuses() As FileStatus = Nothing  doc.getInbox(fileStatuses)  If fileStatuses.Length &lt;= 0 Then     Console.WriteLine("No files found")     Return True End If  Dim fileId As New String(fileStatuses(0).fileId)  Dim allFields As Boolean = True Dim allAttributes As Boolean = True Dim wishedFieldNames As New List(Of String)  Dim fileTypeId As New String(String.Empty) Dim fileTypeNames As New String(String.Empty) Dim fileTypeLabel As New String(String.Empty)  Dim fds() As FieldData = Nothing Dim docinfo() As DocInfo = Nothing  ' getFileInfo fds = doc.getFileInfo(fileId, allFields, allAttributes, wishedFieldNames.ToArray, docinfo, fileTypeId, fileTypeNames, fileTypeLabel)  Console.WriteLine("FileTypeId: {0} FileName: {1} FileTypeLabel", fileTypeId, fileTypeNames, fileTypeLabel) If Not fds Is Nothing Then     For Each fd As FieldData In fds ' print fielddata         Console.WriteLine("Field: {0} Value: {1}", fd.name, fd.value)     Next End If ' fds If Not docinfo Is Nothing Then ' print docinfo     For Each di As DocInfo In docinfo         Console.WriteLine("DocName: {0} DocSize: {1} DocId: {2}", </pre>

	di.name, di.size, di.id) Next Else Console.WriteLine("No docinfo") End If 'docinfo
Explanation	We get a DOCUMENTS file from the logged-in user's Inbox. For this file we get the information on all available fields, the extended fields, the file types, and the attached documents. The respective values are written via the console.
Comment	If no field values or documents are output, the output parameters <b>fieldvalues</b> or <b>documents</b> will become complete obsolete.

### Documents.GetFilesInfo

Description	Determine file data for field values, attached documents and related file type.
Input parameter	<b>fileId</b> : 1 string The unique label of the file to be analyzed. <b>allFields</b> : 1 Boolean Indicates whether all fields should be fetched. <b>allAttributes</b> : 1 Boolean If true, extended fields such as "DlcFile_Owner" will also be fetched. See Appendix 5.2 for more information. <b>wishedFieldNames</b> : 0..n string The names of the fields to be output. The list will be ignored if allFields has the value "true". <b>failOnAllErrors</b> 1 Boolean If true, an exception will be thrown when an error occurs, and processing will terminate.
Output parameter	None
Return value	<b>fileinfos</b> : 0..n <b>FileInfo</b> Determines the <b>FileInfo</b> objects belonging to the DOCUMENTS files which encapsulate the detailed information on the DOCUMENTS files.
Sample VB	<pre> Dim fileStatuses() As FileStatus = Nothing Dim fids As New List(Of String) idFolder = doc.getInbox(fileStatuses)  ' Now iterate through that array and collect the filelds For Each fs As FileStatus In fileStatuses     fids.add(fs.fileId) Next  Dim allFields As Boolean = True Dim allAttributes As Boolean = True Dim wishedFieldNames As New List(Of String) Dim documents() As DocInfo = Nothing Dim fileTypeNames As New String(String.Empty) Dim failOnAllErrors As Boolean = False Dim fis() As FileInfo = Nothing  ' getFilesInfo fis = doc.GetFilesInfo(fids.ToArray, allFields, allAttributes, wishedFieldNames.ToArray, failOnAllErrors) For Each fi As FileInfo In fis </pre>



	<pre> Console.WriteLine("FileName: {0}, fi.fileName)  For Each fd As FieldData In fi.fieldvalues     Console.WriteLine("Field: {0} Value: {1}", fd.name, fd.value) Next For Each docinfo As DocInfo In fi.documents     Console.WriteLine("DocName: {0} DocSize: {1}, docinfo.name,         docinfo.size, docinfo.id) Next Next </pre>
Explanation	We get the DOCUMENTS files from the logged-in user's Inbox, and collect their IDs. We get the information for all fields as well as the attached documents on these DOCUMENTS files. In doing so, the values for the other DOCUMENTS files should also be determined in case of errors that may occur, e.g. "File not found". The respective values are written via the console.
Comment	See also Documents.getFileInfo.

#### Documents.getFileTypes

Description	Determines the file types for the logged-in user.
Input parameter	<b>ignoreRights</b> : 1 Boolean Specifies whether the names of those file types for which the user account used has no read permissions are also to be output.
Output parameter	None
Return value	0..n FileTypeShortDescr An indeterminate number of FileTypeShortDescr objects that contain the name and ID of the file type.
Sample VB	<pre> Dim doc as New DOCUMENTS Try     Dim ftsd() As FileTypeShortDescr     ftsd = doc.getFileTypes(ignoreRights)     For Each sd As FileTypeShortDescr In ftsd         Console.WriteLine("File Type Name: {id: {1}", sd.name, sd.id)     Next Catch ex AS Exception     Console.WriteLine("Logout Message: {0}", ex.Message) End Try </pre>
Explanation	The FileTypes are determined for a DOCUMENTS object disregarding specific user rights, and storing them in an array of FileTypeShortDescr objects, followed by an iteration through the array and outputting ID and name of the determined FileType to the console.
Comment	

#### Documents.getFilingPlans

Description	Determines the filing plans for the logged-in user.
Input parameter	None
Output parameter	None
Return value	0.. n <b>FilingPlan</b> A list of filing plans objects.
Sample VB	<pre> Dim doc as New DOCUMENTS </pre>

	<pre> Try     Dim filingPlans() As FilingPlan = Nothing     filingPlans = doc.getFilingPlans()     For Each fpa As FilingPlan In filingPlans         Console.WriteLine("fp: {0} {1}", fpa.name, fpa.label)     Next     Return True Catch ex As Exception     Console.WriteLine("getFilingPlans: {0}", ex.Message) Return False End Try </pre>
Explanation	The filing plans of a given DOCUMENTS object are determined and the name as well as the label will be printed on the console.
Comment	This method is available since WSDL DOCUMENTS-4.0.1827.

#### `Documents.getFilingPlanXML`

Description	Determines an XML-report of a concrete filing plan.
Input parameter	<b>name</b> 1 String Technical name of the filing plan, for which an XML-report should be returned.
Output parameter	None
Return value	1 StringFilingPlan The XML-report in UTF-8 encoding for the particular filing plan according to the schema from section 5.7.
Sample VB	<pre> Dim doc as New DOCUMENTS Try     Dim filingPlans() As FilingPlan = Nothing     filingPlans = doc.getFilingPlans()     For Each fpa As FilingPlan In filingPlans         Dim xml As New String(String.Empty)         xml = doc.getFilingPlanXML(fpa.name)         Console.WriteLine("FilingPlan: {0} ", fpa.name)         Console.WriteLine(xml)     Next Catch ex As Exception     Console.WriteLine("Message: {0}", ex.Message) End Try </pre>
Explanation	The technical names of all filing plans are determined und then each of the respective XML-representations is printed on the console.
Comment	This method is available since WSDL DOCUMENTS-4.0.1827.

#### `Documents.getFolderStructure`

Description	List folder structure within a default folder.
Input parameter	<b>folderType</b> : 1 string The type label of a personal default folder for which to determine the structure. A list of available folder types can be found in the Appendix <b>Label for personal default folders</b> .
Output parameter	None
Return value	<b>baseFolder</b> : 1 FolderDescription
Sample VB	<pre> Dim fd As FolderDescription fd = doc.getFolderStructure("Favourites") If fd.hasSubFolders Then </pre>

	<pre> For Each fod As FolderDescription In fd.subFolders     Console.WriteLine("Subfolder: {0}", fod.label) Next Else     Console.WriteLine("No subfolders") End If </pre>
Explanation	A description of the folder structure is requested for the "Favorites" folder of a given DOCUMENTS object. If the folder has other subfolders (hasSubFolders), their labels will be written from the console.
Comment	This function does not guarantee that the output order of folders is exactly as displayed on the Web interface.

#### Documents.getInbox

Description	Determines the DOCUMENTS files and status information of these files residing in the logged-in user's Inbox.
Input parameter	None
Output parameter	<b>fileStatuses:</b> 0..n <b>FileStatus</b> An indeterminate number of <b>FileStatus</b> objects, including information on FileId, workflow step and routing status.
Return value	idFolder: 1 string Inbox ID.
Sample VB	<pre> Try     Dim fileStatuses() As FileStatus = Nothing      Dim idFolder As New String(String.Empty)      idFolder = doc.getInbox(fileStatuses)     ' How many files?     Console.WriteLine("Inbox countFiles: {0}", fileStatuses.Length())     Return True Catch ex As Exception     Console.WriteLine("getInbox Message: {0}", ex.Message)     Return False End Try </pre>
Explanation	The Inbox of an already logged-in user will be fetched from a DOCUMENTS object, and the number of DOCUMENTS files will be printed on the console.
Comment	

#### Documents.getMonitor

Description	Determines the routing history (workflow steps) for a DOCUMENTS file.
Input parameter	<b>fileId:</b> ID of the DOCUMENTS file, the monitor entries of which should be fetched.
Output parameter	None
Return value	<b>monitorEntries:</b> 0..n <b>MonitorEntry</b> An indeterminate number of monitor entries that reflect the respective step in the routing history.
Sample VB	<pre> Dim monitorentries() As MonitorEntry = Nothing monitorentries = doc.getMonitor(theFileId) </pre>

	<pre> 'iterates through all MonitorEntries For Each moe As MonitorEntry In monitorentries   'Print out the information of each entry   Console.WriteLine()   Console.WriteLine("Status: {0} fileOK: {1}", moe.status, moe.fileOk)   Console.WriteLine("Executive: {0}", moe.executive)   Console.WriteLine("EntryDate: {0} ResponseDate: {1}", moe.entryDate, moe.responseDate)   Console.WriteLine("Task: {comment: {1}", moe.task, moe.comment) Next </pre>
Explanation	The DOCUMENTS object and file ID must already be declared and instantiated or known. On the doc object, the get monitor method is called and iterated through the returned MonitorEntry array, where the respective properties are written to the console for each entry.
Comment	see also <b>Documents.getWorkflowSteps</b>

### Documents.getProperty

Description	The method reads the properties stored in the file type of a DOCUMENTS file.
Input parameter	<b>fileId</b> : 1 string Technical name of file <b>propertyNames</b> : 0..n string Array including the names of the properties to be read for the DOCUMENTS file. The property names must start with a leading \$.
Output parameter	
Return value	<b>propertyValues</b> : 0..n string String array including the determined values that corresponds to the names.
Sample VB	<pre> Dim idFolder As New String(String.Empty) idFolder = doc.getInbox(fileStatuses) ' Iterate through all inbox files If fileStatuses.Length &gt; 0 Then   For Each fs As FileStatus In fileStatuses     Dim propertyNames As New List(Of String)     propertyNames.Add("\$Test")      Dim propertyValues() As String     propertyValues = Nothing     propertyValues = doc.getProperty(fs.fileId, propertyNames.ToArray)      Dim i As Integer = 0     For Each val As String In propertyValues       Console.WriteLine("Name: {0} Value: {1}", propertyNames.Item(i), val)       i = i + 1     Next   Next End If </pre>
Explanation	The user's Inbox is determined and on each of these DOCUMENTS files the value of the "Test" property is determined and displayed in the command line.
Comment	

## Documents.getSentFolder

Description	Determines the DOCUMENTS files and status information of these files residing in the SentFolder of the logged-in user.
Input parameter	None
Output parameter	<b>fileStatuses:</b> 0..n <b>FileStatus</b> An indeterminate number of <b>FileStatus</b> objects, including information on FileId, workflow step and routing status.
Return value	<b>idFolder:</b> 1 string The Id of the SentFolder.
Sample VB	<pre> Try   Dim fileStatuses() As FileStatus = Nothing    Dim idFolder As New String(String.Empty)    idFolder = doc.getSentFolder(fileStatuses)   For Each fs As FileStatus In fileStatuses     Console.WriteLine("FileStatusFileId: {0} IdWorkflowStep: {status: {2}", fs.fileId, fs.idWorkflowStep, fs.status)   Next   Return True Catch ex As Exception   Console.WriteLine("getSentFolder Message: {0}", ex.Message)   Return False End Try </pre>
Explanation	The SentFolder of the current user is fetched on a DOCUMENTS object and the FileStatus information is displayed for each object residing in the return array.
Comment	

## Documents.getTasks

Description	Determines the tasks for a DOCUMENTS file.
Input parameter	<b>fileId:</b> 1 string Technical name of file.
Output parameter	None
Return value	<b>tasks:</b> 0..n string The tasks for this DOCUMENTS file.
Sample VB	<pre> Try   Dim fileStatuses() As FileStatus = Nothing    Dim idFolder As New String(String.Empty)    ' Get the sentfolder   idFolder = doc.getSentFolder(fileStatuses)   ' Now iterate through that array   For Each fs As FileStatus In fileStatuses     Dim tasks() As String = Nothing      ' get the tasks for this file     tasks = doc.getTasks(fs.fileId)     If tasks.Length &gt; 0 Then       For Each s As String In tasks         Console.WriteLine("Task: {0}", s)       Next     End If   Next End Try </pre>

	End If Next Return True  Catch ex AS Exception Console.WriteLine("GetTask Message: {0}", ex.Message) Return False End Try
Explanation	Initially, we will get the user's SentFolder. The ID of the respective DOCUMENTS file is determined from the returned FileStatus objects and, using this ID, the DOCUMENTS file's tasks, which are available as a string array, are fetched. If the array contains at least one entry, the entire array will be iterated through and the individual tasks will be written one after another to the console.
Comment	

## Documents.getWorkflowPattern

Description	Determines the workflows contained in the system.
Input parameter	None
Output parameter	None
Return value	workflowPattern: 0..n <b>WorkflowPattern</b> An indeterminate number of <b>WorkflowPattern</b> objects including information on ID and name of workflow.
Sample VB	<pre> Try ' here we store the returned WorkflowPattern Dim wps() As WorkflowPattern  'fetch them wps = doc.getWorkflowPattern()  'iterate through all workflows For Each wp As WorkflowPattern In wps     ' and print the name     Console.WriteLine("Name: {0}", wp.nameWorkflowPattern) Next  Return True Catch ex As Exception     Console.WriteLine("WorkflowPattern Message: {0}", ex.Message) Return False End Try </pre>
Explanation	The method for determining the workflows is called for an already logged-in user on a DOCUMENTS object. The name and ID for each object are written to the console in the returned array of <b>WorkflowPattern</b> objects.
Comment	

## Documents.getWorkflowSteps

Description	Determines the workflow steps depending on the specified DOCUMENTS file.
Input parameter	<b>fileId</b> : 1 string Technical name (ID) of the DOCUMENTS file.
Output parameter	None
Return value	WorkflowSteps: 0..n <b>WorkflowStep</b>
Sample VB	<pre> Dim fileStatuses() As FileStatus = Nothing  doc.getInbox(fileStatuses) For Each fs As FileStatus In fileStatuses      Dim wfsteps() As WorkflowStep     ' get the workflowid and get the Steps     wfsteps = doc.getWorkflowSteps(fs.fileId)      For Each wstep As WorkflowStep In wfsteps         Console.WriteLine("Entry: {0}", wstep.entryDate)         Console.WriteLine("Finish: {0}", wstep.finishDate)     </pre>

	Next Next
Explanation	The user's Inbox is read and, based on the file ID, the routing history is read for each file: in doing so, the arrival and initial time are output for each workflow step to the console.
Comment	see also <b>Documents.getMonitor</b>

#### `Documents.listPossibleActions`

Description	Determines the currently allowed workflow steps of a DOCUMENTS file.
Input parameter	<b>fileId</b> : 1 string File ID.
Output parameter	None
Return value	workflowActions: 0..n <b>WorkflowAction</b> An indeterminate number of <b>WorkflowAction</b> objects including information on ID and label of the allowed user action.
Sample VB	<pre> doc.getInbox(fileStatuses) ' Now iterate through all inbox files If fileStatuses.Length &gt; 0 Then     For Each fs As FileStatus In fileStatuses         Console.WriteLine(New String("-", 10))         Console.WriteLine("fileId : {0}", fs.fileId)          Dim workflowActions() As WorkflowAction = Nothing         ' get all actions         workflowActions = doc.listPossibleActions(fs.fileId)         For Each wa As WorkflowAction In workflowActions             ' and print them             Console.WriteLine("Label: {id: {1}}", wa.label, wa.id)         Next     Next Else     Console.WriteLine("No files found") End If </pre>
Explanation	The logged-in user's Inbox is read, and an array including the allowed action is fetched for each file (workflowActions()). For each of these actions, the ID and its label are then written to the console.
Comment	

#### `Documents.listPublicFolders`

Description	Determines the public folders.
Input parameter	<b>typeOption</b> : 1 Integer with value 0 This parameter is reserved for future use and must be 0 now.  <b>sortOption</b> : 1 Integer with value 0 This parameter is reserved for future use in the future and must be 0 now.
Output parameter	None
Return value	folders: 0..n <b>FolderDescription2</b>
Sample VB	Dim fd() As FolderDescription2



	<pre> fd = doc.listPublicFolders(typeOption, sortOption) For Each fd2 As FolderDescription2 In fd     Console.WriteLine("{0} {1}", fd2.label, fd2.name) Next ' fd2 </pre>
Explanation	The code determines the top level in the structure of the public folders.
Comment	Only the folders being released and visible on the web interface for the logged-in user are returned.

### Documents.login

Description	Performs login via the DOCUMENTS proxy.
Input parameter	<p><b>user:</b> String User name under which to perform login.</p> <p><b>principal:</b> String Principal for which to log in the user.</p> <p><b>passwd:</b> String Password via which the user authenticates himself against the proxy.</p> <p><b>code:</b> String/ <b>empty string</b> Reserved for future extensions. An empty string must be passed in this version.</p> <p><b>locale:</b> String Abbreviation for Portal language "de", "en", etc. in which GUI-label, error messages, etc. are output.</p>
Output parameter	None
Return value	<p><b>session:</b> 1 String The return value contains the SessionID assigned to the user in the proxy.</p>
Sample VB	<pre> Dim doc as New DOCUMENTS Dim session As New String(String.Empty) Try     session = doc.login("schreiber","peachit","willi","", "de") Catch ex AS Exception     Console.WriteLine("Login Message: {0}", ex.Message) End Try </pre>
Explanation	The login method is called on a DOCUMENTS object and the determined session is saved in a string. In case of a possible exception the exception text will be written to the console.
Comment	<p>Login is required at the start of the application. Logout needs to be performed only at the end of the application, i.e. it is not required for each individual function call. You should perform the logout in a <b>Finally</b> block, so that logout is ensured also after a catch. Otherwise, the user is considered logged-in until timeout in the proxy, which prevents renewed login.</p> <p>Once login has been performed, logout must also be performed because the session in the proxy cannot be immediately unlocked.</p> <p>Competing logins with the proxy, i.e. two or more logins for the same user are not supported.</p> <p>If login cannot be performed because the user/principal is unknown, an error will be triggered.</p> <p>If the user is to be logged in on a different account, you</p>

	need to use the <b>Documents.trustedLogin</b> function.
--	---

### Documents.logout

Description	Performs logout for the logged-in user.
Input parameter	None
Output parameter	None
Return value	None
Sample VB	<pre>Dim doc as New DOCUMENTS Dim session As New String(String.Empty) Try     doc.logout() Catch ex AS Exception     Console.WriteLine("Login Message: {0}", ex.Message) End Try</pre>
Explanation	The logout method is started on a DOCUMENTS object. In case of a possible exception the exception text will be written to the console.
Comment	If login has been performed, logout must also be performed, because otherwise the session in the proxy will not be immediately unlocked. See also Documents.login.

### Documents.report

Description	Search for documents of a file type and outputting field values as HTML table.
Input parameter	<p><b>filetypes:</b> 0..n string A list of file type names, can be used instead of <b>filetype</b>.</p> <p><b>archives:</b> 0..n string List of Search resource id for an archive. However this function does not support any archives, thus this list should be empty.</p> <p><b>filter:</b> 1 string A search condition, as described in the Appendix under Syntax description for filter expressions; this may remain empty for displaying all files of the transferred type insofar as the user account used has the corresponding permissions.</p> <p><b>sort:</b> 1 string A sort criterion in <b>Field name+</b> notation for sorting in ascending order or <b>Field name-</b> notation for sorting in descending order, or an empty string. Only sorting by a single field is currently possible. Moreover, the field by which to sort should also exist in the <b>columns</b> list.</p> <p><b>columns:</b> 0..n string The names of the fields to be output. Besides index fields of the file type, file attributes such as the title may also be queried. See Appendix 5.2 for more information.</p>
Return value	<p><b>report:</b> String No self-contained and therefore valid HTML page will be output; instead, only a &lt;table&gt; element).</p>
Sample VB	<pre>' The filetypes to be searched Dim filetypes As New List(Of String) filetypes.Add("ftEmployee")</pre>

	<p>' The fields whose values should be returned</p> <pre>Dim columns As New List(Of String) columns.Add("hrLastName")  Dim archives As New List(Of String)  Dim filter As New String(String.Empty) filter = "hrLastName~'Sc*" "hrLastName~'Sc*"  Dim sort As New String(String.Empty)  ' sort in reverse alphabetic order ' the field in the sort string has also to be in the columns-list sort = "hrLastName~"</pre> <p>Dim report As New String(String.Empty)</p> <pre>report = doc.report(filetypes.ToArray, archives.ToArray, columns.ToArray, filter, sort)</pre>
Explanation	<p>The "ftEmployee" file type is searched for all employees whose last names contain the letter combination "sc". The hits are written to the table element descending alphabetical order, depending on the database, and displayed on the console.</p>
Comment	<p>When directly parsing the string returned by the server, it should be noted that some characters of the HTML report are re-encoded through embedding into XML, e.g. "&gt;" in "&amp;gt;", etc. An <b>id</b> parameter containing a string with a unique ID to the found file is added to the output report's &lt;tr&gt; elements.</p> <p>To form the filter expressions, see Appendix under <b>Syntax description</b> for filter expressions.</p> <p>This function is not available for archive files.</p>

## Documents.report2

Description	<p>Search for DOCUMENTS files of one or more file types by outputting a hit list.</p>
Input parameter	<p><b>filetypes</b>: 0..n string A list of file type names</p> <p><b>archives</b>: 0..n string A list of Search resource id for an archive. EE.x archive is not supported by this function.</p> <p><b>filter</b>: 1 string A search condition, as described in Appendix 1; this may remain empty for displaying all files of the transferred type insofar as the user account used has the corresponding permissions.</p> <p><b>sort</b>: 1 string A sort criterion in <b>Field name+</b> notation for sorting in ascending order or <b>Field name-</b> notation for sorting in descending order, or an empty string. Only sorting by a single field is currently possible. Moreover, the sort field must also be present with the <b>columns</b> parameter.</p> <p><b>columns</b>: 0..n String The names of the fields to be output. Besides index fields of the file type, file attributes such as the title may also be</p>

	<p>queried. The latter is implemented via reserved labels as listed in Appendix 5.2. When adding "<b>Hit_id</b>" as a column, that column will list the file ID or, for archive files, the archive file key (see 5.1). When adding "<b>Hit_ArchiveKey</b>" as a column, that column will only list the key of each archive file.</p>
Output parameter	None
Return value	<p><b>hitlist: 1 HitList</b> A hit list including the result rows and columns.</p>
Sample VB	<pre> Dim filetypes As New List(Of String) filetypes.Add("ftRecord") ' The fields whose values should be returned Dim columns As New List(Of String) columns.Add("DlcFile_Id") columns.Add("DlcFile_Created")  Dim archives As New List(Of String)  ' We are searching for files which are created today or yesterday Dim filter As New String(String.Empty) Dim today As String = DateTime.Now.ToShortDateString  Dim yesterday As String = DateTime.Now.AddDays(- 1).ToShortDateString filter = "Search_DateFrom&gt;=" + yesterday + " AND Search_DateUntil&lt;=" + today + ""  ' and sort them in order of their creation date Dim sort As New String(String.Empty) sort = "DlcFile_Created"  'search Dim hitl As New HitList hitl = doc.report2(filetypes.ToArray, archives.ToArray, columns.ToArray, filter, sort)  Dim rows As Integer = hitl.rows Dim cols As Integer = hitl.columns  If Not hitl Is Nothing Then     Dim hds() As HitData = hitl.hit     For i As Integer = 0 To rows - 1         Console.WriteLine(New String("-", 10))          Dim hd As HitData = hds(i)         For j As Integer = 0 To cols - 1             Console.WriteLine("Hit {0} Col {1} Value {2}", i + 1, j + 1, hd.column(j))         Next     Next i End If ' htil </pre>

Explanation	The "ftRecord" file type is searched for the DOCUMENTS files that were created yesterday or today. For this, the corresponding DateTime objects are generated for the filter, and used in the filter. The hits are to be sorted by the creation timestamp of the DOCUMENTS file. The resulting hit list is processed by rows and the values for the two requested columns are written to the console.
Comment	Due to previous technical difficulties with outputting arrays containing zero elements, the function may yet output an empty string instead of an array in its <b>report</b> version in case no files have been found. Applications should therefore check the report's file type prior to accessing the report. To form the filter expressions, see Appendix under <b>Syntax description</b> for filter expressions.

### `Documents.report3`

Description	Search for DOCUMENTS files of one or more file types in one or more archives considering a specific hit list schema.
Input parameter	<b>filetypes, filter, sort,columns:</b> <b>See Documents.report2</b> The <b>columns</b> to be determined must be created in the hit list schema of the ENTERPRISE archive. DlcFile_Id, Hit_ArchiveKey and Hit_Id can only be used in the <b>columns</b> , not with the <b>filter</b> or <b>sort</b> parameters. <b>archives:</b> : 0..n string A list of Search resource id for an archive. <b>hitlistname</b> : 1..1 string For EASY archives, the name of the hit list schema created in the archives. For EAS archives, you may not enter any value here.
Output parameter	None
Return value	<b>hitlist: 1 HitList</b> A hit list including the result rows and columns.

Sample VB	<pre> Dim filetypes As New List(Of String)  ' The fields whose values should be returned ' All of them has to be defined in the hitlistname  Dim columns As New List(Of String) columns.Add("DlcFile_Id") columns.Add("DlcFile_Created")  Dim archives As New List(Of String) archives.Add("ftRecord@peachitStore1")  Dim hitlistname as String hitlistname = ""  Dim filter As New String(String.Empty) Dim today As String = DateTime.Now.ToShortDateString  Dim yesterday As String = DateTime.Now.AddDays(- 1).ToShortDateString filter = "Search_DateFrom&gt;=" + yesterday + " AND Search_DateUntil&lt;=" + today + ""  ' and sort them in order of their creation date Dim sort As New String(String.Empty) sort = "DlcFile_Created"  'search Dim hitl As New HitList hitl = doc.report3(filetypes.ToArray, archives.ToArray, columns.ToArray, filter, sort)  Dim rows As Integer = hitl.rows Dim cols As Integer = hitl.columns  If Not hitl Is Nothing Then     Dim hds() As HitData = hitl.hit     For i As Integer = 0 To rows - 1         Console.WriteLine(New String("-", 10))          Dim hd As HitData = hds(i)         For j As Integer = 0 To cols - 1             Console.WriteLine("Hit {0} Col {1} Value {2}", i + 1, j + 1, hd.column(j))         Next     Next i End If ' htill </pre>
Explanation	<p>In the "ftRecord@peachitStore1" archive, the "ftRecord" file type is searched for DOCUMENTS files that were created yesterday or today. For this, the corresponding DateTime objects are generated for the filter, and used in the filter. The hits are to be sorted by the creation timestamp of the DOCUMENTS file. The resulting hit list is processed by rows and the values for the two requested columns are written to the console.</p>

Comment	See <a href="#">Documents.report2</a> .
---------	---

## Documents.runScript

Description	Run a Portal script.
Input parameter	<p><b>fileId</b>: 1 string The technical name of the DOCUMENTS file to be transferred to the script.</p> <p><b>name</b>: String Name of the script to be called.</p> <p><b>paramList</b>: 0..n string Input parameter for the script as a list. You need to add a parameter name, in pairs, and after that the related value, to the list.</p>
Output parameter	<p><b>errorMsg</b>: String Error text for failed call or erroneous execution of the script.</p> <p><b>returnValue</b>: 1 string The return value of the script.</p>
Return value	<p><b>returnStatus</b> : Integer On starting the operation successfully, this value is 0, otherwise it is an error code.</p>
Sample VB	<pre>Dim fileStatuses() As FileStatus = Nothing  doc.getInbox(fileStatuses) ' Now iterate through all inbox files If fileStatuses.Length &lt;= 0 Then     Console.WriteLine("No files found")     Return False End If  Dim fileId As New String(fileStatuses(0).fileId) Dim name As New String("test")  Dim errorMsg As New String(String.Empty) Dim returnValue As New String(String.Empty)  Dim paramList As New List(Of String) paramList.Add("msg") paramList.Add("This message should be returned")  Dim retStatus As New Integer retStatus = doc.runScript(fileId, name, paramList.ToArray, errorMsg, returnValue) Console.WriteLine("fileId: {0} scriptname: {1}", fileId, name) Console.WriteLine("errorMsg: {errorMsg}") Console.WriteLine("retStatus: {returnValue: {1}}", retVal, returnValue)</pre>
Explanation	<p>The requirement is that a script named "test" including an input parameter named "msg" of the String type and the "Message" default setting have been set. You need to enter "return msg;" as the script code.</p> <p>The fileId of the first DOCUMENTS file is determined from the user's Inbox. Afterwards, the necessary data is assembled for the script call. In doing so, a string list for the parameters to be transferred is built which contains the "msg" and "This message should be</p>

	returned" entries. The output parameters and the retStatus are written to the console after starting the script.
Comment	Available on otrisPortal 5.0o or ELC3.50o or later. Script parameters must currently always be transferred as strings. The return value of the script is also output as a string.

#### Documents.sendFileAdHoc

Description	Directly sending a DOCUMENTS file
Input parameter	<b>fileId</b> : String Technical name of file <b>receivers</b> : 1..n string The name of the users or groups to which to send the DOCUMENTS file. You need to specify at least one recipient. <b>sendMode</b> : String The send type. Valid values are: <b>sequential</b> (one after the other) and <b>parallel_info</b> (concurrently for information). <b>task</b> : String Task specification for the recipients of the DOCUMENTS file. <b>backWhenFinished</b> : Boolean Indicates whether the DOCUMENTS file should be returned to your own user account after the cycle.
Output parameter	None
Return value	None
Sample VB	<pre>Dim fileStatuses() As FileStatus = Nothing  doc.getInbox(fileStatuses)  If fileStatuses.Length &lt;= 0 Then     Console.WriteLine("No files found")     Return False End If  Dim fileId As New String(fileStatuses(0).fileId)  Dim sendMode As New String("parallel_info") Dim task As New String("Info by sendfile") Dim backWhenFinished As Boolean = False  ' we send the file to the group Dim receivers As New List(Of String) receivers.Add("Employees") ' send doc.sendFileAdHoc(fileId, receivers.ToArray, sendMode, task, backWhenFinished)</pre>
Explanation	The first DOCUMENTS file is taken from the logged-in user's Inbox and sent in parallel for information to the "Employees" group.

#### Documents.triggerAction

Description	Triggers a user action for a DOCUMENTS file.
Input parameter	<b>fileId</b> : 1 string



	File ID. <b>actionId</b> : String The ID of the action to be triggered. <b>comment</b> : 1 string The edit note to be entered in the routing history.
Output parameter	None
Return value	None
Sample VB	<pre> Dim fileStatuses() As FileStatus = Nothing doc.getInbox(fileStatuses) If fileStatuses.Length &lt;= 0 Then     Console.WriteLine("No files found")     Return False End If Dim fileId As New String(fileStatuses(0).fileId)  Dim workflowActions() As WorkflowAction = Nothing ' get all actions workflowActions = doc.listPossibleActions(fileId)  For Each wa As WorkflowAction In workflowActions     ' and print them     Console.WriteLine("Label: {id: {1}}", wa.label, wa.id)     doc.triggerAction(fileId, wa.id, "Performed TriggerAction")     Console.WriteLine("Action performed: {0} ", wa.label) Exit For Next </pre>
Explanation	The available user actions are determined for the first DOCUMENTS file in the user's Inbox. The information on the first action are output to the console. The doc.TriggerAction triggers this action also for a DOCUMENTS object.
Comment	

#### Documents.startWorkflow

Description	Starts a workflow for a created DOCUMENTS file.
Input parameter	<b>fileId</b> : 1 string File ID. <b>idWorkflowPattern</b> : 1 string ID of the workflow to the started.
Output parameter	None
Return value	None
Sample VB	<pre> ' Get the Inbox Dim fileStatuses() As FileStatus = Nothing doc.getInbox(fileStatuses) If fileStatuses.Length &lt;= 0 Then     Console.WriteLine("No files found")     Return False End If ' And grab the id of the first file Dim fileId As New String(fileStatuses(0).fileId)  Dim wps() As WorkflowPattern ' Get all Workflowpattern wps = doc.getWorkflowPattern() </pre>

	<pre> If wps.Length &lt;= 0 Then     Console.WriteLine("No workflow found")     Return False End If ' And grab the id of the first workflowpattern Dim workflowId As New String(wps(0).idWorkflowPattern)  ' Now start the workflow for this file doc.startWorkflow(fileId, workflowId) </pre>
Explanation	In a DOCUMENTS object you determine the logged-in user's Inbox and read the fileId of the first DOCUMENTS file. Next, the ID of the first workflow is determined by querying the workflow list, and the workflow is started for the DOCUMENTS file.
Comment	To determine the idWorkflowPattern, see <b>Documents.getWorkflowPattern</b> .

#### Documents.testSession

Description	Checks whether a session exists for the current SOAP client
Input parameter	None
Output parameter	None
Return value	<b>valid:</b> 1 Boolean Shows whether a valid session has been set up for the client.
Sample VB	<pre> While(doc.testSession())     doc.logout() End While </pre>
Explanation	A check is made for a DOCUMENTS object on whether a session for the SOAP proxy has been set up for the client. If this is the case, the client will be logged off.
Comment	

#### Documents.trustedLogin

Description	Performs login via the Documentsproxy, where the user is logged under a different account.
Input parameter	<p><b>user:</b> String The user name for which login to a different account should be performed. The "trustedLoginAccount" property including the value "1" must be present for this user via the DOCUMENTS client.</p> <p><b>principal:</b> String Principal for which to log in the user.</p> <p><b>passwd:</b> String Password via which the user authenticates himself against the proxy.</p> <p><b>asUser:</b> String The user to whom to switch.</p> <p><b>code:</b> String/ <b>empty string</b> Reserved for future extensions. An empty string must be passed in this version.</p> <p><b>locale:</b> String Abbreviation for Portal language "de", "en", etc. in which GUI-label, error messages, etc. are output.</p>
Output parameter	None

Return value	<b>session:</b> 1 string The return value contains the SessionID assigned to the user in the proxy.
Sample VB	<pre>Dim doc as New DOCUMENTS Dim session As New String(String.Empty) Try     ' 'schreiber' login as user 'oppen'     session =         doc.trustedLogin("schreiber","peachit","willi","oppen","",de)     Console.WriteLine("TrustedLogin Message: {0}",doc.UserInfo())  Catch ex AS Exception     Console.WriteLine("TrustedLogin Message: {0}", ex.Message) End Try</pre>
Explanation	The trustedLogin method in which user 'schreiber' is logged in as 'oppen' is started on a DOCUMENTS object. In an exception that may occur, e.g. if the 'trustedLoginAccount' property does not exist with the user or is set to a value not equal to '1', the exception text will be written to the console. The UserInfo method output should output 'Bernard Oppen' when successfully switching users.
Comment	The same restrictions apply as with Documents.login

#### Documents.userInfo

Description	Determines the full name of the user for whom the session is currently set on the proxy.
Input parameter	None
Output parameter	None
Return value	<b>name:</b> String Full name of the user, i.e. first and last name, e.g. "Willi Schreiber".
Sample VB	<pre>Dim doc as New DOCUMENTS Dim name as New String(String.Empty) Try     name = doc.userInfo()     Console.WriteLine("userInfo Name: {name}") Catch ex AS Exception     Console.WriteLine("userInfo Message: {ex.Message}") End Try</pre>
Explanation	The userInfo method is started on a DOCUMENTS object to determine the name of the currently logged-in user. This is then written to the console.
Comment	If no user is logged in and therefore no session set up on the proxy, this will trigger an error.

## 3.2 DOCUMENTS classes

### ArchiveStatus

Description	Contains information on how successful archiving a DOCUMENTS file was.
Member	<b>fileid</b> : 1 string The ID of the DOCUMENTS file to be stored in the archive. <b>messages</b> : 1 string An empty string when the DOCUMENTS file has been archived successfully; else an error string.
Comment	Error strings begin with a technical abbreviation which may be followed by additional specifications separated by pipe symbols. The "DlcErrFieldsNotArchived" abbreviation saying some file fields do not exist in the archive structure and, as a result, were omitted, is a special abbreviation in this context. All other "DlcErr..." labels suggest that the DOCUMENTS file has not been archived. If a DOCUMENTS file has already been archived, DOCUMENTS will choose the same target/source archive. Otherwise, an "active archive" must be assigned to the corresponding file type via the Portal Manager.
Used in	Documents.archiveFiles

### DeleteStatus

Description	Contains status information on a delete operation.
Member	<b>id</b> : 1 string The ID of the affected document/DOCUMENTS file. <b>deleted</b> : 1 Boolean Indicates whether the document was deleted successfully. <b>messages</b> : 1 string If an error occurs with a document, an error string will be entered here. Error strings begin with a technical abbreviation which may be followed by additional specifications separated by pipe symbols. The names always begin with the "DlcErr" character string. In case of errors at file level identical error strings will not be output for each individual document; instead, an exception will be triggered.
Used in	Documents.DeleteDocuments Documents.deleteFiles

### DocInfo

Description	Contains the information on the documents attached to a DOCUMENTS file.
Member	<b>id</b> : 1 string Document ID. <b>name</b> : 1 string Document name. <b>comment</b> : 1 String

	<p>Comment on document. ELC 3.60f/ otris Portal 6.0f or later</p> <p><b>size:</b> 1 string Document size with uniform abbreviation.</p>
Used in	FileInfo

### DocUploadData

Description	Contains information of a document to be uploaded.
Member	<p><b>name:</b> 1 string Document name (= file name including extension).</p> <p><b>register:</b> 1 string The name of the document tab to which to assign the document. This parameter may be omitted only when defining default tabs.</p> <p><b>data:</b> 1 base64Binary The file content as base64 encoded string. In VB a byte array is enough for accommodating the file content. Conversion is automatically performed via SOAP.</p>
Used in	<p>Documents.createFile</p> <p>Documents.editFile</p> <p>Documents.createFile2</p>

### DocUploadData2

Description	Contains information of a document to be uploaded.
Member	<p><b>name:</b> 1 string Document name (= file name including extension).</p> <p><b>register:</b> 1 string The name of the document tab to which to assign the document. This parameter may be omitted only when defining default tabs.</p> <p><b>data:</b> 1 base64Binary The file content as base64 encoded string. In VB a byte array is enough for accommodating the file content. Conversion is automatically performed via SOAP.</p> <p><b>replace:</b> 0..1 Boolean Indicates replacing a document in an active file (if applicable) by the same-named document to be uploaded. This parameter is redundant for an archive file.</p> <p><b>versioning:</b> 0..1 Boolean Indicates versioning a document in an active file (if applicable) by the same-named document to be uploaded in case of <b>replace</b> being true. This parameter is also redundant for an archive file.</p>
Used in	Documents.editFile2

### DocUploadData3

Description	Contains information of a document to be uploaded.
Member	<p><b>name:</b> 1 string Document name (= file name including extension).</p> <p><b>register:</b> 1 string The name of the document tab to which to assign the document. This parameter may be omitted only when defining default tabs.</p> <p><b>path:</b> 1 string</p>

	<p>The relative path to the <code>BlobBasePath</code> from <code>docsoaproxy.ini</code> of the file to be uploaded.</p> <p><b>replace:</b> 0..1 Boolean Indicates replacing a document in an active file (if applicable) by the same-named document to be uploaded. This parameter is redundant for an archive file and <code>Documents.createFile3</code>.</p> <p><b>versioning:</b> 0..1 Boolean Indicates versioning a document in an active file (if applicable) by the same-named document to be uploaded in case of <b>replace</b> being true. This parameter is also redundant for an archive file and <code>Documents.createFile3</code>.</p> <p><b>deleteBlob:</b> 0..1 Boolean Indicates whether the file after upload will be deleted.</p>
Used in	<p><code>Documents.createFile3</code></p> <p><code>Documents.editFile3</code></p>

### FieldData

Description	Contains name and value for a file field.
Member	<p><b>name:</b> 1 string Field name</p> <p><b>value:</b> 1 string Field value</p>
Used in	<p><code>Documents.createFile</code></p> <p><code>Documents.editFile</code></p> <p><code>Documents.getFileInfo</code></p> <p><code>FileInfo</code></p> <p><code>Documents.createFile2</code></p> <p><code>Documents.editFile2</code></p> <p><code>Documents.createFile3</code></p> <p><code>Documents.editFile3</code></p>

### FieldDescription

Description	Contains detailed information on the fields of a <code>FileType</code> .
Member	<p><b>name:</b> 1 string The ergonomic field name in the user's locale.</p> <p><b>id:</b> 1 string The technical name of the field.</p> <p><b>fieldlabel:</b> 0..1 string Serves to get the entire label of the field and its ergonomic name in the current user's language by including "fields,fieldlabel" and "fields,fieldlabel.locale" to the 'categories'-parameter, respectively. In case of no label specified for the field this fieldlabel is an empty string for "fields,fieldlabel" and the technical name for "fields,fieldlabel.locale", respectively. For further information see <code>Documents.describeFileType</code>.</p> <p><b>type:</b> 1 string Indicates to which type the field in <code>DOCUMENTS</code> belongs:</p> <ul style="list-style-type: none"> <li>• string: single line text</li> <li>• text: multiline text</li> <li>• boolean: true value</li> <li>• date: date</li> <li>• enum: enumeration, see below</li> </ul>

	<ul style="list-style-type: none"> <li>• numeric: numeric value</li> <li>• reference: link field to a different DOCUMENTS file</li> <li>• other: None of the above.</li> </ul> <p><b>enum:</b> 0..n string</p> <p>Contains the versions of the enum field if the field is of the "enum" type and the "categories" parameter contains "fields, enum", see Documents.describeFileType. The values for the individual languages are displayed in a single string (1). "de:Jahr; en:Year".</p>
Used in	FileTypeDescription Documents.describeFileType Documents.describeFileType2

## FileInfo

Description	Contains detailed information on the fields of a FileType.
Member	<p><b>idFile:</b> 1 string The unique label of the underlying DOCUMENTS file.</p> <p><b>filetypeId:</b> 1 string ID of file type to which the DOCUMENTS file belongs.</p> <p><b>filetypeName:</b> 1 string Name of file type for this DOCUMENTS file.</p> <p><b>fileTypeLabel:</b> String GUI-label of file type on the Web interface.</p> <p><b>documents:</b> 0..n DocInfo The documents of the DOCUMENTS file including size, name and ID.</p> <p><b>fieldvalues:</b> 0..n FieldData Contains the name/value pairs for the individual file fields.</p>
Used in	DOCUMENTS.GetFilesInfo

## FileStatus

Description	Contains status information on a DOCUMENTS file.
Member	<p><b>fileId:</b> String The DOCUMENTS file ID; for example, it can be used to determine field contents.</p> <p><b>idWorkflowStep:</b> String The current workflow step the DOCUMENTS file is in.</p> <p><b>status:</b> String The status of the DOCUMENTS file in the system. Allowed versions:</p> <ul style="list-style-type: none"> <li>○ DlcFile_Status_Default</li> <li>○ DlcFile_Status_New Newly created DOCUMENTS file</li> <li>○ DlcFile_Status_FollowUp File transferred from resubmission to Inbox</li> <li>○ DlcFile_Status_ToForward User locks DOCUMENTS file and must forward it</li> <li>○ DlcFile_Status_Info User has received DOCUMENTS file for information</li> <li>○ DlcFile_Status_Task User locks the file, must forward it, and the DOCUMENTS file includes a task for the user</li> </ul>

	<ul style="list-style-type: none"> <li>○ <code>DlcFile_Status_CancelWorkflow</code> DOCUMENTS file was canceled by a workflow</li> <li>○ <code>DlcFile_Status_FileBack</code> DOCUMENTS file returned from routing process</li> <li>○ <code>DlcFile_Status_ArchFile_Default</code> File is archive file</li> <li>○ <code>DlcFile_Status_Consultation</code> DOCUMENTS file including enquiry from other user</li> <li>○ <code>DlcFile_Status_Deleted</code> DOCUMENTS file deleted</li> </ul>
Used in	<p><code>Documents.getInbox</code></p> <p><code>Documents.getSentFolder</code></p>

### FileTypeArchiveInfo

Description	Contains identifying data on the archives to which a filetype refers.
Member	<p><b>nameDest:</b> 1 string Name of default target archive for the file type. For EAS, the technical name of the archive server, e.g. "peachitStore1"</p> <p><b>idDest:</b> 1 string Name of the associated, imported archive structure. Empty string in an EAS archive.</p> <p><b>keyDest:</b> 1 string Corresponding key to FileType, as addressed in the archive. For EAS archive file type, followed by "@", followed by technical name of archive server, e.g. "ftRecord@peachitStore1"</p> <p><b>arcSrc:</b> 0..n <b>SourceArchiv</b> List of source archives whose DOCUMENTS files can be reconverted to active processes using this file type.</p>
Used in	<p><code>FileTypeDescription</code></p> <p><code>Documents.describeFileType</code></p> <p><code>Documents.describeFileType2</code></p>

### FileTypeDescription

Description	Contains detailed information on a DOCUMENTS file.
Member	<p><b>id:</b> 0..1 string The ID of the FileType within the system.</p> <p><b>name:</b> 0..1 string The technical name of the FileType.</p> <p><b>archiveinfo:</b> 0..1 <b>FileTypeArchiveInfo</b> Information on the archives to which the file type refers.</p> <p><b>docregisters:</b> 0..n <b>RegisterDescription</b> Information on the document tabs.</p> <p><b>fields:</b> 0..n <b>FieldDescription</b> Description on the individual fields.</p> <p><b>workflowinfo:</b> 0..1 <b>FileTypeWorkflowInfo</b> Information on which workflow is assigned to this file type.</p>
Used in	<code>Documents.describeFileType</code> . The "categories" parameter decides which data structures are populated.

### FileTypeDescription2

Description	Contains detailed information on a DOCUMENTS file. Except for the
-------------	---



	<i>label</i> the data structure is the same as <b>FileTypeDescription</b> .
Member	<b>id:</b> 0..1 string The ID of the FileType within the system. <b>name:</b> 0..1 string The technical name of the FileType. <b>label:</b> : 0..1 String The ergonomic name of the FileType in the current user's language. <b>archiveinfo:</b> 0..1 <b>FileTypeArchiveInfo</b> Information on the archives to which the file type refers. <b>docregisters:</b> 0..n <b>RegisterDescription</b> Information on the document tabs. <b>fields:</b> 0..n <b>FieldDescription</b> Description on the individual fields. <b>workflowinfo:</b> 0..1 <b>FileTypeWorkflowInfo</b> Information on which workflow is assigned to this file type.
Used in	Documents.describeFileType2. The "categories" parameter decides which data structures are populated. This data structure is available since WSDL DOCUMENTS-4.0.1827.

#### FileTypeShortDescr

Description	Contains status information on a DOCUMENTS file.
Member	<b>id:</b> 1 string The ID of the file type within the system. <b>name:</b> String The technical name of the FileType.
Used in	Documents.getFileTypes

#### FileTypeWorkflowInfo

Description	Contains information on the default workflow of a file type
Member	<b>StdForwarding:</b> 1 <b>RegisterDescription</b> Contains a description of the default distribution list. <b>Workflow:</b> 1 <b>WorkflowIdent</b> Description of the current workflow assigned to the file type.
Used in	FileTypeDescription Documents.describeFileType Documents.describeFileType2

#### FilingPlan

Description	Contains the data of a filing plan.
Member	<b>name:</b> 1 String Technical name of the filing plan. <b>label:</b> 1 String Filing plan name on the web interface in the current user's language. <b>description:</b> 1 String
Used in	Documents.getFilingPlans

#### FolderDescription

Description	Describes a level within a default folder
Member	<b>id:</b> 1 string

	<p>Technical name of the folder which can be used, for example, for Documents.BrowseFolder.</p> <p><b>label:</b> 1 string Folder name on the web interface.</p> <p><b>hasSubFolders:</b> 1 Boolean Indicates whether this folder has more subfolders.</p> <p><b>subFolders:</b> 0..n <b>FolderDescription</b> An indeterminate number of folder descriptions of the subfolders allowing recursive navigation through the folder tree.</p>
Used in	Documents.getFolderStructure

## FolderDescription2

Description	Describes a level within a public folder
Member	<p><b>id:</b> 1 String Folder id, this can be used, for example, for Documents.BrowseFolder.</p> <p><b>folderType:</b> 1 String The following folder types are available currently:</p> <ul style="list-style-type: none"> <li>Public</li> <li>PublicDynamicFilter – public with dynamic filter</li> <li>OnlySubFolder – Contains only subfolders</li> </ul> <p><b>label:</b> 1 String Folder name on the web interface.</p> <p><b>labelML:</b> 1 String The entirely multilingual label of the folder.</p> <p><b>name:</b> 1 String Technical name of the folder.</p> <p><b>hasSubFolders:</b> 1 Boolean Indicates whether this folder has more subfolders.</p> <p><b>subFolders:</b> 0..n <b>FolderDescription2</b> A list of folder descriptions of the subfolders allowing recursive navigation through the folder tree.</p>
Used in	Documents.listPublicFolders

## FolderFile

Description	Describes a file within a default folder.
Member	<p><b>id:</b> 1 string File ID within the folder.</p> <p><b>values:</b> 1 StringList The values of the fields for this DOCUMENTS file if requested.</p>
Used in	<p>Documents.getFolderStructure</p> <p>Documents.bowserFolder</p>

## HitData

Description	Contains the data for a hit list created during a search.
Member	<p><b>column:</b> 0..n string The columns in the output report. These may be less than that specified in the <b>Documents.report2</b> method with <b>columns</b> in case a requested field in the file type does not exist or can generally not be read for the user account used.</p> <p><b>reserved:</b> 1 Integer</p>

	Reserved for future functionality.
Used in	HitList Documents.report2

### HitList

Description	Contains the data for a hit list created during a search.
Member	<b>rows:</b> 1 Integer The number of hit list rows; equivalent to number of hits. <b>columns:</b> 1 Integer The column number in the output report. This number may be less than that specified by <i>fields</i> in case a requested field does not exist in the file type or it is not generally legible to the user account used. <b>hit:</b> 0..n <b>HitData.</b> Contains the actual hit values.
Used in	Documents.report2

### MonitorEntry

Description	Contains the data for the monitor entry of a DOCUMENTS file.
Member	<b>status:</b> String File status, as follows: <ul style="list-style-type: none"> <li>• "Angelegt" "Created"</li> <li>• "Wartend" "Waiting"</li> <li>• "Gesperrt" "Locked"</li> <li>• "Weitergeleitet" "Forwarded"</li> <li>• "Durchgeführt" "Processed"</li> <li>• "Informiert" "Informed"</li> <li>• "Beendet" "Finished"</li> <li>• "Zurückgeholt" "Canceled"</li> <li>• "Initiator wartend" "Waiting Initiator"</li> <li>• "Beendet und zurück" "Finished and Back"</li> <li>• "Versendet" "Sent"</li> <li>• "Gelesen" "Read"</li> <li>• "Nicht berücksichtigt" "Not considered"</li> <li>• "Escalation 1" "Escalation 1"</li> <li>• "Escalation 2" "Escalation 2"</li> <li>• "Escalation" "Escalation"</li> <li>• "Fehlgeschlagen" "Failed"</li> <li>• "Angezeigt" "Shown"</li> </ul> <b>executive:</b> 1 string Login of file editor. <b>entryDate:</b> 1 string Date and time of incoming DOCUMENTS file. <b>responseDate:</b> 1 string Date and time of file forwarding. <b>fileOk:</b> 1 Boolean Whether the DOCUMENTS file has been forwarded in "OK" status. <b>task:</b> 1 string The task assigned to the DOCUMENTS file on incoming files for editing. <b>comment:</b> 1 string The comment given to the forwarder.
Used in	Documents.getMonitor

### RegisterDescription

Description	Contains information on identifying a document tab.
Member	<b>name:</b> 1 string Contains the name of a document tab. <b>id:</b> 1 string Document tab ID.
Used in	FileTypeDescription Documents.describeFileType Documents.describeFileType2

### SourceArchiv

Description	Contains information on identifying a source archive.
Member	<b>nameSrc:</b> 1 string Name of source archive <b>idSrc:</b> 1 string Source archive ID <b>keySrc:</b> 1 string Source archive key, as specified in the archive
Used in	FileTypeDescription Documents.describeFileType Documents.describeFileType2

### StringList

Description	Contains an indeterminate number of strings.
Member	<b>string:</b> 0..n strings
Used in	Documents.bowserFolder
Comment	Depending on the programming language and IDE used, string arrays may be encapsulated once again in a string list.

### WorkflowAction

Description	Contains ID and label of an action within a workflow.
Member	<b>id:</b> 1 string The action's ID. <b>label:</b> 1 string The label of the action in the Web interface.
Used in	Documents.listPossibleActions

### WorkflowIdent

Description	Contains version and ID for the current workflow of a file type.
Member	<b>version:</b> 1 string Version name of workflow. <b>id:</b> 1 string Workflow ID.
Used in	FileTypeDescription Documents.describeFileType Documents.describeFileType2

### WorkflowPattern

Description	Contains name and ID of a workflow.
Member	<b>idWorkflowPattern:</b> 1 string

	Workflow ID. <b>nameWorkflowPattern:</b> 1 string Workflow name.
Used in	Documents.getWorkflowPattern

### WorkflowStep

Description	
Member	<p><b>comment:</b> 1 string The comment included in the transition from one workflow step to the next.</p> <p><b>entryDate:</b> 1 string The date and time of incoming DOCUMENTS file ("Received")</p> <p><b>fileOK:</b> 1 string Indicates whether the DOCUMENTS file has been forwarded in "OK" state.</p> <p><b>finishDate:</b> 1 string Date and time of outgoing DOCUMENTS files ("Response")</p> <p><b>groupFlag:</b> 1 Boolean Indicates whether this workflow step has been assigned to a group.</p> <p><b>hasAgent:</b> 1 Boolean Indicates whether this workflow step has been edited by a delegate.</p> <p><b>idWorkflowStep:</b> 1 string Workflow step ID</p> <p><b>locksFile:</b> 1 Boolean Indicates whether this workflow step exclusively locks the DOCUMENTS file for editing.</p> <p><b>redirectionDescription:</b> 1 string Contains brief information on who was the delegate for which employee on making edits.</p> <p><b>status:</b> 1 string The state of the DOCUMENTS file in the workflow step. See <b>MonitorEntry/status</b></p> <p><b>task:</b> 1 string The task that the editor of the workflow step has been assigned.</p> <p><b>user:</b> 1 string Editor's login. This may also be a script or, in automated operation, for example, a "decision".</p> <p><b>wasRedirected:</b> 1 Boolean Indicates whether this workflow step is based on an enquiry.</p>
Used in	Documents.getWorkflowSteps see also Documents.getMonitor

## 3.3 Other functions

### WSDL

Description	Read WSDL from active proxy through an Http request.
Sample VB	Try

	<pre> Dim url AS String url = "/?wsdl"  Dim webRequest As System.Net.HttpWebRequest webRequest = Nothing  Dim webResponse As System.Net.HttpWebResponse webResponse = Nothing  ' doc.Url Proxy location and port webRequest = CType(System.Net.WebRequest.Create(doc.Url + url), System.Net.HttpWebRequest)  webResponse = CType(webRequest.GetResponse(), System.Net.HttpWebResponse)  Dim receiveStream As System.IO.Stream = webResponse.GetResponseStream()  Dim encode As System.Text.Encoding = System.Text.Encoding.GetEncoding(0)  Dim readStream As New System.IO.StreamReader(receiveStream, encode)  Console.WriteLine("Response stream received")  Dim read(256) As [Char] Dim count As Integer = readStream.Read(read, 0, 256)  Console.WriteLine("WSDL ..." + System.Environment.NewLine)  While count &gt; 0     Dim str As New [String](read, 0, count)     Console.WriteLine(str)     count = readStream.Read(read, 0, 256) End While  Console.WriteLine("... WSDL") readStream.Close() webResponse.Close()  Catch ex AS Exception     Console.WriteLine("sampleGetWSDL Message: {ex.Message}") End Try </pre>
Explanation	We will determine the current proxy WSDL via an Http-Get call

## 4. HTTPS support (SSL/TLS)

Since DOCUMENTS 4.0d #1870 the SOAP proxy supports also HTTPS connections as an alternative to HTTP connections from SOAP clients.

The SOAP proxy side HTTPS support is solely restricted to a transport encryption. A client authentication by the SOAP proxy is NOT supported.

This Documentation supposes knowledge of SSL/TLS functionality and of the using of digital certificates (X.509). The configurations of the SOAP proxy and code samples for the applications are described here.

### 4.1 Configurations in the docsoapproxy.ini

By default the HTTPS support is disabled. When an encrypted connection shall be used, the following parameters in the INI-file `\soapproxy\docsoapproxy.ini` have to be configured:

- `SSL=1`  
enables the SSL support.
- `keyfile`  
Absolute path or relative path to the installation path of a certificate/key file. The pem-file contains at least the private RSA key and the certificate of the SOAP proxy. Intermediate certificates from a certificate chain may follow, if necessary.

The private key has to be included in PEM format (base64 encoded) enclosed by

```
-----BEGIN ENCRYPTED PRIVATE KEY-----
```

and

```
-----END ENCRYPTED PRIVATE KEY-----.
```

The individual certificates are also included in PEM format (base64 encoded) enclosed respectively by

```
-----BEGIN CERTIFICATE-----
```

und

```
-----END CERTIFICATE-----.
```

- `keypasswd`  
Password to read the private key in the key file. The password has to be specified in plain text.

If a server authentication should be performed at the client side, then the correspondent trusted CA root certificates are needed.

Depending on the technology used by the client different certificate stores are in use. A client created via .net or PortalScripting under Windows uses the certificate

store of the operation system implicitly. Windows opens the appropriate administration tool for example with the command line call “MMC certmgr.msc”. In contrast a gSOAP client needs an own root certificate file (see 4.3 for more information). The client application itself must here take charge of the preparation and regular update of the root certificate file.

If an attacker will succeed in foisting a fake root certificate on the client, or if the client will trust in a compromised root certificate, the connection would no longer be secure.

## 4.2 Creating a self-signed X.509 certificate

For simple test purpose you can create a self-signed X.509 certificate. You only need OpenSSL installed on the server (e.g. together with EDA -> C:\Program Files\Documents4\eas\http\bin\openssl.exe). With the following command a 2048 bit long RSA-key will be generated and saved as `testkey.pem`. A self-signed X.509 certificate will also be created and stored in `testcert.pem`. The certificate is valid for 365 days.

```
openssl req -x509 -days 365 -newkey rsa:2048 -out
testcert.pem -keyout testkey.pem
```

Under Windows it may be necessary to specify the configuration file (e.g. C:\Program Files\Documents4\eas\http\conf\openssl.cnf) explicitly with `-config <path-to-openssl.cnf>`.

```
openssl req -x509 -days 365 -newkey rsa:2048 -out
testcert.pem -keyout testkey.pem -config <path-to-
openssl.cnf>
```

During the generation some details of the certificate will be requested. You can skip the not needed fields by entering a point ‘.’:

```
writing new private key to 'testkey.pem'
Enter PEM pass phrase:
Verifying - Enter PEM pass phrase:
-----
Country Name (2 letter code) [AU]:DE
State or Province Name (full name) [Some-State]:NRW
Locality Name (eg, city) []:Dortmund
Organization Name (eg, company) [Internet Widgits Pty Ltd]:CompanyName
Organizational Unit Name (eg, section) []:IT
Common Name (e.g. server FQDN or YOUR name) []:localhost
Email Address []:test@domain.de
```



The concatenated contents of `testkey.pem` and `testcert.pem` build the content of the new `server.pem` (keyfile). The specified password above matches the option `keypasswd`. For verifying the server on the client side the file `testcert.pem` will be used.

### 4.3 HTTPS/SSL capable gSOAP client with Visual C++ 2008®

In section 2.7 we have described, how to create a gSOAP client with Visual C++ 2008®. Now we want to transform it into a HTTPS capable client (see [Secure SOAP Clients with HTTPS/SSL](#) from the gSOAP user guide for more information). For this task we assume that OpenSSL is available on the client. For the project the following settings should be performed:

- Add include path to OpenSSL (e.g. `C:\openssl-win32\include`) under Project > Properties > C/C++ > General > Additional Include Directories.
- Enter the option `-DWITH_OPENSSL` under Project > Properties > C/C++ > Command Line > Additional Options.
- Add OpenSSL library path (e.g. `C:\openssl-win32\lib`) under Project > Properties > Linker > General > Additional library Directories.
- Enter `ssleay32.lib` and `libeay32.lib` under Project > Properties > Linker > Input > Additional library Dependencies.

Now we adjust the code as follows.

```
int _tmain(int argc, _TCHAR* argv[])
{
    // Create a gSOAP client object.
    DOCUMENTS doc;

    // Endpoint URL of soapproxy (change as needed)
    doc.endpoint = "https://localhost:11001";

    // Init SSL
    soap_ssl_init();

    // Init gSOAP context
    if( soap_ssl_client_context(doc.soap,
        SOAP_SSL_DEFAULT, //flags for SSL client/server authentication
        settings
        NULL, //keyfile (cert+key): required only when client must
        authenticate to server
        NULL, //password to read the key
        "testcert.pem", //cafile to store trusted root certificates
        NULL, //capath to directory with trusted root certificates
        NULL //if randfile!=NULL: use a file with random data to seed
        randomness
    ) )
```

```

    {
        soap_print_fault(doc.soap, stderr);
        exit(1);
    }

    // Sample for login

    // ...

    return 0;
}

```

Thereby we have used the self-signed certificate `testcert.pem` created in 4.2.

Finally we copy the DLLs `ssleay32.dll` and `libeay32.dll` into the directory, where the application (e.g. `docsoapsample.exe`) is located. It is essential to use correct DLLs, which match the linked import libraries. Otherwise the application will likely crash.

#### 4.4 Further HTTPS/SSL capable clients

As mentioned before, a client created via .net or PortalScripting uses the CA root certificates from the Windows certificate store implicitly. For test purpose we use here the certificate `testcert.pem` from 4.2 and import it using the certificate snap-in (`certmgr.msc`) as follows:

- Click the folder (Trusted Root Certification Authorities) that you want to import the certificate into;
- Select the menu `Action > All Tasks > Import`;
- Click `Next` and then follow the instructions.

##### WCF client using Visual C#

The WCF client created according to section 2.8 using Visual C# cannot be transformed into a HTTPS capable one. It expects “http” as URI-schema. However, you can create a HTTPS capable WCF client with Visual C# in the same way. The only thing we need to do in advance is changing the URI-schema “http” of the location in `DOCUMENTS.wsdl` to “https”:

```
<SOAP:address location="https://localhost:11001"/>
```

Then follow the steps described in the section 2.8.

##### VB client

The VB client from section 2.6 can easily be turned into a HTTPS capable client by changing the SOAP proxy URL from HTTP to HTTPS.

```

Dim doc as New DOCUMENTS

' Die URL unter der der Proxy für diese Anwendung erreichbar
ist.

```

```
doc.Url = "https://localhost:11001"
```

#### PortalScripting client

The JavaScript client described in the PortalScripting documentation can be turned into a HTTPS capable client in the same way as the VB client.

```
// URL for the SOAP Proxy  
var url = "https://localhost:11001/";
```

## 5. Appendix

### 5.1 Archive file key

#### 5.1.1 EE.i:

The *Archive key* is the unique identifier of an archive.

Format:

```
$(#Location)\ArchiveName@technical name of archive server in  
DOCUMENTS
```

Example:

```
$(#STANDARD)\EINRECH@enterprisei
```

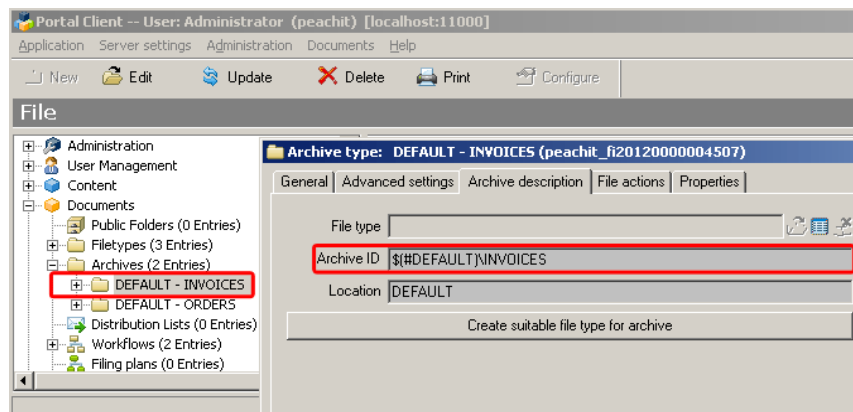


Fig. 8: EE.i archive key including details dialog on imported archive structure

The *archive file key* is a unique identifier of a file within an archive. These keys contain some components twice. The reason is that DOCUMENTS uses the part before the '|' as a pure file identifier, whereas the back part identifies the archive structure.

Format:

```
$(#Location)\ArchiveName,FileId,VersionId|$(#Location)\Archi  
veName@technical name of archive server
```

Example:

```
$(#STANDARD)\EINRECH,00000009,001|$(#STANDARD)\EINRECH@eei1
```

### 5.1.2 EE.x:

The *View key* is the unique identifier of a view, which is associated to at least one schema within an archive:

Format:

```
Unit=UnitName/Instance=InstanceName/View=ViewName@technical name  
of archive server in DOCUMENTS
```

Example:

```
Unit=Default/Instance=Default/View=EASY@enterprisex
```

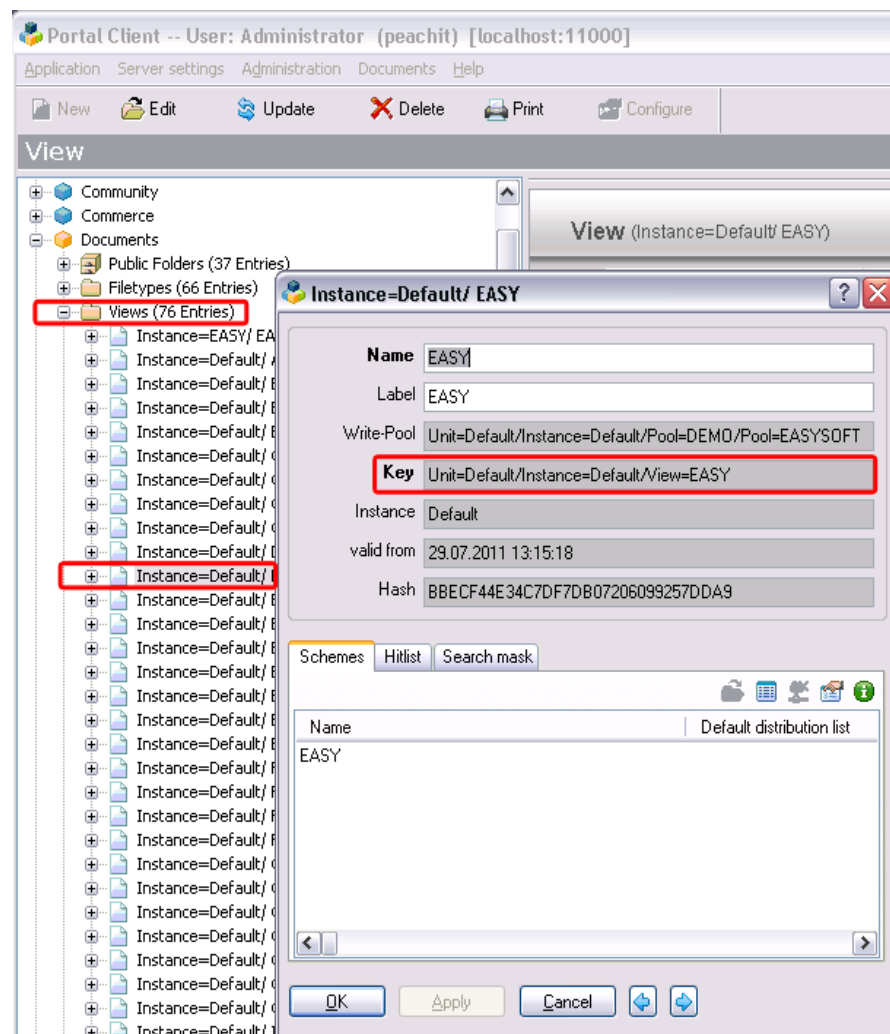


Fig. 9: EE.x view key including details dialog on imported archive structure

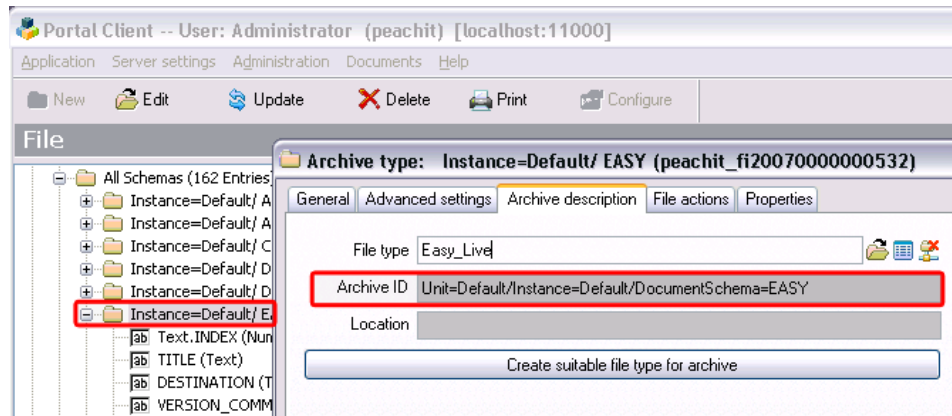
The **Schema key** is the identifier of a schema within an archive.

Format:

```
Unit=UnitName/Instance=InstanceName/DocumentSchema=SchemaName
```

Example:

```
Unit=Default/Instance=Default/DocumentSchema=EASY
```



*Fig. 10: EE.x schema key including details dialog on imported archive structure*

The *archive file key* of an EE.x file is formed as follows:

Format:

```
Unit=UnitName/Instance=InstanceName/Pool=PoolName  
/Document=FileID|Schema-key@technical name of archive server
```

Example:

```
Unit=Default/Instance=Default/Pool=DEMO/Pool=EASY/Document=E  
ASY.45C76F221D3E11DF92AF080027B22D11|Unit=Default/Instance=D  
efault/DocumentSchema=EASY@eex1
```

### 5.1.3 EAS:

An archive key of EAS is composed as follows:

Format:

```
FileType@technical name of archive server in DOCUMENTS
```

Example:

```
ftRecord@peachitStore1
```

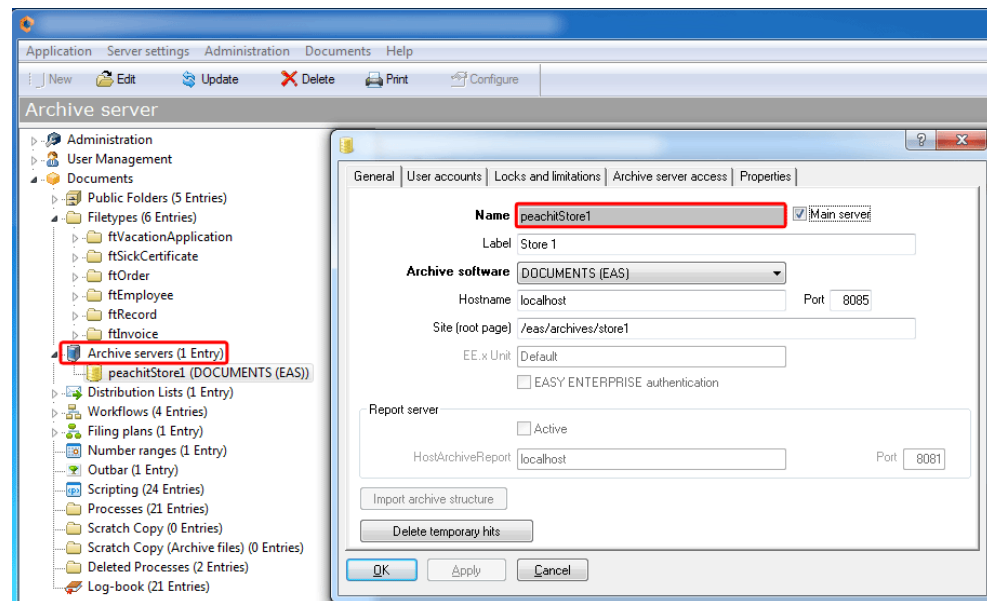


Fig. 11: Archive server EAS

The *archive file key* of an EAS file performs as follows:

Format:

```
Archive file GUID|FileType@technical name of archive server
```

Example:

```
f5b2d475-587e-4020-bf15-7017722db169|ftRecord@peachitStore1
```

## 5.2 Search resource id for an archive

Archive software	Search resource id
EAS	Filetype@Archive server
	z.B. Filetype1@store1
EE.i	\$(#Location)\Archive name@Archive server
	z.B. \$(#STANDARD)\REGTEST@eei1
EE.x	View-Key@Archive server
	z.B. Unit=Default/Instance=Default/View=REGTEST@eex1

The search resource ids (see also Archive file key) are used by Documents.report2 and Documents.report3.

## 5.3 Destination identifier for archive

Archive software	Destination identifier for archive
EAS	Filetype@Archive server
	z.B. Filetype1@store1
EE.i	\$(#Location)\Archive name@Archive server
	z.B. \$(#STANDARD)\REGTEST@eei1
EE.x	View-Key*Schema-Key@Archive server
	z.B. Unit=Default/Instance=Default/View=REGTEST*Unit=Default/Instance=Default/DocumentSchema=REGTEST@eex1

The destination identifiers for archive (see also Archive file key) are used by the following functions:

- Documents.createFile
- Documents.createFile2
- Documents.createFile3
- Documents.editFile
- Documents.editFile2
- Documents.editFile3

## 5.4 Field labels for reading file attributes in reports/search

There are two categories of specific labels that can be used as field names to output file attributes. The labels of the first category are globally applicable. However, for technical reasons later versions may make changing these names necessary.

Label	Meaning
DlcFile_Title	File title
DlcFile_Created	Date created



DlcFile_Owner	File owner
DlcFile_LastModified	Last modified
DlcFile_LastEditor	Last editor
DlcFile_Id	Unique technical file name

The second category of specific labels will never change. If, however, a field of the same name exists for one of these names, then the corresponding field value instead of the file attribute of the same name will be output for that file type.

Label	Meaning
Title	File title
CreatedAt	Date created
FileOwner	File owner
LastModifiedAt	Last modified
lastEditor	Last editor
id	Unique technical file name

## 5.5 Label for personal default folders

Label	Meaning
Favorites	Favorites folder
Inbox	Inbox folder
Sent	Sent
SendingFinished	"Finished Sending" folder
InWork	"In Progress" folder"
FollowUp	Resubmission folder
Deleted	"Deleted" folder"
Tasks	Tasks folder
LastUsed	"Last Used" folder"
InTrouble	"InTrouble" folder"
Used in	Documents.getFolderStructure Documents.bowserFolder

## 5.6 Syntax description for filter expressions

The global notation for individual filter conditions for the Documents.report and Documents.report2 operations is:

Field name Comparison operator Value
--------------------------------------

The following *comparison operators* are allowed: = (equal to), < (less than), > (greater than ), <= (less than or equal to ), >= (greater than or equal to ), ~ (contains) and <> or != (not equal to).

The comparison operator should neither be preceded nor followed by spaces. If a value contains spaces, it must be enclosed in quotes.

Except for range searches, the advanced DOCUMENTS search always uses the ~ (contains) operator. Only when using this operator the search term will be split into individual words. All other operators will process the search term as a whole.

Example:

- The `User='Eva Frisch'` condition is met only by files, where the `User` field contains exactly the text `Eva Frisch` without any other characters preceding it or following it.
- By contrast, the `User~'Eva Frisch'` condition will be interpreted either as `'Eva&Frisch'` (logical and) or as `'Eva|Frisch'` (logical or). It depends on the server configuration (see `LogicalAndBetweenSearchPhrases` in the „documents.ini“). With search method 0 the according result is a list of files containing both character strings at arbitrary positions in the field (and-setting) or just one of them (or-setting). Here the result is the same as with inputting both words (without quotes) in the Web form for advanced DOCUMENTS search.

The filter condition should be as follows when searching for files where Eva Frisch or Stefan Gross have been entered as the users:

```
User~'"Eva Frisch"|"Stefan Gross"'
```

Multiple filter conditions can currently only be linked with `AND` (logical "and") operators. `OR` operators are not yet available at this level.

Advanced file attributes can be used in the filter expressions as follows:

Label	Meaning
"Search_Fulltext"	Full text search
"Search_DateFrom"	From
"Search_DateUntil"	To
"DlcFile_Title"	File title
"Search_Owner"	File owner
"Search_LastEditor"	Last editor
"Search_ModDateFrom"	From (Last modified)
"Search_ModDateUntil"	To (Last modified)

For further details about filter expressions (such as archive related restrictions) please read also the chapter „Using filter expressions with `FileResultSets`“ in the *PortalScripting* reference manual.

## 5.7 Schema for `getFilingPlanXML` output

```
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <xsd:element name="FilingPlan" type="FilingPlanEntryList"/>
  <xsd:complexType name="FilingPlanEntryList">
    <xsd:sequence>
      <xsd:element name="FilingPlanEntry"
type="FilingPlanEntryType"/>
    </xsd:sequence>
    <xsd:attribute name="name" type="xsd:string"/>
    <xsd:attribute name="label" type="xsd:string"/>
  </xsd:complexType>
</xsd:schema>
```

```

</xsd:complexType>
<xsd:complexType name="FilingPlanEntryType">
  <xsd:sequence>
    <xsd:element name="Index" type="xsd:string"/>
    <xsd:element name="Label" type="xsd:string"/>
    <xsd:element name="Custom1" type="xsd:string"/>
    <xsd:element name="Custom2" type="xsd:string"/>
    <xsd:element name="Custom3" type="xsd:string"/>
    <xsd:element name="FilingPlanEntry"
type="FilingPlanEntryType"/>
  </xsd:sequence>
  <xsd:attribute name="id" type="xsd:string"/>
</xsd:complexType>
</xsd:schema>

```

## 5.8 Client timeouts

SOAP clients often implement a maximum waiting time for the SOAP proxy's response. After that period of time the client aborts the connection. The occurrence of such a client timeout leads to a session loss. Speaking more precisely, the logout is no longer possible afterwards, because the client lacks a valid session Id. The sequence diagram from Fig. 12 illustrates the occurrence of a client timeout after 60s.

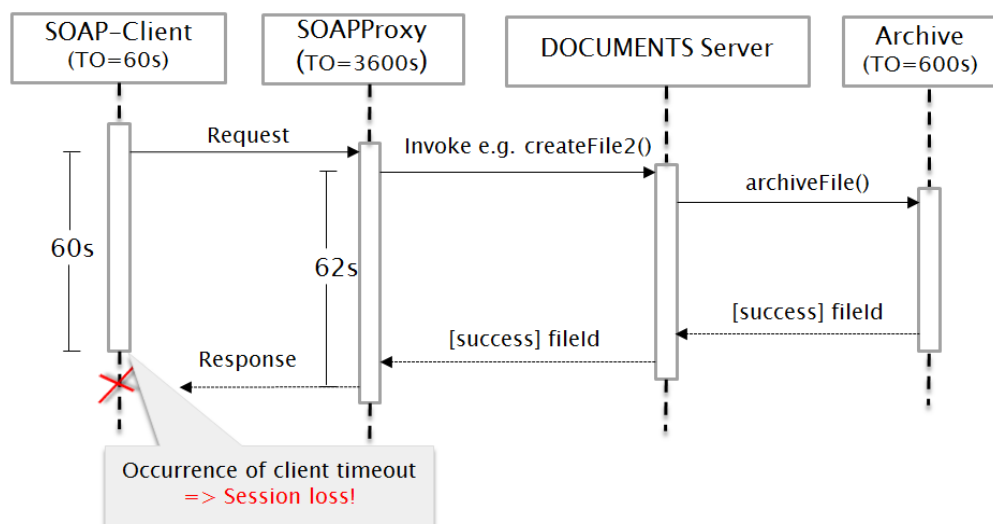


Fig. 12: Sequence diagram of a client timeout

The session, which has been lost from the client's point of view, it is still alive within the proxy and the server. If this occurs frequently, all licenses will be

allocated after a while (limit: 3 SOAP sessions per user), and further logins will be rejected. Only when the proxy timeout period has also elapsed (see Fig. 3), a lost session will be freed again. To avoid this scenario the period of client timeouts should be amply dimensioned, if they are not dispensable.

## 6. Index

!=	89
~	89
<	89
<=	89
<>	89
=	89
>	89
>=	89
65001	15
Abmeldung	58
actionId	64
addDocs	29, 30, 31, 38, 40, 42
allAttributes	46, 48
allFields	46, 48
AND	90
Anmeldung	57
Archiv	68
Key	84
archiveFiles	26
archiveinfo	35, 37, 72, 73
archives	58, 59, 61
ArchiveStatus	68
Archivmappen	
deleteFiles	34
getDocument	45
getFileInfo	46
Key	
EAS	87
EE.i	84
EE.x	86
Archivmappen-Key	40, 42
editFile	38
arcSrc	72
asUser	66
Aufgabenordner	89
Ausprägung	70
Autotext	44
autoTextNames	44
autoTextValues	44
backWhenFinished	64
base64Binary	69
baseFolder	50
Bearbeiter	89, 90
Bearbeitung	89
Bearbeitungskommentar	64
Bedingung	90
Benutzeraktion	64
Benutzerrechte	49
Benutzers	67
Bis	90
boolean	70
browseFolder	27
cancelWorkflow	29
categories	35, 37
chcp	15
code	57, 66
Codepage	15
column	74
columns	58, 59, 75
<b>comment</b>	
DocInfo	68
followUp	43
MonitorEntry	75
triggerAction	64
WorkflowStep	77
count	27
CreatedAt	89
createFile	29, 69, 70
createFile2	30
data	45, 69
date	70
Dateiname	45
Datentyp	10
deleted	68
Deleted	89
deleteDocuments	32
deleteFiles	34
deleteOnSuccess	26
DeleteStatus	68
describeFileType2	37
describeFileTypes	34, 37, 70, 71, 72, 73, 76
description	35, 38, 73
direktEAS	30, 31
DlcErr	28
DlcFile_Created	88
DlcFile_Id	89
DlcFile_LastEditor	89
DlcFile_LastModified	89
DlcFile_Owner	89
DlcFile_Title	88, 90
docId	45
docIds	32
DocInfo	68
docregisters	35, 37, 72, 73
documents	47, 71
DOCUMENTS.vb	10
DocumentsHost	7
DocumentsPort	7
DocUploadData	69
DocUploadData2	69
Dokumente	46, 48
Download	45

hinzufügen .....	29, 30, 31, 38, 40, 42
löschen .....	32
Dokumentenregisters .....	69, 76
Download .....	45
Ebene .....	73
Eigentümer .....	89
Eingangskorb .....	43
Eingangsordner .....	89
Encoding .....	8
enthält .....	89
entryDate .....	75, 77
enum .....	35, 37, 70
Ergebniszeilen .....	60, 61
errorMsg .....	63
Erstelldatum .....	88, 89
executive .....	75
failOnAllErrors .....	48
Favoritenordner .....	89
Favourites .....	89
Fehlerstring .....	68
Fehlerstrings .....	28
Feldbezeichner .....	88
Feldname .....	
Sortierung .....	58
Feldwerte .....	46, 48
FieldData .....	70
FieldDescription .....	70
fieldlabel .....	35, 37, 70
fieldNames .....	27
fields .....	29, 30, 31, 35, 37, 38, 40, 42, 72, 73
fieldvalues .....	47, 71
fileId .....	40, 42, 52, 65
ArchivStatus .....	68
cancelWorkflow .....	29
createFile .....	29, 31, 32
deleteDocument .....	32
editFile .....	38
FileInfo .....	46, 48
FileStatus .....	71
followUp .....	43
getAutotext .....	44
getDokument .....	45
getMonitor .....	51
getTasks .....	53
getWorkflowSteps .....	55
listPossibleActions .....	56
runScript .....	63
sendFile .....	64
triggerAction .....	64
fileIds .....	
archivieren .....	26
deleteFiles .....	34
fileinfo .....	48
FileInfo .....	71
fileId .....	71
fileOk .....	75
fileOK .....	77
FileOwner .....	89
files .....	27
FileStatus .....	71
fileStatuses .....	51, 53
filetype .....	40, 42
createFile .....	29, 30, 31
editFile .....	38
FileTypeArchiveInfo .....	72
FileTypeDescription .....	72
FileTypeDescription2 .....	72
filetypeId .....	47, 71
fileTypeLabel .....	47, 71
fileName .....	71
filetypes .....	58, 59
FileTypeShortDescr .....	73
FileTypeWorkflowInfo .....	73
fileTypeName .....	47
filter .....	58, 59
Filterausdrücke .....	58, 59, 61, 89
Filterbedingung .....	90
Finally .....	57
finishDate .....	77
FolderDescription .....	73
FolderDescription2 .....	74
FolderFile .....	74
folderId .....	27
folderName .....	27
folders .....	56
folderType .....	27, 50
followUp .....	43
FollowUp .....	89
Funktionsumfang .....	4
Gelöscht .....	89
getAutoText .....	44
getDocument .....	45
getFileId .....	45
getFileInfo .....	46
getFilesInfo .....	48, 71
getFileTypes .....	49, 73
getFilingPlans .....	49
getFilingPlanXML .....	50
getFolderStructure .....	50
getInbox .....	51, 72
getMonitor .....	51
getProperty .....	52
getSentFolder .....	53, 72
getTasks .....	53
getWorkflowPattern .....	55
getWorkflowSteps .....	55
gleich .....	89
größer .....	89
größer oder gleich .....	89
groupFlag .....	77

hasAgent .....	77	label .....	73, 74, 76
hasSubFolders .....	73, 74	FileTypeDescription2 .....	73
HasValue .....	44	LastEditor .....	89
headline .....	27	LastModifiedAt .....	89
hit .....	75	LastUsed .....	89
HitData .....	74	Leerzeichen .....	89
Hit-Id .....	59	listPossibleActions .....	56
hitlist .....	60, 61	listPublicFolders .....	56
HitList .....	75	locale .....	57, 66
hitlistname .....	61	locksFile .....	77
Hochkommata .....	89	login .....	57, 66
HTTP GET .....	77	logout .....	58
id .....	89	Löschvorgangs .....	32, 34
DeleteStatus .....	68	Mappe	
describeFileTypes .....	35, 37	identifizieren .....	39, 41
DocInfo .....	68	löschen .....	34
FieldDescription .....	70	Mappenattributen .....	88
FolderDescription .....	73	Mappenbesitzer .....	90
FolderDescription2 .....	74	Mappenstatus .....	75, 77
FolderFile .....	74	Mappentitel .....	88, 89, 90
Mappentyp .....	72, 73	Mappentyp	
RegisterDSdescription .....	76	Detailinformation .....	34
WorkflowAction .....	76	Mappentypen .....	49
WorkflowIdent .....	76	messages .....	68
idDest .....	72	mime .....	45
idFolder .....	53	Mimetype .....	45
getInbox .....	51	monitorEntries .....	51
idSrc .....	76	MonitorEntry .....	75
idWorkflowPattern .....	65, 76	name .....	69, 73, 74
idWorkflowStep .....	71, 77	describeFileTypes .....	35, 37
ignoreRights .....	49	DocInfo .....	68
In Arbeit .....	89	DocUploadData .....	69
Inbox .....	51, 89	FieldData .....	70
initialValue .....	35, 37	FieldDescription .....	70
InTrouble .....	89	FileTypeDescription .....	72, 73
InWork .....	89	FileTypeShortDescr .....	73
isUTC .....	43	getDocument .....	45
Key		RegisterDescription .....	76
Archivmappen		runScript .....	63
EAS .....	87	userInfo .....	67
EE.i .....	84	nameDest .....	72
EE.x .....	86	Namen .....	67
Schema .....	86	nameSrc .....	76
View .....	85	nameWorkflowPattern .....	76
keyDest .....	72	newId	
keyfield .....	38, 40, 42	editFile .....	39
keySrc .....	76	editFile2 .....	40, 42
keyvalue .....	38, 40, 42	nextIndex .....	27
kleiner .....	89	Nullable .....	43
kleiner oder gleich .....	89	numeric .....	70
Kommunikation .....	12	Oder .....	90
Konkurrierende Anmeldungen .....	57	Öffentliche Ordner .....	74
Konsolenanwendung .....	14	Ordner	
Konsolenschriftart .....	16	Mappen auflisten .....	27
Konvertierung .....	69	Ordnerstruktur .....	50

other .....	70	sort .....	58, 59
paramList .....	63	sortOption .....	56
passwd .....	57, 66	SourceArchiv .....	76
Portalscript .....	63	Spezialbezeichnen .....	88
PortalServerEncoding .....	8	Standardordner .....	28
preview .....	27	Standardordners .....	50
previousIndex .....	27	startIndex .....	27
principal .....	57, 66	Startobjekt .....	14
Projektmappenexplorer .....	14	startWorkflow .....	65
propertyNames .....	52	status .....	32, 71, 75, 77
propertyValues .....	52	statusList .....	26, 34
Proxy .....	5, 57	StdForwarding .....	73
Debugging .....	8	string .....	70, 76
Kommandozeilenaufruf .....	5	StringList .....	76
Konfigurationsdatei .....	6	subFolders .....	73, 74
ProxyPort .....	6	Suchbegriff .....	89
ProxyTimeout .....	6	Suche .....	58
receivers .....	64	Syntaxbeschreibung .....	58, 59, 61, 89
redirectionDescription .....	77	System.Web.Services .....	13
reference .....	70	System.Xml .....	13
register .....	69	table .....	59
RegisterDescription .....	76	task .....	64, 75, 77
replace .....	69	tasks .....	53
report .....	58	Tasks .....	89
report2 .....	59	testSession .....	66
report3 .....	61	text .....	70
Request .....	9	timeAbsolute .....	43
reserved .....	74	timeAbsoluteSpecified .....	43
Response .....	9	Timeout .....	57
responseDate .....	75	Title .....	89
returnStatus .....	63	Trefferliste .....	60, 61
returnValue .....	63	triggerAction .....	64
rows .....	75	trustedLogin .....	66
runScript .....	63	Typbezeichner .....	50
Schema		type .....	70
key .....	86	typeOption .....	56
Search_DateFrom .....	90	Überschreiben .....	39, 41
Search_DateUntil .....	90	ungleich .....	89
Search_Fulltext .....	90	Unterordner .....	28
Search_LastEditor .....	90	Url .....	5, 11
Search_ModDateFrom .....	90	Ursprungarchives .....	76
Search_ModDateUntil .....	90	Ursprungsarchiv .....	68, 72
Search_Owner .....	90	user .....	57, 66, 77
sendFileAdHoc .....	64	userInfo .....	67
SendingFinished .....	89	utf-8 .....	15
sendMode .....	64	valid .....	66
SequenceSessionId .....	7	value .....	70
session .....	57, 67	values .....	74
Session .....	12, 66, 67	Vergleichsoperatoren .....	89
SessionId .....	7	Verknüpfungen .....	90
size .....	45, 68	Versendung .....	89
Skript .....	63	Versendungshistorie .....	64
soaproxy .....	5	version .....	76
SOAP-Schicht .....	11	versioning .....	69
SOAP-Session .....	12, 66	View	



Key .....	85	starten .....	65
Visual Basic .....	10	WorkflowAction .....	76
Visual Studio 2005® .....	10	workflowActions.....	56
Volltextsuche .....	90	WorkflowIdent .....	76
Von .....	90	workflowinfo .....	35, 37, 72, 73
wasRedirected .....	77	WorkflowPattern.....	55, 76
Web Services Enhancements .....	13	Workflowschritt .....	56, 71
Webservice .....	5	WorkflowStep .....	77
Wiedervorlage		WorkflowSteps.....	55
Datum setzen .....	43	WSDL.....	9, 10, 77
Wiedervorlageordner .....	43, 89	wsdl.exe.....	10
wishedFieldNames .....	46, 48	Zielarchivs.....	72
Workflow .....	73, 76	Zuletzt benutzt .....	89
beenden .....	29		

## 7. Table of Figures

Fig. 1: Command line call docsoaproxy --h .....	6
Fig. 2: docsoaproxy.ini as configuration file .....	6
Fig. 3: Sequence diagram of a SOAP proxy timeout .....	7
Fig. 4: Changing the SessionIds as part of login/logout .....	13
Fig. 5: Selecting the startup object for the project .....	14
Fig. 6: Setting the Unicode utf-8 codepage to 65001 .....	16
Fig. 7: Setting the console font Lucida Console .....	16
Fig. 8: EE.i archive key including details dialog on imported archive structure .....	84
Fig. 9: EE.x view key including details dialog on imported archive structure .....	85
Fig. 10: EE.x schema key including details dialog on imported archive structure ..	86
Fig. 11: Archive server EAS .....	87
Fig. 12: Sequence diagram of a client timeout .....	91