Manual for developers

officeatwork ClientSuite API

Creation of Manual
24. September 2020

officeatwork AG has prepared this manual with the greatest possible care so as to ensure that the information contained herein is easy to understand, accurate and reliable. Nevertheless, officeatwork AG is in no way liable for any issues which have any connection with this manual, including – and without restriction – its standard quality and availability for special purposes. From time to time, officeatwork AG will revise the software described in this manual and reserves the right to do so without prior advice to the customer. Under no circumstances is officeatwork AG liable for indirect, special or incidental damages resulting from the purchase or use of this manual or the information contained herein. This guarantee exclusion has no impact on the statutory rights of the user.

Copyright© 1992–2020 officeatwork AG, Switzerland.
All rights reserved.

officeatwork® is a registered trademark of officeatwork AG.

Microsoft® Word, Microsoft® Office, Windows®, Windows 95™, Windows 98™, Windows NT®, Windows XP®, Windows Vista, Windows 7, Windows 8, Windows 10 and MS-DOS™ are trademarks of the Microsoft Corporation.

Other names of companies, products or services may be trademarks or registered trademarks of the respective owners.

Inhaltsverzeichnis

Über diesen Guide

Für wen ist der Guide bestimmt

This book has been written for software developers that want to implement an interface to officeatwork.

Was deckt dieser Guide ab

This manual illustrates the process of integrating an officeatwork interface in your application. It explains all parameters available to the developer. It also provides a best praxis architecture on how to implement the officeatwork interface.

Das sollten Sie bereits wissen

You should be familiar with the general use of computers, especially with the XML notation. Programming knowledge is of advantage.

Typographic conventions

Before reading this guide, you should be familiar with the typographic conventions used.

The following graphic descriptions highlight sections of text with particular significance.

Formatting Convention	Type of Information
Triangle ➢	Step-by-step procedure. You can follow these instructions to perform a specific task.
Bold Typeface	Objects needed for selection, such as menus, buttons, items in a list or table headers.
CAPITAL LETTERS	Key legends on the keyboard. For example `SHIFT`, `CTRL` or `ALT`.
KEY+KEY	Key combinations which must be pressed at the same time are marked with +. Examples: `CTRL+P` or `ALT+F4`.

CHAPTER 1

Introduction

There are many reasons why business applications want to integrate with Microsoft Office. Here are a few reasons:

- Re-use of existing Templates
- Re-use of existing Corporate Design
- Re-use of user skills for editing documents

In order to better understand the challenges you face when integrating Microsoft Office into business applications, we will analyse a few of the most common concepts.

After that we will have a look at the officeatwork approach of bringing together your business application with Microsoft Office.

Microsoft Office integration concepts

Microsoft Office and business applications do not always concur. Basically all applications need specific and specially created templates, which in turn generate many different copied templates. Additionally, it is seldom the case that Microsoft Office data can be accessed from business applications such as ERP, CRM, DMS, etc.

These discrepancies mean that the necessary information needs to be recorded again and again. That is an absolute waste of time and also creates opportunities for errors.

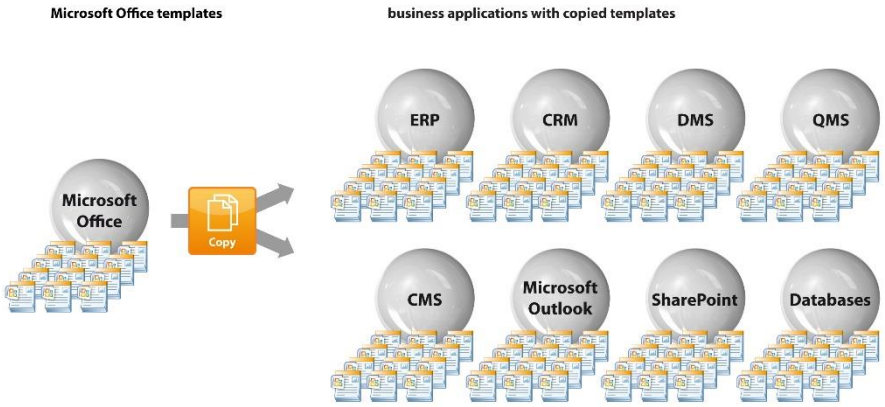

Figure 1: Typical Microsoft Office integration architecture

Integration via Mail-Merge

Typically a static Office template like for instance a «Letter.dot» file is imported into the business application. The template is then modified to include mail-merge fields as placeholders for the business application data.

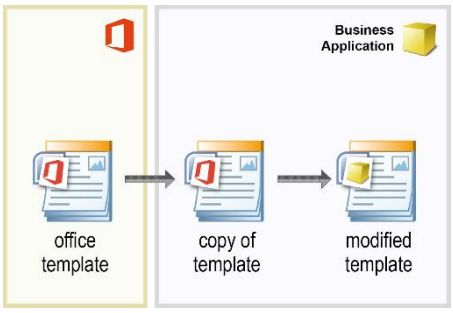

Figure 2: creating a business application specific Office template

The business application directly manipulates that document by controlling the mail-merge function of the Office application, by using VBA (Visual Basic for Applications or any other supported programming language). In this process it writes the business data to a mail-merge compatible file and then opens the template. This is when the user returns to finish the document using the mail-merge functionality.

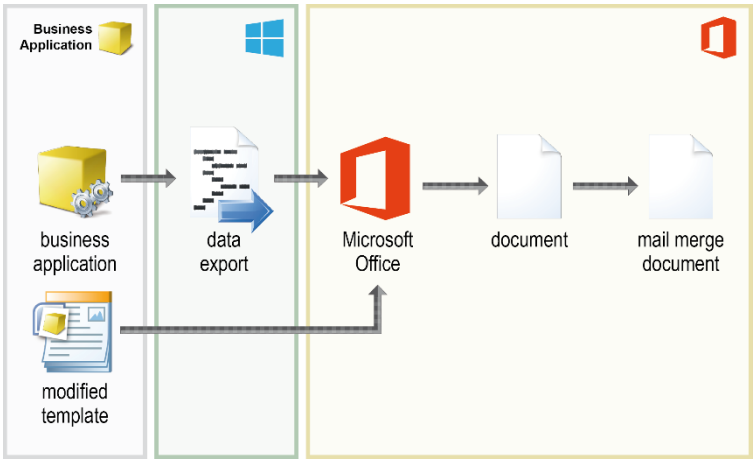

Figure 3: business application Office integration concept via mail-merge function

Pros:

- Using existing functions reduces effort of integration.

Cons:

- Duplication of already existing templates had to additional adjustments in case of design or data changes (e.g. telephone)
- Direct dependency on the Office application version and its offered functionality
- In-depth knowledge about the Office application required
- Testing for each new Officeversion is necessary
- Intensive maintenance
- The whole integration cycle needs to be done for each business application separately

Integration via Bookmarks, DDE/OLE and Co.

Typically a static Office template like for instance a «Letter.dot» file is imported into the business application. The template then gets modified to include bookmarks and other placeholders for the business application data.

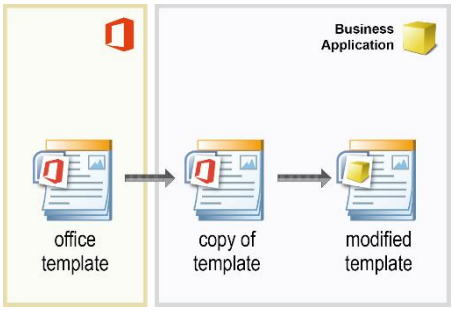

Figure 4: creating a business application specific Office template

The business application then directly manipulates that document by using VBA (Visual Basic for Applications or any other supported programming language). In this process it first generates a new document from the template and then writes the business data directly into it. It depends on the depth of the integration whether the document is presented to the user or processed for output directly by the application.

DDE (Dynamic Data Exchange) is a technology for communication between multiple applications under Microsoft Windows. A common use of DDE was for custom-developed applications to control off-the-shelf software. For example, a custom in-house application might use DDE to open a Microsoft Excel spreadsheet and fill it with data, by opening a DDE conversation with Excel and sending it DDE commands. Today, however, one could also use the Excel object model with OLE Automation (part of COM).

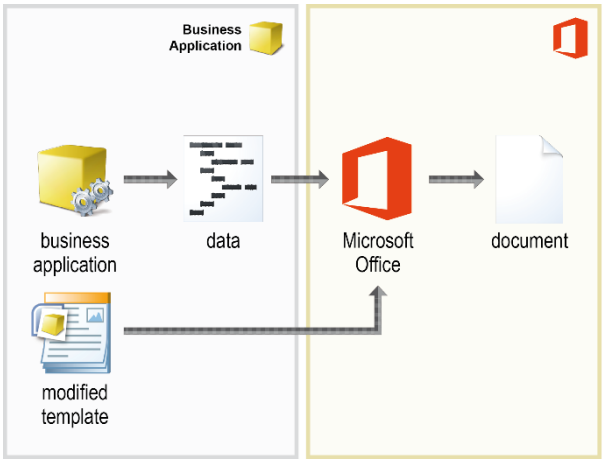

Figure 5: business application Office integration concept via bookmarks, DDE/OLE and Co.

Pros:

- Highly flexible as the full object-model of the Office applications his available to manipulate.
- Standardised interface between many different applications with the potential of reusing the knowledge gained in such an integration. (DDE/OLE)

Cons:

- Direct dependency on the Office application version and its offered functionality
- In-depth knowledge about the Office application required
- Testing for each new Officeversion is necessary
- Duplication of already existing templates
- Intensive maintenance
- The whole integration cycle needs to be done for each business application separately
- Enhancement in functionality often requires re-programming of the interface
- Limited to reduced function-set offered by server applications (DDE/OLE)
- All involved applications need to be running for this integration concept to work. (DDE/OLE)

officeatwork integration concept

officeatwork is a flexible link between Microsoft Office and your business applications. Personal and enterprise information can be automated and directly used in your Office documents. Your Office templates can be directly linked to your business applications like ERP, CRM, QMS, SharePoint, DMS, etc by using the officeatwork XML API interface. It is no longer necessary to duplicate templates.

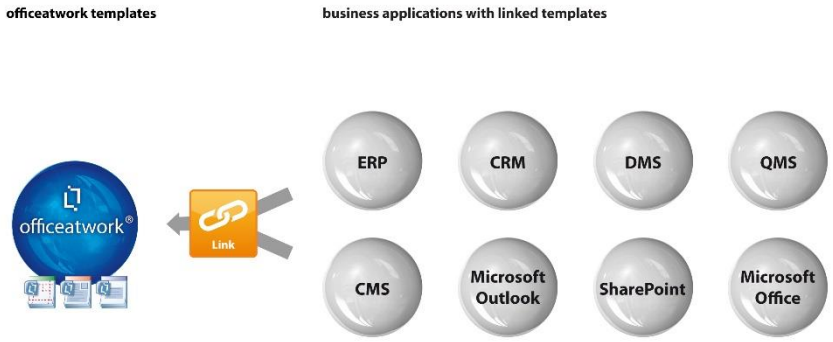

Figure 6: officeatwork integration concept

The main benefit of the officeatwork integration architecture is the fact that no longer copies of templates need to be imported into your business application. Instead your business application can link to existing Office templates using standard XML language. In this process the business application compiles its requirements and data into an XML string and passes that onto officeatwork. officeatwork will then process the XML automatically. Your business application does not need to understand how to create a document in Microsoft Office.

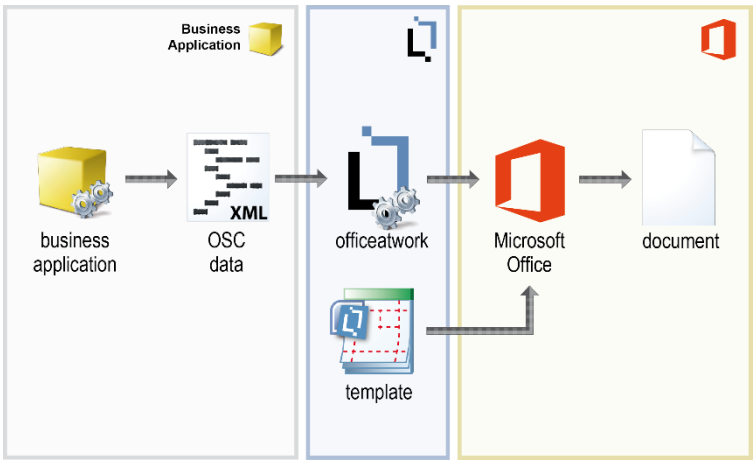

Figure 7: officeatwork business application integration concept

Pros:

- No duplication of already existing templates which therefore reduces maintenance when changing design, Logos, data (e.g. telephone)
- No direct dependency on the Office application version and its offered functionality.

- No in-depth knowledge about the Office application required.
- No testing for each new Officeversion is necessary.
- Maintenance friendly – no reprogramming for new templates or data attributes necessary.
- Business application can share same templates, no separate integration necessary.
- No re-programming of the interface to enhance functionality.

Cons:

- Limited to functionality offered by officeatwork.

CHAPTER 2

officeatwork Integration Architecture for Business Applications

Overview

officeatwork offers two directions of integrating with your business application. The first is from officeatwork to your business application.

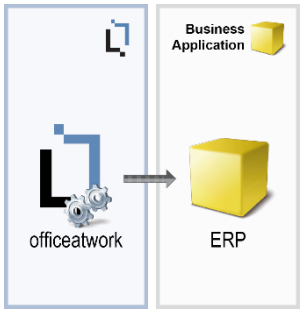

Figure 8: officeatwork interacting with a business application

Here the starting point is officeatwork. From within Microsoft Office officeatwork fetches data from your business application and retrieves them to your Office document. This option is mostly used to fetch address-information from for example your ERP or CRM system as well as user information from for example your Active Directory.

The other direction is from your business application to officeatwork.

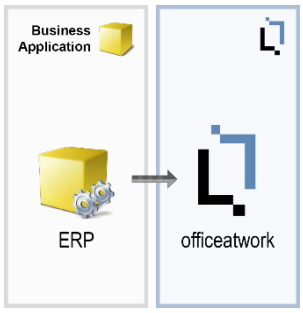

Figure 9: Business application interacting with officeatwork

Here the starting point is your business application. Your business application can be extended so that it can create documents in Microsoft Office using officeatwork functionality. It does this by sending standardised XML formatted instructions to officeatwork. A common example for this option is for instance the creation of a quote based on the information held within your ERP system.

This second option is where the officeatwork API comes into action. It is designed to enable your business applications to communicate in a standardised and flexible way with officeatwork via XML. XML is a widely accepted technology and recommended to be used to communicate between systems.

This book only covers the API integration option. All variations of the first integration option are explained in a separate manual.

Interaction concepts

officeatwork offers different ways for you to create documents via the officeatwork XML API. The parameters of the various ways are the same, just the way officeatwork is involved differs. The three available ways are:

- Passing the parameters to the officeatwork EDC Server using a REST Web-Service.
- Saving the parameters to a file and then sending the OS an Open command to execute that file on a computer having the officeatwork Client Suite installed.
- Passing the parameters to a method in an officeatwork ActiveX component installed with the officeatwork Client Suite.

officeatwork EDC Server

officeatwork XML API can be triggered by passing the parameters to the officeatwork EDC Server using a REST Web-Service. The technical requirements and the Web-Service interface are documented in a separate maual.

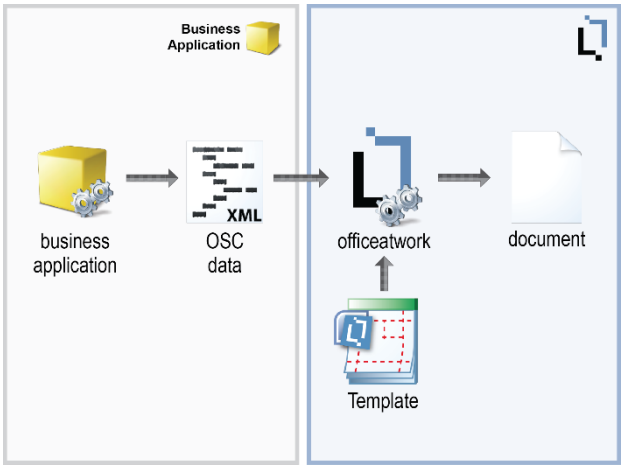

Figure 10: business application Office integration concept via officeatwork EDC Server

Open File with officeatwork Client Suite

officeatwork XML API can be triggered by opening a file with the extension *.OSC (officeatwork shortcut file) on a computer having the officeatwork Client Suite installed. The content of that file must be in XML format and must follow the XML schemes of the officeatwork XML API. The following sample shows a simple example of such an *.OSC file.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<Parameters>
   <CreateDocument>
      <TemplateID>letter</TemplateID>
   </CreateDocument>
</Parameters>
```

The structure of the XML API will be covered in a later chapter. At this point it is important to know that a simple XML formatted file will allow you to interact with officeatwork. Simply double-click your OSC file and officeatwork will open the file and execute the XML formatted instructions.

Please note that when you are working with a web server that serves html pages containing links to OSC files, you must add a MIME type for the OSC files on that web server. In addition to this setting you can replace the first line (encoding line) in the OSC file with the corresponding officeatwork document function. Otherwise clicking on links pointing to OSC files will open the OSC file in your web browser instead of executing the officeatwork XML API. So make sure your web servers have a MIME type **application/osc** for the extension **.osc** defined.

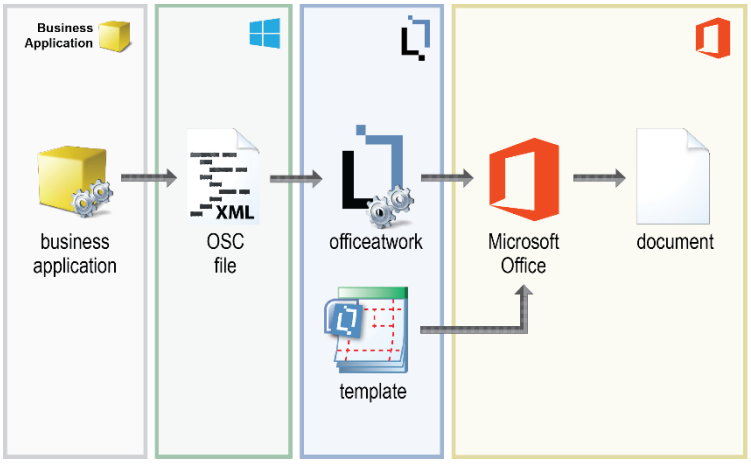

Figure 11: business application Office integration concept via officeatwork shortcut file

Calling Method in officeatwork Client Suite

officeatwork can also be triggered by calling specific officeatwork methods. All available methods will be discussed in a later chapter.

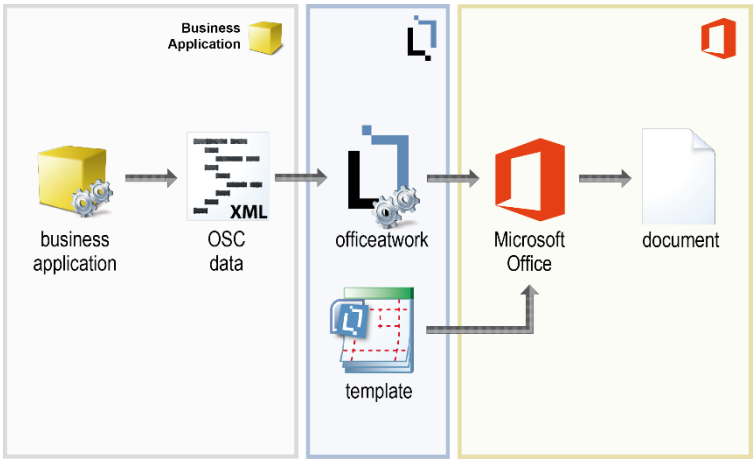

Figure 12: business application Office integration concept via officeatwork specific method

The code sample shows how to call an officeatwork method within a VBA environment:

```
Sub CreateLetter ()

    Dim lOawAPI As oawAPI.API
    Dim lParam As String

    lParam = "<?xml version="1.0" encoding="ISO-8859-1"?>"
    lParam = lParam & "<Parameters>"
    lParam = lParam & "    <CreateDocument>"
    lParam = lParam & "        <TemplateID>Letter</TemplateID>"
    lParam = lParam & "    </CreateDocument>"
    lParam = lParam & "</Parameters>"

    Set lOawAPI = New oawAPI.API

    If (lOawAPI.ExecuteXML(lParam) = -1) Then
       MsgBox "successfully created document"
    Else
       MsgBox "the document could not be created"
    End If

    Set lOawAPI = Nothing

End Sub
```

Basics

General

The XML parameter is, as the name already indicates, a parameter written in XML format. Therefore, all rules and regulations on how to present information in XML format apply.

Just as a reminder, we have listed a few characters that are used to structure the information in XML format and therefore are not allowed to be used elsewhere. If you want to use one of these characters for any purpose other than structuring your XML (for representing your business data for example), you must replace those characters with the equivalent replacement as listed below:

Reserved Character	equivalent replacement
&	&amp;
'	&apos;
>	&gt;
<	&lt;
"	&quot;

Encoding

If you plan to include special characters like **ä**, **é**, etc. within your XML parameter, you must include an encoding tag at the beginning of your XML file. This will make sure your special characters are correctly interpreted.

Sample encoding tag

```
<?xml version="1.0" encoding="ISO-8859-1"?>
```

XML Parameter

The XML parameter consists of easy-to-follow instructions that are processed when passed to officeatwork. The following simple XML file is a sample that, when executed, will create a letter and present the finished document to the user.

```
<Parameters>
    <CreateDocument>
        <TemplateID>Letter</TemplateID>
        <ShowDocumentWizard>0</ShowDocumentWizard>
    </CreateDocument>
</Parameters>
```

Parameters tag

All instructions must be enclosed in a `<Parameters>` tag.

Syntax

```
<Parameters>
</Parameters>
```

Other XML tags on the root level of the XML file will be ignored by officeatwork. The «CreateDocument» tag within the «Parameters» tag will instruct officeatwork to create a new document. The tags within the «CreateDocument» tag describe how to create the document. The «TemplateID» tag lets officeatwork know what template to use for the creation of the new document. In this case it will be a document based on the template with the filename «letter». The «ShowDocumentWizard» tag is set to «0» so the document wizard will be completed without any user interaction necessary. All default values in the wizard will apply.

Recommended integration architecture

Based on many different architectures, we observed that we clearly favour one specific architecture for many reasons. This architecture uses a mixture of the two available interaction concepts.

Overview

The business application uses an officeatwork shortcut template file with the file extension *.OSCT file as a base to generate a new OSC file. During this process it replaces placeholders within the OSCT representing business application values with the proper values. The OSCT may also contain specific business application instructions like loops or counters. The resulting OSC file from the processed OSCT is then processed by officeatwork.

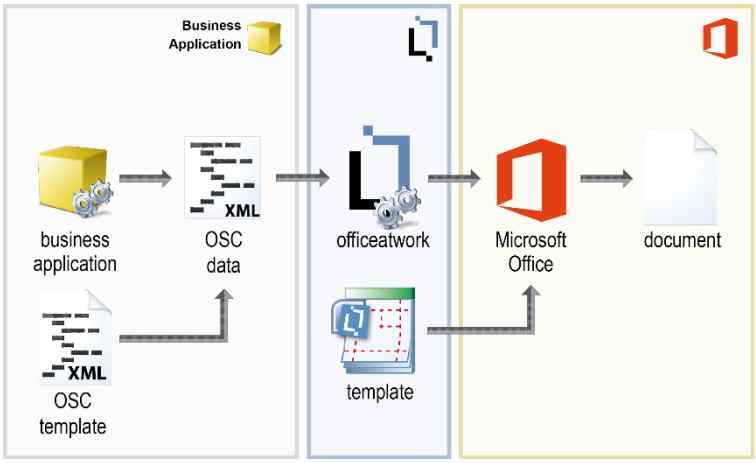

Figure 13: recommended architecture for creating an Office document out of your business application via officeatwork

This architecture has many advantages:

- Limited programming necessary – the business application only needs to be taught how to process the OSC template file. This can be implemented so that new data items available in your business application will not require the reprogramming of the interface to officeatwork. Neither does new officeatwork functionality require a reprogramming of the interface.
- Optimal job sharing – by separating the interface into different parts (Process, Definitions, Design) the development cycle and complexity is kept to a minimum.
- Easy testing – as the final result coming from your business application is an OSC file, you do not need to wait until the programming is finished to test the interface. You can just create sample OSC files and double-click them to test your definitions and design.
- Simple Debugging – as the result coming from the business application is a file, it can easily be analysed. You can easily isolate individual parts of the file to find out any mistakes in the document creation process.

Samples

Sample 1:

This simple OSC template file creates an Invoice summary document and prints it with an officeatwork output-management variant. The document is closed without saving.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<Parameters>
  <CreateDocument>
    <TemplateID>Invoice</TemplateID>
    <ShowDocumentWizard>0</ShowDocumentWizard>
    <Language>{{Field("account_lcid")}}</Language>
    <MasterProperties>
      <MasterProperty IDName="Company" Where="IDName" Is="{{Field("organisation_name")}}"/>
      <MasterProperty IDName="Contactperson" Where="IDName" Is="{{Field("contactperson_name")}}" />
      <MasterProperty IDName="Signature1" Where="IDName" Is="{{Field("signature1_name")}}" />
      <MasterProperty IDName="Signature2" Where="IDName" Is="{{Field("signature2_name")}}" />
      <MasterProperty IDName="Recipient">
        <Field Name="CompleteAddress" Value="{{Field("invoice_address")}}"/>
      </MasterProperty>
      <MasterProperty IDName="CustomField">
        <Field Name="DocumentType" Value="{{Translation("invoice")}}"/>
        <Field Name="YourReference" Value="{{Field("invoice_ref")}}"/>
        <Field Name="Account" Value="{{Field("account_displayname")}}"/>
        <Field Name="Project" Value="{{Field("project_displayname")}}"/>
      </MasterProperty>
    </MasterProperties>
    <Bookmarks>
      <Bookmark Name="Subject" Value="{{Translation("number")}} {{Field("invoice_number")}}"/>
    </Bookmarks>
    <Contents>
      <Content ID = "Invoice Summary Title"></Content>
{{Loop("IncidentTotals", "
      <Content ID = "Invoice Summary Item">
        <Value Name="incident_type" Value="{{LoopField("IncidentTotals", "incident_type")}}"/>
        <Value Name="total" Value="{{LoopField("IncidentTotals", "total")}}"/>
        <Value Name="vat" Value="{{LoopField("IncidentTotals", "vat")}}"/>
      </Content>
")}}
{{Loop("InvoiceTotals", "
      <Content ID = "Invoice Summary Total">
        <Value Name="total_incl_vat" Value="{{LoopField("InvoiceTotals", "total_incl_vat")}}"/>
        <Value Name="total_vat_relevant" Value="{{LoopField("InvoiceTotals", "total_vat_relevant")}}"/>
        <Value Name="total_vat_irrelevant" Value="{{LoopField("InvoiceTotals", "total_vat_irr")}}"/>
        <Value Name="vat" Value="{{LoopField("InvoiceTotals", "vat")}}"/>
      </Content>
")}}
      <Content ID = "Invoice Accounting Details"></Content>
    </Contents>
    <Output>
      <Print UID="{{Field("printprofile_uid")}}" ShowDialog="0" />
    </Output>
    <CloseDocument>-1</CloseDocument>
  </CreateDocument>
</Parameters>
```

Please note that we recommend using {{ and }} as identifier for business application placeholders. Make sure you have removed all placeholders in the final OSC file that gets passed on to officeatwork.

Sample 2:

This sample creates two different documents. The first is a summary document used in the accounting department. The second is a document concerning the same data but offers more details. It is printed for internal use as well as an original to be sent to the customer. In the end the documents are closed without saving. The whole process is fully automated and requires no interaction from the user.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<Parameters>

{{Loop("Documents", "
  <CreateDocument>
    <TemplateID>Letter</TemplateID>
    <ShowDocumentWizard>0</ShowDocumentWizard>
    <Language>{{LoopField("Documents", "account_lcid")}}</Language>
    <MasterProperties>
      <MasterProperty IDName="Company" Where="IDName" Is="{{LoopField("Documents", "organisation")}}" />
      <MasterProperty IDName="Contactperson" Where="IDName" Is="{{LoopField("contactperson")}}" />
      <MasterProperty IDName="Signature1" Where="IDName" Is="{{LoopField("signature1")}}" />
```

```
      <MasterProperty IDName="Signature2" Where="IDName" Is="{{LoopField("signature2")}}" />
      <MasterProperty IDName="Recipient">
        <Field Name="CompleteAddress" Value="{{LoopField("Documents", "invoice_address")}}" />
      </MasterProperty>
      <MasterProperty IDName="CustomField">
        <Field Name="DocumentType" Value="{{Translation("invoice")}}" />
        <Field Name="YourReference" Value="{{LoopField("Documents", "invoice_ref")}}" />
        <Field Name="Account" Value="{{LoopField("Documents", "account_displayname")}}" />
        <Field Name="Project" Value="{{LoopField("Documents", "project_displayname")}}" />
      </MasterProperty>
    </MasterProperties>
    <Bookmarks>
      <Bookmark Name="Subject" Value="{{Translation("number")}} {{LoopField("Documents", "invoiceno")}}" />
    </Bookmarks>
    <Contents>
      <Content ID = "Invoice Summary Title" />

{{Loop("IncidentTotals", "
      <Content ID = "Invoice Summary Item">
        <Value Name="incident_type" Value="{{LoopField("IncidentTotals", "incident_type")}}" />
        <Value Name="total" Value="{{LoopField("IncidentTotals", "total")}}" />
        <Value Name="vat" Value="{{LoopField("IncidentTotals", "vat")}}" />
      </Content>
")}}

{{Loop("InvoiceTotals", "
      <Content ID = "Invoice Summary Total">
        <Value Name="total_incl_vat" Value="{{LoopField("InvoiceTotals", "total_incl_vat")}}" />
        <Value Name="total_vat_relevant" Value="{{LoopField("InvoiceTotals", "total_vat_relevant")}}" />
        <Value Name="total_vat_irrelevant" Value="{{LoopField("InvoiceTotals", "total_vat_irr")}}" />
        <Value Name="vat" Value="{{LoopField("InvoiceTotals", "vat")}}" />
      </Content>
")}}

      <Content ID = "Invoice Accounting Details" />
    </Contents>
    <Output>
      <Print UID="{{Field("printprofile_uid")}}" ShowDialog="0" />
    </Output>
    <CloseDocument>-1</CloseDocument>
  </CreateDocument>

  <CreateDocument>
    <TemplateID>Letter</TemplateID>
    <ShowDocumentWizard>0</ShowDocumentWizard>
    <Language>{{LoopField("Documents", "account_lcid")}}</Language>
    <MasterProperties>
      <MasterProperty IDName="Company" Where="IDName" Is="{{LoopField("Documents", "organisation")}}" />
      <MasterProperty IDName="Contactperson" Where="IDName" Is="{{LoopField("contactperson")}}" />
      <MasterProperty IDName="Signature1" Where="IDName" Is="{{LoopField("signature1")}}" />
      <MasterProperty IDName="Signature2" Where="IDName" Is="{{LoopField("signature2")}}" />
      <MasterProperty IDName="Recipient">
        <Field Name="CompleteAddress" Value="{{LoopField("Documents", "invoice_address")}}" />
      </MasterProperty>
      <MasterProperty IDName="CustomField">
        <Field Name="DocumentType" Value="{{Translation("invoice")}}" />
        <Field Name="YourReference" Value="{{LoopField("Documents", "invoice_ref")}}" />
        <Field Name="Account" Value="{{LoopField("Documents", "account_displayname")}}" />
        <Field Name="Project" Value="{{LoopField("Documents", "project_displayname")}}" />
      </MasterProperty>
    </MasterProperties>
    <Bookmarks>
      <Bookmark Name="Subject" Value="{{Translation("number")}} {{LoopField("Documents", "invoiceno")}}" />
    </Bookmarks>
    <Contents>
      <Content ID = "Invoice Summary Title"></Content>

{{Loop("IncidentTotals", "
      <Content ID = "Invoice Summary Item">
        <Value Name="incident_type" Value="{{LoopField("IncidentTotals", "incident_type")}}" />
        <Value Name="total" Value="{{LoopField("IncidentTotals", "total")}}" />
        <Value Name="vat" Value="{{LoopField("IncidentTotals", "vat")}}" />
      </Content>
")}}

{{Loop("InvoiceTotals", "
      <Content ID = "Invoice Summary Total">
        <Value Name="total_incl_vat" Value="{{LoopField("InvoiceTotals", "total_incl_vat")}}" />
        <Value Name="total_vat_relevant" Value="{{LoopField("InvoiceTotals", "total_vat_relevant")}}" />
        <Value Name="total_vat_irrelevant" Value="{{LoopField("InvoiceTotals", "total_vat_irr")}}" />
        <Value Name="vat" Value="{{LoopField("InvoiceTotals", "vat")}}" />
      </Content>
")}}

      <Content ID = "Invoice Payment Instructions" />
      <Content ID = "Invoice Marketing" />
      <Content ID = "Invoice Journal Title" />
```

```
{{Loop("Journal", "
      <Content ID = "Invoice Journal Subtitle">
        <Value Name="incident_type" Value="{{LoopField("Journal", "incident_type")}}" />
      </Content>

{{Loop("JournalItems", "
      <Content ID = "Invoice Journal Item">
        <Value Name="accounting_date" Value="{{LoopField("JournalItems", "accounting_date")}}" />
        <Value Name="subject" Value="{{LoopField("JournalItems", "subject")}}" />
        <Value Name="unit_price" Value="{{LoopField("JournalItems", "unit_price")}}" />
        <Value Name="quantity" Value="{{LoopField("JournalItems", "quantity")}}" />
        <Value Name="unit" Value="{{LoopField("JournalItems", "unit")}}" />
        <Value Name="total_without_discount" Value="{{LoopField("JournalItems", "total_without_dis")}}" />
        <Value Name="discount_pct" Value="{{LoopField("JournalItems", "discount_pct")}}" />
        <Value Name="discount_description" Value="{{LoopField("JournalItems", "discount_description")}}" />
        <Value Name="discount" Value="{{LoopField("JournalItems", "discount")}}" />
        <Value Name="total" Value="{{LoopField("JournalItems", "total")}}" />
        <Value Name="vat" Value="{{LoopField("JournalItems", "vat")}}" />
      </Content>
")}}

")}}

    </Contents>
    <Output>
      <Print UID="{{Field("printprofile_uid")}}" ShowDialog="0" />
      <Print UID="{{Field("printprofile_uid2")}}" ShowDialog="0" />
    </Output>
    <CloseDocument>-1</CloseDocument>
  </CreateDocument>
")}}

</Parameters>
```

CHAPTER 3

officeatwork Methods

Introduction

The officeatwork methods are available from an ActiveX component that allows you to interact with officeatwork.

When calling an officeatwork methods, remember to include the officeatwork ActiveX API-Component named **officeatwork API** in your project. Otherwise the methods are not available to you. The component is located in the **ClientSuite Folder** within your officeatwork application directory. The filename of the component is **oawAPI.exe**

ExecuteXML

Executes the passed XML parameter.

Syntax

```
ExecuteXML ( pParam : String ) : Long
```

Parameters

The function **ExecuteXML** has the following parameters:

Name	Description
pParam	String. An XML string containing officeatwork specific instructions. The string has to be structured in a predefined format.

Return value

The **ExecuteXML** function returns the following values:

Type	Description
Long	can have one of the following values: **-1**, representing a successful execution – this means all instructions defined in the parameter were successfully executed. **0**, representing the value of an unsuccessful execution – this means one or many instructions were not successfully executed.

TemplateChooser

Shows the Template Chooser, with all chooser roots also shown in the Template Chooser in the officeatwork system tray menue, to allow the user to choose one template.

Syntax

```
TemplateChooser ( pParam : String ) : Long
```

Parameters

The function **TemplateChooser** has the following parameters:

Name	Description
pParam	String. An XML string containing officeatwork specific instructions. The string has to be structured in a predefined format.

```
<Parameters>
  <Solution></Solution>
</Parameters>
```

The XML can contain the following elements:

Element	Description
Solution	Optional string element. The solution IDName to be matched to the current solution's IDName. If the current solution does not match this solution IDName, then no Template Chooser is shown and the return parameter will state in unseccessfull execution. If this element is omitted, the current solution is used to show the Template Chooser.

Return value

The **TemplateChooser** function returns an XML string:

```
<Results>
  <Successful></Successful>
  <Solution></Solution>
  <TemplateID></TemplateID>
  <TemplatePath></TemplatePath>
  <TemplateFilename></TemplateFilename>
  <TemplateFullname></TemplateFullname>
</Results>
```

The XML can contain the following elements:

Element	Description
Successfull	Required string element. Can contain one of the following values: **-1** represents a successful execution **0** represents an unsuccessful execution
Solution	Optional string element. If a solution is loaded at the time of execution, thie element will contain the IDName of the current solution. If no solution is currently available this element is omitted.
TemplateID	Optional string element. The filename of the chosen template without the path and the file extension. If the chosen file is a Windows or officeatwork shortcut file, then the destination of the shortcut is returned. If no filename is chosen or the file does not exist, this element is omitted.
TemplatePath	Optional string element. The path without the filename of the chosen template. If the chosen file is a Windows or officeatwork shortcut file, then the destination of the shortcut is returned. If no filename is chosen or the file does not exist, this element is omitted.
TemplateFilename	Optional string element.The filename without the path of the chosen template. If the chosen file is a Windows or officeatwork shortcut file, then the destination of the

	shortcut is returned. If no filename is chosen or the file does not exist, this element is omitted.
TemplateFullname	Optional string element.The complete filename with path of the chosen template. If the chosen file is a Windows or officeatwork shortcut file, then the destination of the shortcut is returned. If no filename is chosen or the file does not exist, this element is omitted.

CHAPTER 4

XML Parameter structure and conventions

Introduction

The XML parameter conforms to the following XML structure/convention.

Root Elements

Within the parameters, root elements can be used to create or edit files. Each root element can contain instruction elements. An alphabetically list of the instruction elements with their explanation can be found later in this manual. The following table shows an overview of these root elements and the corresponding instruction elements. The availability is shown with one of the following two bullets:

- The functionality of these instruction elements are identically in all root elements these are available in.
- The functionality of these instruction elements slightly differ in the various root elements. See instruction element descriptions for more details.

CreateDocument

Allows you to create a new document. All its sub-elements (instruction elements) describe how to create the document. To create multiple documents, add multiple `CreateDocument` root elements.

Syntax

```
<CreateDocument>
</CreateDocument>
```

Sample

This sample will create a new document based on the Letter template.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<Parameters>
   <CreateDocument>
      <TemplateID>Letter</TemplateID>
   </CreateDocument>
</Parameters>
```

EditDocument

Allows you to edit an existing document. All its sub-elements (instruction elements) describe how to edit the document. To edit multiple documents, add multiple `EditDocument` root elements.

Syntax

```
<EditDocument>
</EditDocument>
```

Sample

This sample will open an existing document and change the content of the subject bookmark to «This is my new subject».

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<Parameters>
  <EditDocument>
    <DocumentFullName>%Documents%\ExampleLetter.doc</DocumentFullName>
    <Bookmarks>
      <Bookmark Name="Subject" Value="This is my new subject"/>
    </Bookmarks>
  </EditDocument>
</Parameters>
```

CreatePresentation

Allows you to create a new presentation. All its sub-elements (instruction elements) describe how to create the presentation. To create multiple presentations, add multiple `CreatePresentation` root elements.

Syntax

```
<CreatePresentation>
</CreatePresentation>
```

Sample

This sample will create a new presentation based on the Presentation template.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<Parameters>
   <CreatePresentation>
      <TemplateID>Presentation</TemplateID>
   </CreatePresentation>
</Parameters>
```

CreateWorkbook

Allows you to create a new workbook. All its sub-elements (instruction elements) describe how to create the workbook. To create multiple workbooks, add multiple `CreateWorkbook` root elements.

Syntax

```
<CreateWorkbook>
</CreateWorkbook>
```

Sample

This sample will create a new workbook based on the Workbook template.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<Parameters>
   <CreateWorkbook>
      <TemplateID>Workbook</TemplateID>
   </CreateWorkbook>
</Parameters>
```

Instruction Elements

The following instruction elements are available:

- DocumentFullName
- ProtectionType
- Password
- Language
- TemplateID
- ShowDocumentWizard
- ShowCustomDialog
- Contents
- Profile
- MasterProperties
- DocumentVariables
- CustomDocumentProperty
- BuiltInDocumentProperties
- Bookmarks
- Output
- SaveAsLocation
- CloseDocument
- ReplaceExisting

Bookmarks

Set text to Word bookmarks.

Syntax

```
<Bookmarks>
   <Bookmark Name="" Value="" />
</Bookmarks>
```

Content

The `Bookmarks` element can contain the following sub-elements:

Name	Description
Bookmark	Optional to many elements. This element represents an individual Word bookmark.

The `Bookmark` element has the following attributes:

Name	Description
Name	Required string. The name of the Word bookmark.
Value	Required string. The value to be allocated to the Word bookmark.

Sample

```
<Bookmarks>
   <Bookmark Name="Subject" Value="This is the Subject" />
   <Bookmark Name="Text" Value="And this is the text" />"
</Bookmarks>
```

BuiltInDocumentProperties

Set the value of Word built-in document properties.

Syntax

```
<BuiltInDocumentProperties>
   <BuiltInDocumentProperty Name="" Value="" />
</BuiltInDocumentProperties>
```

Content

The `BuiltInDocumentProperties` **element can contain the following sub-elements:**

Name	Description
BuiltInDocumentProperty	Optional to many elements. This element represents an individual Word built-in document property.

The `BuiltInDocumentProperty` **element has the following attributes:**

Name	Description
Name	Required string. The name of the Word built-in document property.
Value	Required string. The value to be allocated to the Word built-in document property.

List of modifiable built-in document properties

- Title
- Subject
- Author
- Keywords
- Category
- Comments
- Manager
- Company

Sample

```
<BuiltInDocumentProperties>
   <BuiltInDocumentProperty Name="Author" Value="Harry Smith" />
   <BuiltInDocumentProperty Name="Subject" Value="Offer B763Gk" />
</BuiltInDocumentProperties>
```

CloseDocument

Defines if the active document shall be closed or left open.

Syntax

```
<CloseDocument></CloseDocument>
```

Content

String. Defines if the document will be closed.

Value	Description
-1	The document will be closed.
0	Default. The document will remain open.

Sample

```
<CloseDocument>-1</CloseDocument>
```

Contents

Inserts Smart-Contents to the document. Single contents or all contents in a specific folder can be added. Values can also be passed along with your contents.

Syntax

```
 <Contents>
<ShowContentChooser></ShowContentChooser>
<Content ID="" LCID="">
    <Value Name="" Value="" />
</Content>
<ContentFolder ID="" LCID="" />
 </Contents>
```

Content

The `Contents` element can contain the following sub-elements:

Name	Description
ShowContentChooser	Optional string element. This element is only considered in root element CreatePresentation in the officeatwork Client Suite and defines if the Content Chooser is shown to the user. 0 is the default value and will not show the Content Chooser -1 will show the Content Chooser Content files added with the following elements are already preselected in the Content Chooser. The user can add more content files or remove the already preselected content files.
Content	Optional to many elements. This element represents an individual content, which will be inserted into the document or will be preselected in the Content Chooser as explained in the element `ShowContentChooser`.
ContentFolder	Optional to many elements. This element represents a folder containing numerous content files. The content files in this folder and the content files in the sub-folders are inserted recursively in alphabetically order – sub-folders before files – or they will be preselected in the Content Chooser as explained in the element `ShowContentChooser`.

The `Content` element has the following attributes and sub-elements:

Name	Description
ID	Required string. The filename without the path and the file extension of the Smart-Content.
LCID	Optional string. This string is only considered in root elements CreateDocument and EditDocument in the officeatwork Client Suite and represents the language ID used to insert the content. If omitted the language ID of the destination document will apply.
Value	Optional to many elements. This element contains a single name-value pair. If the associated content contains an appropriate value document function (see appendix for document functions), the value of the value element will be placed at the position of the document function in the process of inserting the content into the target document.

The `Value` element has the following attributes:

Name	Description
Name	Required string. The name of the value. Used to match a value document function.
Value	Required string. The value that is used as a result of the corresponding value document function.

The `ContentFolder` element has the following attribute:

Name	Description
ID	Required string. The name without the path of the folder containing content files.
LCID	Optional string. This string is only considered in root elements CreateDocument and EditDocument in the officeatwork Client Suite and represents the language ID used to insert the content files. If omitted the language ID of the destination document will apply.

Sample 1

```
<Contents>
   <Content ID = "01 Introduction">
      <Value Name="Object" Value="Three trees" />
   </Content>
   <Content ID="01 Explanation" />
   <Content ID="02 Conclusion" LCID="1033" />
</Contents>
```

Sample 2

```
<Contents>
   <ContentFolder ID="User Guide"/>
</Contents>
```

Sample 3

```
<Contents>
   <Content ID="01 Introduction" />
   <Content ID="02 TOC" />
   <ContentFolder ID="User Guide" />
</Contents>
```

Sample 4

```
<Contents>
   <ShowContentChooser>-1</ShowContentChooser>
   <ContentFolder ID="Organisational Overview"/>
</Contents>
```

CustomDocumentProperties

Set Word custom document properties. If the custom document property does not exist, it will be created.

Syntax

```
<CustomDocumentProperties>
  <CustomDocumentProperty Name="" Value="" />
</CustomDocumentProperties>
```

Content

The `CustomDocumentProperties` **element can contain the following sub-elements:**

Name	Description
CustomDocumentProperty	Optional to many elements. This element represents an individual Word custom document property.

The `CustomDocumentProperty` **element has the following attributes:**

Name	Description
Name	Required string. The name of the Word custom document property.
Value	Required string. The value to be allocated to the Word custom document property.

Sample

```
<CustomDocumentProperties>
   <CustomDocumentProperty Name="Price_ SX12_H4" Value="23.00" />
   <CustomDocumentProperty Name="Price_ L3" Value="12.50" />"
</CustomDocumentProperties>
```

DocumentFullName

Defines the name and path of an existing document to edit.

Syntax

```
<DocumentFullName></DocumentFullName>
```

Content

The complete path and file name of an existing document.

Sample

```
<DocumentFullName>G:\Reports\Report2006.doc</DocumentFullName>
```

DocumentVariables

Sets Word document variables. If the document variable does not exist, it will be created.

Syntax

```
<DocumentVariables>
   <DocumentVariable Name="" Value="" />
</DocumentVariables>
```

Content

The `DocumentVariables` **element can contain the following sub-elements:**

Name	Description
DocumentVariable	Optional to many elements. This element represents an individual Word document variable.

The `DocumentVariable` **element has the following attributes:**

Name	Description
Name	Required string. The name of the Word document variable.
Value	Required string. The value to be allocated to the Word document variable.

Sample

```
<DocumentVariables>
   <DocumentVariable Name="MeetingDate" Value="12.03.2004"/>"
   <DocumentVariable Name="MeetingLocation" Value="Room C324"/>"
</DocumentVariables>
```

IgnoreValidation

Defines if the validation in the document wizard shall be ignored.

Syntax

```
<IgnoreValidation></IgnoreValidation>
```

Content

String. Defines if the validation will be ignored.

Value	Description
-1	The validation in the document wizard will be ignored.
0	Default. The validation in the document wizard will be executed.

Sample

```
<IgnoreValidation>-1</IgnoreValidation>
```

Journal

Creates an empty journal file after the created document is closed in Microsoft Office Word.

Syntax

```
<Journal>
  <Filename></Filename>
</Journal>
```

Content

The `Journal` element has the following sub-elements:

Name	Description
Filename	Required string element. Defines the filename with path of the journal file that has to be created.

Sample

```
<Journal>
   <Filename>C:\path\filename.txt</Filename>
</Journal>
```

Language

Sets the document language. If this element is empty or omitted for new documents, the document language can be specified in the document wizard. If the document wizard is not shown, the user's default document language is used.

Syntax

```
<Language></Language>
```

Content

Language ID (LCID). A list of available Language ID's can be found in the Appendix of this manual.

Sample

```
<Language>2055</Language>
```

MasterProperties

Sets officeatwork specific Master-Properties in the document.

Syntax

```
<MasterProperties>
  <MasterProperty IDName="" Where="" Is="" >
    <Field Name="" Value="" />
  </MasterProperty>
</MasterProperties>
```

Content

The `MasterProperties` **element can contain the following sub-elements:**

Name	Description
MasterProperty	Optional to many elements. This element represents an individual officeatwork Master-Property.

The `MasterProperty` **element can contain the following attributes and sub-elements:**

Name	Description
IDName	Required string. The IDName of the Master-Property.
Where	Optional string. The name of the field, which's value is compared to the value passed in the `Is` attribute to search for an entry in the Master-Property data. If omitted or empty, the default Master-Property entry is selected.
Is	Optional string. The value to be compared to the value of the field defined in the `Where` attribute to search for an entry in the Master-Property data. If omitted or empty, the default Master-Property entry is selected.
Field	Optional to many elements. This element represents a single Master-Property field. If a field has an implicit value due to the selection of a profile or a Master-Property, the `Field` element will explicitly overwrite this value.

Remark: The sub-elements `where` and `is` are only in the osc format available.

The `Field` **element has the following attributes:**

Name	Description
Name	Required string. The name of the Master-Property field.
Value	Required string. The value to be passed to the Master-Property field.

Sample

```
<MasterProperties>
   <MasterProperty IDName="Organisation" Where="IDName" Is="Headquarters" />
   <MasterProperty IDName="Contactperson" Where="IDName" Is="Sample Peter" />
   <MasterProperty IDName="Signature1" Where="IDName" Is="Sample Peter">
      <Field Name="Function" Value="Clerk" />
   </MasterProperty>
   <MasterProperty IDName="Signature2" Where="UID" Is="2003121817293296325874" />
   <MasterProperty IDName="Recipient">
      <Field Name="CompleteAddress">Sample Ltd.
Marketing
Mrs. Jolanda Sample
Street 12
1111 City</Field>
      <Field Name="Company">Sample Ltd.</Field>
      <Field Name="Department">Marketing</Field>
      <Field Name="FullName">Mrs. Jolanda Sample</Field>
      <Field Name="AddressStreet">Street 12</Field>
      <Field Name="AddressZIP">1111</Field>
      <Field Name="AdressCity">City</Field>
      <Field Name="Telephone">+41 44 444 4444</Field>
      <Field Name="EMail">jolanda.sample@sample.com</Field>
      <Field Name="Introduction">Dear Mrs. Sample</Field>
      <Field Name="Closing">Best Regards</Field>
   </MasterProperty>
</MasterProperties>
```

MasterPropertySet

Sets officeatwork specific Master-Property Set in the document.

Syntax

```
<MasterPropertySet IDName="">
   <MasterProperties>
   </MasterProperties>
</MasterPropertySet>
```

Content

The `MasterPropertySet` **element can contain the following sub-elements:**

Name	Description
MasterProperties	Optional to many elements. This element represents a collection of officeatwork Master-Properties.

The `MasterPropertySet` **element can contain the following attributes and sub-elements:**

Name	Description
IDName	Required string. The IDName of the Master-Property Set.

Sample

```
<MasterPropertySet IDName="FirstSet">
   <MasterProperties>
      <MasterProperty IDName="Recipient">
         <Field Name="CompleteAddress">Sample Ltd.
Marketing
Mrs. Jolanda Sample
Street 12
1111 City</Field>
         <Field Name="Company">Sample Ltd.</Field>
         <Field Name="Department">Marketing</Field>
         <Field Name="FullName">Mrs. Jolanda Sample</Field>
         <Field Name="AddressStreet">Street 12</Field>
         <Field Name="AddressZIP">1111</Field>
         <Field Name="AdressCity">City</Field>
         <Field Name="Telephone">+41 44 444 4444</Field>
         <Field Name="EMail">jolanda.sample@sample.com</Field>
         <Field Name="Introduction">Dear Mrs. Sample</Field>
         <Field Name="Closing">Best Regards</Field>
      </MasterProperty>
   </MasterProperties>
</MasterPropertySet>
<MasterPropertySet IDName="SecondSet">
   <MasterProperties>
      <MasterProperty IDName="Recipient">
         <Field Name="CompleteAddress">Example Corporation
Marketing
Mr. John Miller
Avenue 4th
3456 City</Field>
         <Field Name="Company">Example Corporation</Field>
         <Field Name="Department>Marketing</Field>
         <Field Name="FullName">Mr. John Miller</Field>
         <Field Name="AddressStreet">Avenue 4th</Field>
         <Field Name="AddressZIP">3456</Field>
         <Field Name="AdressCity">City</Field>
         <Field Name="Telephone">+41 55 555 5555</Field>
         <Field Name="EMail">john.miller@examplecorporation.com</Field>
         <Field Name="Introduction">Dear Mr. Miller</Field>
         <Field Name="Closing">Best Regards</Field>
      </MasterProperty>
   </MasterProperties>
</MasterPropertySet>
```

MasterPropertySets

Sets officeatwork specific Master-Properties in the document.

Syntax

```
<MasterPropertySets>
     <MasterPropertySet IDName="">
  </MasterPropertySets>
```

Content

The `MasterPropertySets` **element can contain the following sub-elements:**

Name	Description
MasterPropertySet	Optional to many elements. This element represents an individual officeatwork Master-Propertyset.

The `MasterPropertySets` **element has not attributes.**

Sample

```
<MasterPropertySets>
   <MasterPropertySet IDName="FirstSet">
      <MasterProperties>
         <MasterProperty IDName="Recipient">
            <Field Name="CompleteAddress">Sample Ltd.
Marketing
Mrs. Jolanda Sample
Street 12
1111 City</Field>
            <Field Name="Company">Sample Ltd.</Field>
            <Field Name="Department">Marketing</Field>
            <Field Name="FullName">Mrs. Jolanda Sample</Field>
            <Field Name="AddressStreet">Street 12</Field>
            <Field Name="AddressZIP">1111</Field>
            <Field Name="AdressCity">City</Field>
            <Field Name="Telephone">+41 44 444 4444</Field>
            <Field Name="EMail">jolanda.sample@sample.com</Field>
            <Field Name="Introduction">Dear Mrs. Sample</Field>
            <Field Name="Closing">Best Regards</Field>
         </MasterProperty>
      </MasterProperties>
   </MasterPropertySet>
   <MasterPropertySet IDName="SecondSet">
      <MasterProperties>
         <MasterProperty IDName="Recipient">
            <Field Name="CompleteAddress">Example Corporation
Marketing
Mr. John Miller
Avenue 4th
3456 City</Field>
            <Field Name="Company">Example Corporation</Field>
            <Field Name="Department">Marketing</Field>
            <Field Name="FullName">Mr. John Miller</Field>
            <Field Name="AddressStreet">Avenue 4th</Field>
            <Field Name="AddressZIP">3456</Field>
            <Field Name="AdressCity">City</Field>
            <Field Name="Telephone">+41 55 555 5555</Field>
            <Field Name="EMail">john.miller@examplecorporation.com</Field>
            <Field Name="Introduction">Dear Mr. Miller</Field>
            <Field Name="Closing">Best Regards</Field>
         </MasterProperty>
      </MasterProperties>
      …
   </MasterPropertySet>
</MasterPropertySets>
```

Output

Uses one or more of the officeatwork output variants to output the document.

Syntax

```
<Output>
   <Open />
   <Print UID="" ShowDialog="" />
   <Send UID="" ShowEmail="" Filename="" />
   <Save UID="" ShowDialog="" Path="" Filename="" ReplaceExisting="" >
      <Open>
   </Save>
</Output>
```

Content

The `Output` element can contain the following sub-elements:

Name	Description
Print	Optional to many elements. This element triggers an officeatwork print variation to be executed.
Send	Optional to many elements. This element triggers an officeatwork send variation to be executed.

Save	Optional to many elements. This element triggers an officeatwork save variation to be executed.

The `Print` element can contain the following attributes:

Name	Description
UID	Required string. Represents the UID of the desired officeatwork print variant.
ShowDialog	Optional element. Indicates if the print dialogue should be presented to the user during the print process. **-1** is the default value and will show the dialogue **0** will suppress the dialogue

The `Send` element can contain the following attributes:

Name	Description
UID	Required string. Represents the UID of the desired officeatwork send variant.
ShowEmail	Optional string. Indicates if the e-mail should be presented to the user during the send process. **-1** is the default value and will show the dialogue **0** will not show the e-mail
Filename	Optional string. Depending on the configuration in the officeatwork solution and the output method instruction (OOMI) file, this attribute defines the file name for the attachment.

The `Save` element can contain the following attributes:

Name	Description
UID	Required string. Represents the UID of the desired officeatwork save variant.
ShowDialog	Optional string. Indicates if the save dialogue should be presented to the user during the save process. **-1** is the default value and will show the dialog **0** will suppress the dialogue.
Path	Optional string. Depending on the configuration in the officeatwork solution and the output method instruction (OOMI) file, this attribute defines the path for the file to be saved to.
Filename	Optional string. Depending on the configuration in the officeatwork solution and the output method instruction (OOMI) file, this attribute defines the file name to be used for the new file.
ReplaceExisting	Optional string. Defines if an existing file shall be replaced with the new one. **-1** will replace any existing file **0** is the default value and will not save the new file if a file already exists at the same destination location.
Open	Optional element. This element triggers to open the document at the end of the process.

Sample 1

```
<Output>
   <Print UID="64756436753645756" ShowDialog="-1"/>
</Output>
```

Sample 2

```
<Output>
   <Send UID="576867876875676546" ShowEmail="-1" Filename="Report.pdf"/>
</Output>
```

Sample 3

```
<Output>
   <Save UID="345345685746547546775" ShowDialog="-1" Path="C:\Reports\" Filename="Report.pdf"
ReplaceExisting="-1"/>
</Output>
```

Sample 4

```
<Output>
   <Save UID="345345685746547546775" ShowDialog="-1" Path="C:\Reports\" Filename="Report.pdf"
ReplaceExisting="-1" >
       <Open/>
   </Save>
</Output>
```

Password

Sets a password for the document. The document will not be protected only by setting a password. To activate the document protection for the document, the `ProtectionType` element has to be specified as well. If you include the `ProtectionType` element but omit the `Password` element, the document will be protected with no (empty) password.

Syntax

```
<Password></Password>
```

Content

The password to be applied to the document.

Sample

```
<Password>1234</Password>
```

Profile

Defines a user profile for the document.

Syntax

```
<Profile Where="" Is="" />
```

Content

The `Profile` element has the following attributes:

Value	Description
Where	Required string. The name of the field, which's value is compared to the value passed in the `Is` attribute. If omitted or empty, the default user profile is selected.
Is	Required string. The value to be compared to the value of the field defined in the `Where` attribute. If omitted or empty, the default user profile is selected.

Default behaviour

If the `Profile` element is omitted or the `Profile` attributes are missing, the user's default profile is used. If the user has no default profile, the first profile is used. If the user has no profile, no profile is used. If Master-Properties are specified within the XML parameter, these Master-Properties are applied after the profile is applied.

Sample

```
<Profile Where="IDName" Is="Miller Daniela" />
```

ProtectionType

Sets a protection of a specific type for the document. If the protection has to be password protected, then the `Password` element has to be specified as well. If the `Password` element is empty or omitted, then the document will be protected with no (empty) password.

Syntax

```
<ProtectionType></ProtectionType>
```

Content

Value representing the type of protection you want to apply to your document.
-1 is the default value and stands for no protection.
2 stands for forms protection.

Sample

```
<ProtectionType>2</ProtectionType>
```

ReplaceExisting

Defines whether an already existing file with the same filename at the same location will be replaced. This element is used in conjunction with the `SaveAsLocation` element.

Syntax

```
<ReplaceExisting></ReplaceExisting>
```

Content

String. Defines whether the file will be repaced if the file specified in the `SaveAsLocation` already exists.

Value	Description
-1	The existing file will be replaced.
0	Default. The active document will not be saved, if another document already exists with the same filename at the same location.

Sample

```
<ReplaceExisting>-1</ReplaceExisting>
```

Save

Defines if the document is saved after editing an existing document.

Syntax

```
<Save></Save>
```

Content

String. Defines if the document is saved.

Value	Description
-1	The document will be saved.
0	Default. The document will not be saved.

Sample

```
<Save>-1</Save>
```

SaveAsLocation

Defines the location where an active document shall be saved. If the file already exists, then the element `ReplaceExisting` can be used to define whether the file will be replaced.

Syntax

```
<SaveAsLocation></SaveAsLocation>
```

Content

The content of the SaveAsLocation element is a path and filename.

Sample

```
<SaveAsLocation>G:\Invoices\Invoice 5968463.doc</SaveAsLocation>
```

ServerProperties

Sets the content of ServerProperties in Office 2007 and newer documents.

Syntax

```
<ServerProperties>
   <ServerProperty Name="" Value="" />
</ServerProperties>
```

Content

The `ServerProperties` **element can contain the following sub-elements:**

Name	Description
ServerProperty	Optional to many elements. This element represents an individual server property.

The `ServerProperty` **element has the following attributes:**

Name	Description
Name	Required string. The name of the server property.
Value	Required string. The value to be allocated to the server property.

Sample

```
<ServerProperties>
   <ServerProperty Name="DocumentType" Value="Invitation"/>
   <ServerProperty Name="Segment" Value="Finance"/>"
</ServerProperties>
```

ShowCustomDialog

Defines if the custom dialogue defined in a Smart-Template will automatically be completed during the document creation process or if it shall halt during that process and present the dialogue to the user for manual completion.

Syntax

```
<ShowCustomDialog></ShowCustomDialog>
```

Content

String. Defines if the custom dialogue is shown to the user.

Value	Description
-1	Default. The custom dialogue will be presented to the user for manual completion.
0	The custom dialogue will automatically run through without user interaction. All the users default settings apply except values set by the XML parameter.

Sample

```
<ShowCustomDialog>0</ShowCustomDialog>
```

ShowDocumentWizard

Defines if the document wizard will automatically be completed during the document creation process or if it shall halt during that process and present the wizard to the user for manual completion.

Syntax

```
<ShowDocumentWizard></ShowDocumentWizard>
```

Content

String. Defines if the wizard is shown to the user.

Value	Description
-1	Default. The wizard will be presented to the user for manual completion.
0	The wizard will automatically run through without user interaction. All the users default settings apply except values set by the XML parameter.

Sample

```
<ShowDocumentWizard>0</ShowDocumentWizard>
```

TableOfContent

Inserts a table of content into the presentation.

Syntax

```
<TableOfContent></TableOfContent>
```

Content

The IDName name of the table of content that will be inserted.

Sample

```
<TableOfContent>Agenda</TableOfContent>
```

TemplateChooserParameters

This element with it's sub-elements predefines the search dialog of the Template-Chooser.

Syntax

```
<TeamplateChooserParamters>
   <TemplateFolder></TemplateFolder>
   <TemplateSearchPattern></TemplateSearchPattern>
   <TemplateChooserProtectedMode></TemplateChooserProtectedMode>
</TemplateChooserParamters>
```

TemplateFolder

String. Defines the root folder of the search dialog.

TemplateSearchPattern

String. Defines the search pattern in the search dialog.

TemplateChooserProtectedMode

String. Defines the protection mode of the search dialog.

Value	Description
-1	The predefined values can't be edited by the user.
0	Default: The predefined values can be edited by the user.

Sample 1

The following example predefines the root folder "03 Marketing" (including subfolders) as search folder. All templates are shown and the user can change the search folder.

```
<TeamplateChooserParamters>
   <TemplateFolder>03 Marketing</TemplateFolder>
</TemplateChooserParamters>
```

Sample 2

The following example predefines the root folder "02 Sales" (including subfolders) as search folder. Where all templates will be lister with the naming pattern "Offer*". The user can not change the folder or the pattern.

```
<TeamplateChooserParamters>
   <TemplateFolder>02 Sales</TemplateFolder>
   <TemplateSearchPattern>Offer</TemplateSearchPattern>
   <TemplateChooserProtectedMode>-1</TemplateChooserProtectedMode>
</TemplateChooserParamters>
```

TemplateID

Defines the template that should be used to create a new document or presentation.

Syntax

```
<TemplateID></TemplateID>
```

Content

A file name of a template without its file extension. Optionally the path and the extension can be added as well. If the path and the extension are omitted, then the first template with the corresponding file name in the current officeatwork solution will be used to create a new document. If this element is omitted, empty or the referenced template could not be found, the user will be prompted to choose a template manually with the Template Chooser.

Sample

```
<TemplateID>Letter</TemplateID>
```

Values

Sets values for the appropriate value document functions within your document.

Syntax

```
<Values>
   <Value Name="" Value="" />
</Values>
```

Content

The `Values` element can contain the following sub-elements:

Name	Description
Value	Optional element. This element represents an individual value.

The `Value` element has the following attributes:

Name	Description
Name	Required string. The name corresponding to the name used in the value document function.
Value	Required string. The value to be allocated to the appropriate value document function.

Sample

```
<Values>
   <Value Name="Description" Value="Ink-Paper 500pc."/>
   <Value Name="Price" Value="4.50"/>"
</Values>
```

VBA Sample

The following example is written in VBA and creates a new document based on the letter template:

```
Sub OfficeatworkApiSample()
    Dim lOawAPI As oawAPI.API
    Dim lParam As String

    lParam = "<Parameters>" & _
                 "<CreateDocument>" & _
                     "<Language>2055</Language>" & _
                     "<TemplateID>Letter</TemplateID>" & _
                     "<SaveAsLocation>C:\MyDocuments\Document002.doc</SaveAsLocation>" & _
                     "<ReplaceExisting>-1</ReplaceExisting>" & _
                     "<CloseDocument>0</CloseDocument>
    lParam = lParam & _
                     "<Contents>" & _
                         "<Content ID=""Karton_SX12_H4""/>" & _
                         "<Content ID=""Schlauch_L3"">" & _
                             "<Value Name=""PositionNr"" Value=""1""/>" & _
                             "<Value Name=""Description"" Value=""Schlauch mit T-Anschluss""/>" & _
                             "<Value Name=""Price"" Value=""31.70""/>" & _
                         "</Content>" & _
                         "<Content ID=""Karton_SX12_H4"" LCID=""2057""/>" & _
                         "<Content ID=""Titel"" Bookmark=""Text"" InsertionMethod=""0""/>" & _
                     "</Contents>"
    lParam = lParam & _
                     "<Profile Where=""IDName"" Is=""Standard""/>" & _
                     "<MasterProperties>" & _
                         "<MasterProperty IDName=""Recipient"">" & _
                             "<Field Name=""Company"" Value=""Muster AG""/>" & _
                             "<Field Name=""Department"" Value=""Marketing""/>" & _
                             "<Field Name=""FullName"" Value=""Frau Jolanda Testerli""/>" & _
                             "<Field Name=""AddressStreet"" Value=""Matten 12""/>" & _
                             "<Field Name=""AddressZIP"" Value=""1111""/>" & _
                             "<Field Name=""AddressCity"" Value=""Musterfingen""/>" & _
                             "<Field Name=""Telephone"" Value=""+41 (0)44 444 4444""/>" & _
                             "<Field Name=""EMail"" Value=""jolanda.testerli@muster.com""/>" & _
                             "<Field Name=""Introduction"" Value=""Sehr geehrte Frau Testerli""/>" & _
                             "<Field Name=""Closing"" Value=""Freundliche Grüsse""/>" & _
                         "</MasterProperty>" & _
                     "</MasterProperties>"
    lParam = lParam & _
                     "<DocumentVariables>" & _
                         "<DocumentVariable Name=""MeetingDate"" Value=""12.03.2004""/>" & _
                         "<DocumentVariable Name=""MeetingPlace"" Value=""Room C324""/>" & _
                     "</DocumentVariables>" & _
                     "<CustomDocumentProperties>" & _
                         "<CustomDocumentProperty Name=""Preis_Karton_SX12_H4"" Value=""23.00""/>" & _
                         "<CustomDocumentProperty Name=""Preis_Schlauch_L3"" Value=""12.50""/>" & _
                     "</CustomDocumentProperties>" & _
                     "<BuiltInDocumentProperties>" & _
                         "<BuiltInDocumentProperty Name=""Author"" Value=""Peter Muster""/>" & _
                         "<BuiltInDocumentProperty Name=""Subject"" Value=""Offerte Gartenbau""/>" & _
                     "</BuiltInDocumentProperties>" & _
                     "<Bookmarks>" & _
                         "<Bookmark Name=""Preis_Schraube_3d6"" Value=""0.25""/>" & _
                         "<Bookmark Name=""Preis_Mutter_3d5"" Value=""0.10""/>" & _
                     "</Bookmarks>"
    lParam = lParam & _
                 "</CreateDocument>" & _
             "</Parameters>"

    Set lOawAPI = New oawAPI.API

    If (lOawAPI.ExecuteXML(lParam) = -1) Then
        MsgBox "successfully created document"
    Else
        MsgBox "the document could not be created"
    End If

    Set lOawAPI = Nothing

End Sub
```

CHAPTER 5

officeatwork «TemplateChooser» Method

Introduction

The officeatwork **TemplateChooser** method is an officeatwork API function that allows you to use the officeatwork Template Chooser in the officeatwork Client Suite. When called, it will present the Template Chooser to the user and will ask him to pick a template. Depending on the user's rights, the selection of the templates could be limited. The return value contains the corresponding TemplateID of the template that the user chose.

When calling the officeatwork **TemplateChooser** method, remember to include the officeatwork ActiveX API-Component named **officeatwork API** in your project. Otherwise the methods are not available to you. The component is located in the **Common Folder** within your officeatwork application directory. The filename of the component is «oawAPI.exe»

Syntax

TemplateChooser (pParam: String) : String

Parameter

The function `TemplateChooser` has the following parameters:

Name	Description
pParam	String. An XML string containing officeatwork specific instructions. The string has to be structured in a predefined format:

```
<Parameters>
</Parameters>
```

Return value

The `TemplateChooser` function returns the following values:

Type	Description
String	An XML string with various elements within the **<Results>** element.

Content

The <Results> element has the following sub-elements

Name	Description
Successful	Indicates whether the function has been successfully completed. **-1** successful **0** failed Sample: `<Successful>-1</Successful>`

Solution	Returns the ID of the currently active officeatwork solution. Sample: <Solution>examplecom</Solution>
TemplateID	Returns the ID of the chosen template. Sample: <TemplateID>Letter</TemplateID>
TemplatePath	Returns the path of the chosen template. Sample: <TemplatePath>C:\officeatwork\Solutions\MySolution\MasterTemplates\</TemplatePath>
TemplateFilename	Returns the filename of the chosen template. Sample: <TemplateFilename>Letter.owt</TemplateFilename>
TemplateFullname	Returns the filename including the path of the chosen template. Sample: <TemplateFullname> C:\officeatwork\Solutions\MySolution\MasterTemplates\Letter.owt </TemplateFullname>

Sample

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<Results>
   <Successful>-1</Successful>
   <Solution>examplesolutioncom</Solution>
   <TemplateID>Letter</TemplateID>
   <TemplatePath>C:\officeatwork\Solutions\MySolution\MasterTemplates\</TemplatePath>
   <TemplateFilename>Letter.owt</TemplateFilename>
   <TemplateFullname>C:\officeatwork\Solutions\MySolution\MasterTemplates\Letter.owt</TemplateFullname>
</Results>
```

Sample

```
Public Sub ShowTempalteChooser()

    Dim lXml As String
    Dim lOawAPI As oawAPI.API

    Set lOawAPI = New oawAPI.API
    lXml = lOawAPI.TemplateChooser("<?xml version=""1.0"" encoding=""ISO-8859-
1""?><Parameters></Parameters>")

    Debug.Print lXml
    Set lOawAPI = Nothing

End Sub
```

Appendix

File System Variables

% Shortcut %	Destination
ADMINTOOLS	<Benutzer>\Startmenü\Programme\Verwaltung
ALTSTARTUP	Startup
APPDATA	<Benutzer>\Anwendungsdaten
COMMONADMINTOOLS	All Users\Startmenü\Programme\Verwaltung
COMMONALTSTARTUP	Common Startup
COMMONAPPDATA	All Users\Anwendungsdaten
COMMONDESKTOPDIRECTORY	All Users\Desktop
COMMONDOCUMENTS	All Users\Dokumente
COMMONFAVORITES	All Users\Favoriten
COMMONPROGRAMS	All Users\Startmenü\Programme
COMMONSTARTMENU	All Users\Startmenü
COMMONSTARTUP	All Users\Startmenü\Autostart
COMMONTEMPLATES	All Users\Vorlagen
COOKIES	<Benutzer>\Cookies
DESKTOP	<Desktop>
DESKTOPDIRECTORY	<Benutzer>\Desktop
FAVORITES	<Benutzer>\Favoriten
FONTS	Windows\Fonts
HISTORY	<Benutzer>\Lokale Einstell.\Verlauf
INTERNET_CACHE	<Benutzer>\Lokale Einstell.\Temp. Internet Files
LOCALAPPDATA	<Benutzer>\Lokale Einstell.\Anwendungsdaten
MYPICTURES	Eigene Bilder
NETHOOD	<Benutzer>\Netzwerkumgebung
OFFICEATWORK	PROGRAMFILES & "\officeatwork"
PERSONAL	Eigene Dateien
PRINTHOOD	<Benutzer>\Druckumgebung
PROFILE	Benutzerprofil
PROGRAMFILES	C:\Programme
PROGRAMFILESCOMMON	C:\Programme\Gemeinsame Dateien
PROGRAMS	Startmenü\Programme
RECENT	<Benutzer>\Recent
SENDTO	<Benutzer>\SendTo
STARTMENU	<Benutzer>\Startmenü
STARTUP	Startmenü\Programme\Autostart
SYSTEM	GetSystemDirectory()
TEMP	TEMP

% Shortcut %	Destination
TEMPLATES	<Benutzer>\Vorlagen
WINDOWS	GetWindowsDirectory()

Remark: The set of available file system variables depends on the operating system version. For further information please check the website of the manufacturer.

LCID's

Language	ID
Afrikaans - South Africa	1078
Albanian - Albania	1052
Amharic - Ethiopia	1118
Arabic - Saudi Arabia	1025
Arabic - Algeria	5121
Arabic - Bahrain	15361
Arabic - Egypt	3073
Arabic - Iraq	2049
Arabic - Jordan	11265
Arabic - Kuwait	13313
Arabic - Lebanon	12289
Arabic - Libya	4097
Arabic - Morocco	6145
Arabic - Oman	8193
Arabic - Qatar	16385
Arabic - Syria	10241
Arabic - Tunisia	7169
Arabic - U.A.E.	14337
Arabic - Yemen	9217
Armenian - Armenia	1067
Assamese	1101
Azeri (Cyrillic)	2092
Azeri (Latin)	1068
Basque	1069
Belarusian	1059
Bengali	1093
Bengali (Bangladesh)	2117
Bosnian (Bosnia/Herzegovina)	5146
Bulgarian	1026
Burmese	1109
Catalan	1027
Cherokee - United States	1116
Chinese - People's Republic of China	2052
Chinese - Singapore	4100
Chinese - Taiwan	1028
Chinese - Hong Kong SAR	3076

Language	ID
Chinese - Macao SAR	5124
Croatian	1050
Croatian (Bosnia/Herzegovina)	4122
Czech	1029
Danish	1030
Divehi	1125
Dutch - Netherlands	1043
Dutch - Belgium	2067
Edo	1126
English - United States	1033
English - United Kingdom	2057
English - Australia	3081
English - Belize	10249
English - Canada	4105
English - Caribbean	9225
English - Hong Kong SAR	15369
English - India	16393
English - Indonesia	14345
English - Ireland	6153
English - Jamaica	8201
English - Malaysia	17417
English - New Zealand	5129
English - Philippines	13321
English - Singapore	18441
English - South Africa	7177
English - Trinidad	11273
English - Zimbabwe	12297
Estonian	1061
Faroese	1080
Farsi	1065
Filipino	1124
Finnish	1035
French - France	1036
French - Belgium	2060
French - Cameroon	11276
French - Canada	3084

Language	ID
French - Democratic Rep. of Congo	9228
French - Cote d'Ivoire	12300
French - Haiti	15372
French - Luxembourg	5132
French - Mali	13324
French - Monaco	6156
French - Morocco	14348
French - North Africa	58380
French - Reunion	8204
French - Senegal	10252
French - Switzerland	4108
French - West Indies	7180
Frisian - Netherlands	1122
Fulfulde - Nigeria	1127
FYRO Macedonian	1071
Gaelic (Ireland)	2108
Gaelic (Scotland)	1084
Galician	1110
Georgian	1079
German - Germany	1031
German - Austria	3079
German - Liechtenstein	5127
German - Luxembourg	4103
German - Switzerland	2055
Greek	1032
Guarani - Paraguay	1140
Gujarati	1095
Hausa - Nigeria	1128
Hawaiian - United States	1141
Hebrew	1037
Hindi	1081
Hungarian	1038
Ibibio - Nigeria	1129
Icelandic	1039
Igbo - Nigeria	1136
Indonesian	1057
Inuktitut	1117
Italian - Italy	1040

Language	ID
Italian - Switzerland	2064
Japanese	1041
Kannada	1099
Kanuri - Nigeria	1137
Kashmiri	2144
Kashmiri (Arabic)	1120
Kazakh	1087
Khmer	1107
Konkani	1111
Korean	1042
Kyrgyz (Cyrillic)	1088
Lao	1108
Latin	1142
Latvian	1062
Lithuanian	1063
Malay - Malaysia	1086
Malay - Brunei Darussalam	2110
Malayalam	1100
Maltese	1082
Manipuri	1112
Maori - New Zealand	1153
Marathi	1102
Mongolian (Cyrillic)	1104
Mongolian (Mongolian)	2128
Nepali	1121
Nepali - India	2145
Norwegian (Bokmål)	1044
Norwegian (Nynorsk)	2068
Oriya	1096
Oromo	1138
Papiamentu	1145
Pashto	1123
Polish	1045
Portuguese - Brazil	1046
Portuguese - Portugal	2070
Punjabi	1094
Punjabi (Pakistan)	2118
Quecha - Bolivia	1131

Language	ID
Quecha - Ecuador	2155
Quecha - Peru	3179
Rhaeto-Romanic	1047
Romanian	1048
Romanian - Moldava	2072
Russian	1049
Russian - Moldava	2073
Sami (Lappish)	1083
Sanskrit	1103
Sepedi	1132
Serbian (Cyrillic)	3098
Serbian (Latin)	2074
Sindhi - India	1113
Sindhi - Pakistan	2137
Singhalese - Sri Lanka	1115
Slovak	1051
Slovenian	1060
Somali	1143
Sorbian	1070
Spanish - Spain (Modern Sort)	3082
Spanish - Spain (Traditional Sort)	1034
Spanish - Argentina	11274
Spanish - Bolivia	16394
Spanish - Chile	13322
Spanish - Colombia	9226
Spanish - Costa Rica	5130
Spanish - Dominican Republic	7178
Spanish - Ecuador	12298
Spanish - El Salvador	17418
Spanish - Guatemala	4106
Spanish - Honduras	18442
Spanish - Latin America	58378
Spanish - Mexico	2058
Spanish - Nicaragua	19466
Spanish - Panama	6154
Spanish - Paraguay	15370
Spanish - Peru	10250
Spanish - Puerto Rico	20490

Language	ID
Spanish - United States	21514
Spanish - Uruguay	14346
Spanish - Venezuela	8202
Sutu	1072
Swahili	1089
Swedish	1053
Swedish - Finland	2077
Syriac	1114
Tajik	1064
Tamazight (Arabic)	414
Tamazight (Latin)	1119
Tamil	1097
Tatar	1092
Telugu	1098
Thai	1054
Tibetan - Bhutan	2129
Tibetan - People's Republic of China	1105
Tigrigna - Eritrea	2163
Tigrigna - Ethiopia	1139
Tsonga	1073
Tswana	1074
Turkish	1055
Turkmen	1090
Uighur - China	1152
Ukrainian	1058
Urdu	1056
Urdu - India	2080
Uzbek (Cyrillic)	2115
Uzbek (Latin)	1091
Venda	1075
Vietnamese	1066
Welsh	1106
Xhosa	1076
Yi	1144
Yiddish	1085
Yoruba	1130
Zulu	1077

API Samples

OSC File-Samples

Bookmark

This sample OSC file shows how to create a document and set the bookmark **Subject** to **my new subject**. The template used is the **letter** template and the document language is set to **English UK (2057)**.

This sample can be found in the officeatwork example solution, which is part of the officeatwork application installer. The sample file is located in the folder **Examples\OSC Files** of the officeatwork solution and is named **Sample Bookmark.osc**.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<Parameters>
   <CreateDocument>
      <Language>2057</Language>
      <TemplateID>Letter</TemplateID>
      <ShowDocumentWizard>0</ShowDocumentWizard>
      <Bookmarks>
         <Bookmark Name="Subject" Value="my new subject"/>
      </Bookmarks>
   </CreateDocument>
</Parameters>
```

ContentFolder

This sample OSC file shows how to create a document that includes all Smart-Contents within the **Produkteofferte** folder including its subfolders. The template used is the **offer** template and the document language is set to **German Switzerland (2055)**.

This sample can be found in the officeatwork example solution, which is part of the officeatwork application installer. The sample file is located in the folder **Examples\OSC Files** of the officeatwork solution and is named **Sample ContentFolder.osc**.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<Parameters>
   <CreateDocument>
      <Language>2055</Language>
      <TemplateID>Offer</TemplateID>
      <ShowDocumentWizard>0</ShowDocumentWizard>
      <Contents>
         <ContentFolder ID = "Produkteofferte"/>
      </Contents>
   </CreateDocument>
</Parameters>
```

CustomFields

This sample OSC file shows how to create a document and set values of custom fields. The template used is the **Master-Template_Custom Fields_Master Properties** template and the document language is set to **English UK (2057)**.

This sample can be found in our officeatwork example solution, which is part of the officeatwork application installer. The sample file is located in the folder **Examples\OSC Files** of the officeatwork solution and is named **Sample CustomFields.osc**.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<Parameters>
   <CreateDocument>
      <Language>2057</Language>
      <TemplateID>Master-Template_Custom Fields_Master Properties</TemplateID>
      <ShowDocumentWizard>-1</ShowDocumentWizard>
      <MasterProperties>
         <MasterProperty IDName="CustomField">
            <Field Name="External" Value="EX3"/>
            <Field Name="Ref" Value="K.4587.322"/>
            <Field Name="No" Value="83672"/>
         </MasterProperty>
      </MasterProperties>
   </CreateDocument>
</Parameters>
```

Letter

This sample OSC file shows how to create a document based on the **Letter** template. The document language is set to **English UK (2057)**.

This sample can be found in our officeatwork example solution, which is part of the officeatwork application installer. The sample file is located in the folder **Examples\OSC Files** of the officeatwork solution and is named **Sample Letter.osc**.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<Parameters>
   <CreateDocument>
      <Language>2057</Language>
      <TemplateID>Letter</TemplateID>
      <ShowDocumentWizard>-1</ShowDocumentWizard>
   </CreateDocument>
</Parameters>
```

Multiple Documents

This sample OSC file shows how to create multiple documents in one go. The template used is the **letter** template and the document language is set to **German Switzerland (2055)**.

This sample can be found in our officeatwork example solution, which is part of the officeatwork application installer. The sample file is located in the folder **Examples\OSC Files** of the officeatwork solution and is named **Sample Multiple Documents.osc**.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<Parameters>
    <CreateDocument>
        <Language>2055</Language>
        <TemplateID>Letter</TemplateID>
        <ShowDocumentWizard>0</ShowDocumentWizard>
        <SaveAsLocation>%DESKTOP%\Letter Version 1.doc</SaveAsLocation>
        <ReplaceExisting>-1</ReplaceExisting>
        <CloseDocument>-1</CloseDocument>
        <Contents>
            <Content ID = "01 Firmenbildung"></Content>
            <Content ID = "02 Handelsregister"></Content>
            <Content ID = "03 Firmenrecherche"></Content>
            <Content ID = "04 Rechtsträger"></Content>
            <Content ID = "05 Firmenschutz"></Content>
        </Contents>
    </CreateDocument>
    <CreateDocument>
        <Language>2055</Language>
        <TemplateID>Letter</TemplateID>
        <ShowDocumentWizard>0</ShowDocumentWizard>
        <SaveAsLocation>%DESKTOP%\Letter Version 2.doc</SaveAsLocation>
        <ReplaceExisting>-1</ReplaceExisting>
        <CloseDocument>-1</CloseDocument>
        <Contents>
            <Content ID = "01 Firmenbildung"></Content>
            <Content ID = "03 Firmenrecherche"></Content>
            <Content ID = "05 Firmenschutz"></Content>
        </Contents>
    </CreateDocument>
    <CreateDocument>
        <Language>2055</Language>
        <TemplateID>Letter</TemplateID>
        <ShowDocumentWizard>0</ShowDocumentWizard>
        <SaveAsLocation>%DESKTOP%\Letter Version 3.doc</SaveAsLocation>
        <ReplaceExisting>-1</ReplaceExisting>
        <CloseDocument>-1</CloseDocument>
        <Contents>
            <Content ID = "02 Handelsregister"></Content>
            <Content ID = "04 Rechtsträger"></Content>
        </Contents>
    </CreateDocument>
</Parameters>
```

Multiple Recipients

This sample OSC file shows how to create a document with multiple recipients. Please note that the first two recipients are marked for printing and that the second recipient will be the active recipient selected in the document wizard and displayed on the document. The template used is the **Letter** template and the document language is set to **English UK (2057)**.

This sample can be found in our officeatwork example solution, which is part of the officeatwork application installer. The sample file is located in the folder **Examples\OSC Files** of the officeatwork solution and is named **Sample Multiple Recipients.osc**.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<Parameters>
    <CreateDocument>
        <Language>2057</Language>
        <TemplateID>Letter</TemplateID>
        <ShowDocumentWizard>-1</ShowDocumentWizard>
        <MasterProperties>
            <MasterProperty IDName="Recipient">
                <Field Name="IDName" Value="Sample Recipient 1"/>
                <Field Name="CompleteAddress" Value="Sample No 1
Address
Street
City"/>
                <Field Name="Company" Value="Company Sample 1"/>
                <Field Name="Department" Value="Department Sample 1"/>
                <Field Name="FullName" Value="FullName Sample 1"/>
                <Field Name="AddressStreet" Value="AddressStreet Sample 1"/>
                <Field Name="AddressZIP" Value="1111"/>
                <Field Name="AdressCity" Value="AdressCity Sample 1"/>
                <Field Name="Telephone" Value="+11 (0)11 111 1111"/>
                <Field Name="EMail" Value="1.1@examplesolution.com"/>
                <Field Name="RecipientPrint" Value="-1"/>
                <Field Name="RecipientActive" Value="0"/>
            </MasterProperty>
            <MasterProperty IDName="Recipient">
                <Field Name="IDName" Value="Sample Recipient 2"/>
                <Field Name="CompleteAddress" Value="Sample No 2
Address
Street
City"/>
                <Field Name="Company" Value="Company Sample 2"/>
                <Field Name="Department" Value="Department Sample 2"/>
                <Field Name="FullName" Value="FullName Sample 2"/>
                <Field Name="AddressStreet" Value="AddressStreet Sample 2"/>
                <Field Name="AddressZIP" Value="2222"/>
                <Field Name="AdressCity" Value="AdressCity Sample 2"/>
                <Field Name="Telephone" Value="+22 (0)22 222 2222"/>
                <Field Name="EMail" Value="2.2@examplesolution.com"/>
                <Field Name="RecipientPrint" Value="-1"/>
                <Field Name="RecipientActive" Value="-1"/>
            </MasterProperty>
            <MasterProperty IDName="Recipient">
                <Field Name="IDName" Value="Sample Recipient 3"/>
                <Field Name="CompleteAddress" Value="Sample No 3
Address
Street
City"/>
                <Field Name="Company" Value="Company Sample 3"/>
                <Field Name="Department" Value="Department Sample 3"/>
                <Field Name="FullName" Value="FullName Sample 3"/>
                <Field Name="AddressStreet" Value="AddressStreet Sample 3"/>
                <Field Name="AddressZIP" Value="3333"/>
                <Field Name="AdressCity" Value="AdressCity Sample 3"/>
                <Field Name="Telephone" Value="+33 (0)33 333 3333"/>
                <Field Name="EMail" Value="3.3@examplesolution.com"/>
                <Field Name="RecipientPrint" Value="0"/>
                <Field Name="RecipientActive" Value="0"/>
            </MasterProperty>
        </MasterProperties>
    </CreateDocument>
</Parameters>
```

Print

This sample OSC file shows how to create a document and print it using an officeatwork output variant. The template used is the **Baubewilligung** template and the document language is set to **German Switzerland (2055)**.

This sample can be found in our officeatwork example solution, which is part of the officeatwork application installer. The sample file is located in the folder **Examples\OSC Files** of the officeatwork solution and is named **Sample Print.osc**.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<Parameters>
   <CreateDocument>
      <Language>2055</Language>
      <TemplateID>Baubewilligung</TemplateID>
      <ShowDocumentWizard>0</ShowDocumentWizard>
      <Output>
         <Print UID="20030107111850943437505 37" ShowDialog="0" />
      </Output>
      <CloseDocument>-1</CloseDocument>
   </CreateDocument>
</Parameters>
```

Recipient

This sample OSC file shows how to create a document and set its recipient information. The template used is the **Letter** template and the document language is set to **English UK (2057)**.

This sample can be found in our officeatwork example solution, which is part of the officeatwork application installer. The sample file is located in the folder **Examples\OSC Files** of the officeatwork solution and is named **Sample Recipient.osc**.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<Parameters>
   <CreateDocument>
      <Language>2057</Language>
      <TemplateID>Letter</TemplateID>
      <ShowDocumentWizard>-1</ShowDocumentWizard>
      <MasterProperties>
         <MasterProperty IDName="Recipient">
            <Field Name="CompleteAddress" Value="Muster AG
Marketing
Mrs. Jolanda Smith
Matten 12
1111 Musterlingen"/>
            <Field Name="Company" Value="Muster AG"/>
            <Field Name="Department" Value="Marketing"/>
            <Field Name="FullName" Value="Mrs. Jolanda Smith"/>
            <Field Name="AddressStreet" Value="Matten 12"/>
            <Field Name="AddressZIP" Value="1111"/>
            <Field Name="AdressCity" Value="Musterlingen"/>
            <Field Name="Telephone" Value="+41 (0)44 444 4444"/>
            <Field Name="EMail" Value="jolanda.smith@examplesolution.com"/>
            <Field Name="Introduction" Value="Dear Mrs. Smith"/>
            <Field Name="Closing" Value="Kind regards"/>
         </MasterProperty>
      </MasterProperties>
   </CreateDocument>
</Parameters>
```

Save

This sample OSC file shows how to create a document and save it using an officeatwork output variant. The template used is the **Baubewilligung** template and the document language is set to **German Switzerland (2055)**.

This sample can be found in our officeatwork example solution, which is part of the officeatwork application installer. The sample file is located in the folder **Examples\OSC Files** of the officeatwork solution and is named **Sample Save.osc**.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<Parameters>
   <CreateDocument>
      <Language>2055</Language>
      <TemplateID>Baubewilligung</TemplateID>
      <ShowDocumentWizard>0</ShowDocumentWizard>
      <Output>
         <Save UID="2006121210441235887611" ShowDialog="0" Path="%Desktop%" Filename="Baubewilligung.pdf"
ReplaceExistingFile="-1" />
      </Output>
      <CloseDocument>-1</CloseDocument>
   </CreateDocument>
</Parameters>
```

Send

This sample OSC file shows how to create a document and send it via email using an officeatwork output variant. The template used is the **Baubewilligung** template and the document language is set to **German Switzerland (2055)**.

This sample can be found in our officeatwork example solution, which is part of the officeatwork application installer. The sample file is located in the folder **Examples\OSC Files** of the officeatwork solution and is named **Sample Send.osc**.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<Parameters>
   <CreateDocument>
      <Language>2055</Language>
      <TemplateID>Baubewilligung</TemplateID>
      <ShowDocumentWizard>0</ShowDocumentWizard>
      <Output>
         <Send UID="200612121039582129211O" ShowEmail="-1" Filename="Baubewilligung.pdf" />
      </Output>
      <CloseDocument>-1</CloseDocument>
   </CreateDocument>
</Parameters>
```

VBA Samples

Content

This code sample shows how to create a document including numerous contents. The template used is the **Letter** template and the document language is set to **German Switzerland (2055)**.

```
Sub OfficeatworkApiSample()

    Dim lOawAPI As oawAPI.API
    Dim lParam As String

    lParam = "<?xml version=""1.0"" encoding=""ISO-8859-1""?>"
    lParam = "<Parameters>" & _
                "<CreateDocument>" & _
                    "<Language>2055</Language>" & _
                    "<TemplateID>Letter</TemplateID>" & _
                    "<ShowDocumentWizard>-1</ShowDocumentWizard>"

    lParam = lParam & _
                    "<Contents>" & _
                        "<Content ID=""Sample""/>" & _
                        "<Content ID=""Sample with values"">" & _
                            "<Value Name=""Beschreibung"" Value=""Karton 1000 Stk. Tischset""/>" & _
                            "<Value Name=""Farbe"" Value=""blau/rot""/>" & _
                            "<Value Name=""Preis"" Value=""123.45""/>" & _
                        "</Content>" & _
                        "<Content ID=""Sample with values"" " & _
                                "LCID=""2057"" Bookmark=""Text"" InsertionMethod=""0"">" & _
                            "<Value Name=""Beschreibung"" Value=""Karton 1000 Stk. Tischset""/>" & _
                            "<Value Name=""Farbe"" Value=""blau/rot""/>" & _
                            "<Value Name=""Preis"" Value=""123.45""/>" & _
                        "</Content>" & _
                    "</Contents>"

    lParam = lParam & _
                "</CreateDocument>" & _
             "</Parameters>"

    Set lOawAPI = New oawAPI.API

    If (lOawAPI.ExecuteXML(lParam) = -1) Then
        MsgBox "Document created successfully."
    Else
        MsgBox "Document could not be created."
    End If

    Set lOawAPI = Nothing

End Sub
```

Letter

This code sample shows how to create a document based on the **letter** template. The document language is set to **German Switzerland (2055)**.

```
Sub OfficeatworkApiSample()

    Dim lOawAPI As oawAPI.API
    Dim lParam As String

    lParam = "<Parameters>" & _
                "<CreateDocument>" & _
                    "<Language>2055</Language>" & _
                    "<TemplateID>Letter</TemplateID>" & _
                    "<ShowDocumentWizard>-1</ShowDocumentWizard>" & _
                "</CreateDocument>" & _
             "</Parameters>"

    Set lOawAPI = New oawAPI.API

    If (lOawAPI.ExecuteXML(lParam) = -1) Then
        MsgBox "Document created successfully."
    Else
        MsgBox "Document could not be created."
    End If

    Set lOawAPI = Nothing

End Sub
```

Multiple Documents

This code sample shows how to create multiple documents in one go. The template used is the **Offer** template and the document language is set to **German Switzeland (2055)**.

```
Sub OfficeatworkApiMultiple()

    Dim lOawAPI As oawAPI.API
    Dim lParam As String

    lParam = "<?xml version=""1.0"" encoding=""ISO-8859-1""?>"
    lParam = "<Parameters>" & _
                "<CreateDocument>" & _
                    "<TemplateID>Offer</TemplateID>" & _
                    "<ShowDocumentWizard>-1</ShowDocumentWizard>" & _
                    "<Language>2055</Language>" & _
                "</CreateDocument>" & _
                "<CreateDocument>" & _
                    "<TemplateID>Letter</TemplateID>" & _
                    "<ShowDocumentWizard>-1</ShowDocumentWizard>" & _
                    "<Language>2055</Language>" & _
                "</CreateDocument>" & _
             "</Parameters>"

    Set lOawAPI = New oawAPI.API

    If (lOawAPI.ExecuteXML(lParam) = -1) Then
        MsgBox "Document created successfully."
    Else
        MsgBox "Document could not be created."
    End If

    Set lOawAPI = Nothing

End Sub
```

Recipient

This code sample shows how to create a document including recipient information. The template used is the **Letter** template and the document language is set to **English UK (2057)**.

```
Sub OfficeatworkApiSample()

    Dim lOawAPI As oawAPI.API
    Dim lParam As String

    lParam = "<?xml version=""1.0"" encoding=""ISO-8859-1""?>"
    lParam = "<Parameters>" & _
                 "<CreateDocument>" & _
                     "<Language>2057</Language>" & _
                     "<TemplateID>Letter</TemplateID>" & _
                     "<ShowDocumentWizard>-1</ShowDocumentWizard>"

    lParam = lParam & _
                     "<MasterProperties>" & _
                         "<MasterProperty IDName=""Recipient"">" & _
                             "<Field Name=""Company"" Value=""Muster AG""/>" & _
                             "<Field Name=""Department"" Value=""Marketing""/>" & _
                             "<Field Name=""FullName"" Value=""Mrs. Jolanda Smith""/>" & _
                             "<Field Name=""AddressStreet"" Value=""Matten 12""/>" & _
                             "<Field Name=""AddressZIP"" Value=""1111""/>" & _
                             "<Field Name=""AddressCity"" Value=""Musterlingen""/>" & _
                             "<Field Name=""Telephone"" Value=""+41 (0)44 444 4444""/>" & _
                             "<Field Name=""EMail"" Value=""jolanda.smith@examplesolution.com""/>" & _
                             "<Field Name=""Introduction"" Value=""Dear Mrs. Smith""/>" & _
                             "<Field Name=""Closing"" Value=""Kind regards""/>" & _
                         "</MasterProperty>" & _
                     "</MasterProperties>"

    lParam = lParam & _
                 "</CreateDocument>" & _
             "</Parameters>"

    Set lOawAPI = New oawAPI.API

    If (lOawAPI.ExecuteXML(lParam) = -1) Then
        MsgBox "Document successfully created."
    Else
        MsgBox "Document could not be created."
    End If

    Set lOawAPI = Nothing

End Sub
```

CHAPTER 6

Support

Get access to a wide range of support resources on officeatwork Connect (connect.officeatwork.com) such as:

- Knowledge Base
- Q & A
- Download Center
- Installers
- Manuals
- Video guides
- Forum
- Glossary
- etc.

To access officeatwork Connect you need to register your Microsoft-Account at www.officeatwork.com → Connect

---

All support options and resources can be found on the website www.officeatwork.com → Support

More services offered by officeatwork such as Education and Consulting can be found on the website www.officeatwork.com → Services

Index

Click here to enter text.