Guideline for Developers

EDC Server Web Services

This guide has been created using officeatwork Advanced.
6. April 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.

Table of Contents

About this guide

For whom is the guide intended

This book has been written for information technology developers that develop applications to interact with officeatwork EDC Server web services.

What is covered in this guide

This manual contains a developing guideline to develop applications with interactions to the EDC Server Webservices.

Knowledge required

You should be familiar with the development of applications consuming web services.

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

EDC Server Web Services overview

The following chapter will help you to develop an application to use and interact with the officeatwork EDC Server Web Services.

The definition for the web service url is not part of this document. You find the definition in the EDC Server Installation Guideline.

Web Service specification overview

Available methods

The following methods are available:

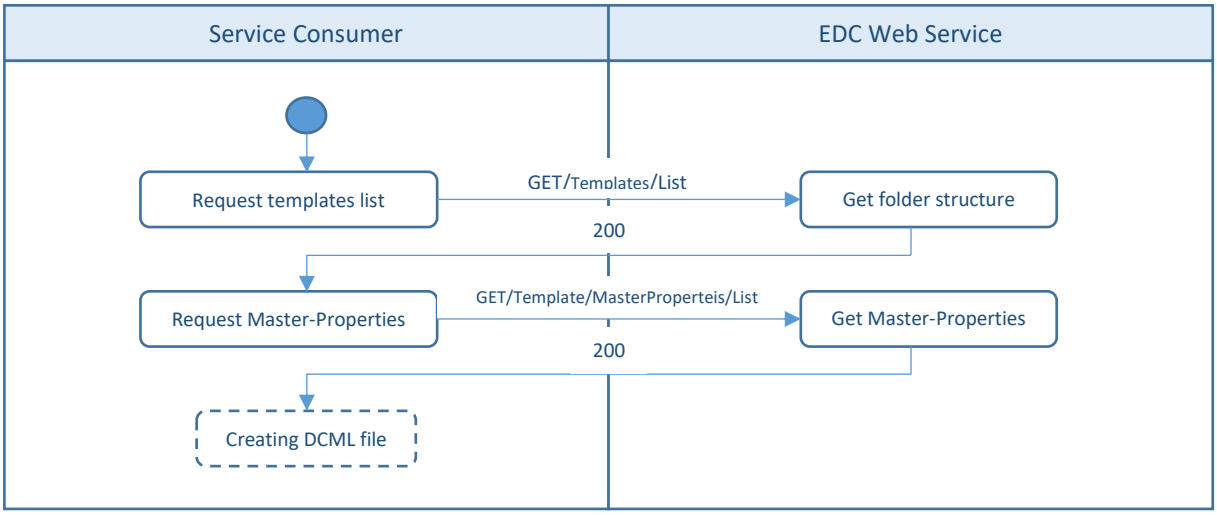
officeatworkWizard.svc			
Type	Method	Parameter	Return Value
GET	/Templates/List	Optional: Language code	Folder structure and templates with id, name and filename
GET	/Template/MasterProperties/List	Template ID Optional: Language code, FieldValues	MasterProperties and their values
POST	/Template/Download	templateFileName	Template filename and template file in binary format
GET	/Templates/List/Cache/Clear	-	-

officeatworkOrders.svc			
Type	Method	Parameter	Return Value
POST	/Orders/New	Order specifications file	Order token
POST	/Orders/{orderToken}/File	Order token, Document file	-
POST	/Orders/{orderToken}/Instructions/New	Order token, Instructions file	Instruction token
GET	/Orders/{orderToken}/Instructions/{instructionToken}/State	Order token, Instruction token	Instruction state & File token(s)
GET	/Orders/{orderToken}/Instructions/{instructionToken}/Report	Order token, Instruction token	Report file name with report file in binary format
GET	/Orders/{orderToken}/File/{fileToken}	Order token, File token	File name with file in binary format
POST	/Orders/{orderToken}/CleanUp	Order token	-

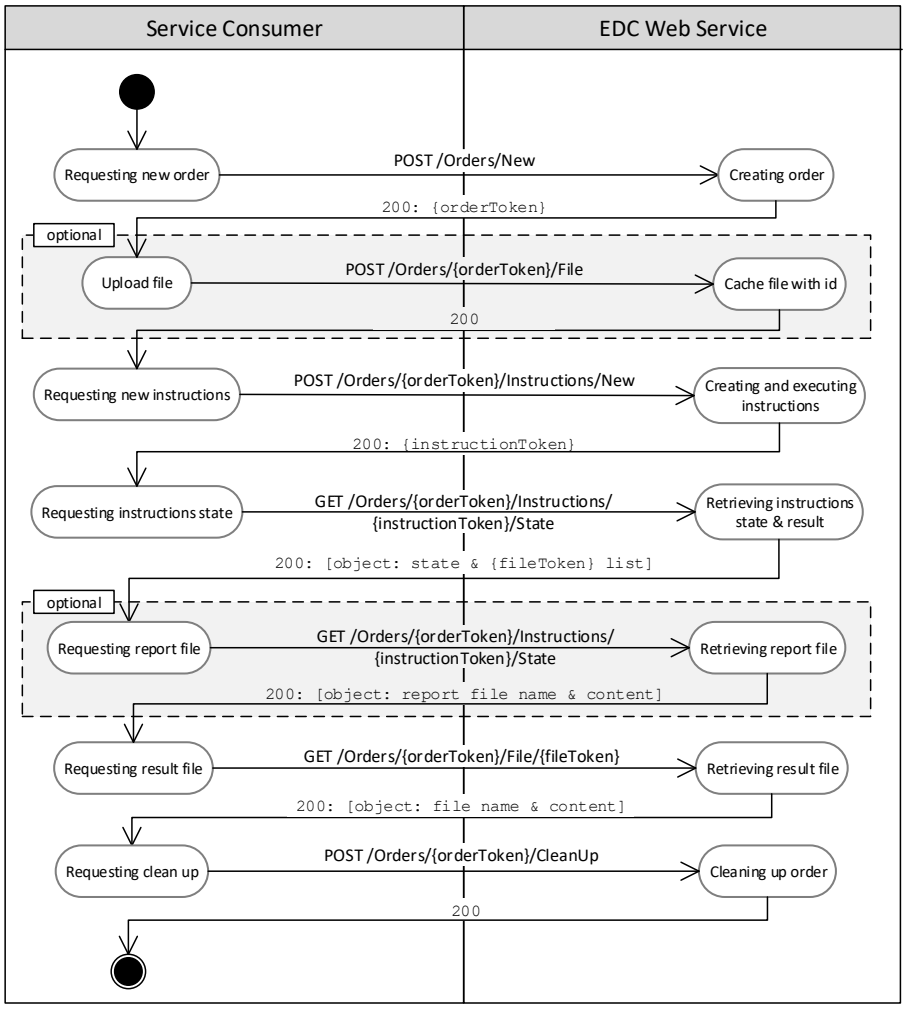
The methods are all RESTful. To verify if a call to one of the methods was successful, one can check the HTTP Status code response. A list of the basic HTTP Status codes is attached in the appendix.

Call sequence

The following fugure shows a typical wizard selection process as a call sequence with each service.



The following figure shows a typical document creation process as a call sequence with each service.



Supported formats

The EDC Server Web Service supports the following formats:

- XML
- JSON

Depending on which format should be used the desired format can be specified directly in the URI of the service call. The call pattern is defined as follows:

```
http://{serviceurl}/EDCServer/{responseformat}/officeatworkOrders.svc/{method}
```

Therefore if the responding format of the EDC Server Web Service should be in XML the service call can be set up as

```
http://{serviceurl}/EDCServer/xml/officeatworkOrders.svc/{method}
```

Otherwise if the responding format should be a JSON object the service call needs to be set up as

```
http://{serviceurl}/EDCServer/json/officeatworkOrders.svc/{method}
```

The URI is case insensitive.

Web Service specification methods

This section provides detailed information for each service method.

Templates

Name

List

HTTP method

Get

URL

http://{serviceurl}/EDCServer/{responseformat}/officeatworkWizard.svc/Templates/List

URL Example

```
http://intern.sample.net/officeatwork/samplenet/samplenet/EDCServer/json/officeatworkWizard.svc/Templates/List
```

Description

Receives a list of the available templates and the folder structure.

Request

Optional «Language» as POST parameter in the Request Body as XML string [Content-Type: text/xml] or as JSON object [Content-Type: text/json].

If no language is defined or the language is not supported, the solution default language will be choosen. If no solution default language is defined, the language «en-us» will be choosen.

XML Example

```
<Templates>
 <Language>en-us</Language>
</Templates>
```

JSON Example

```
{
  "Templates": {
    "Language": "de-ch",
  }
}
```

Response

The available «template list and structure» either as XML string or as JSON object.

XML Example

```
<TemplatesGetResult xmlns:i="http://www.w3.org/2001/XMLSchema-instance">
   <Structure>
      <Folders>
         <Folder>
            <Name>01 Corporate Templates</Name>
            <Templates>
               <Template>
                  <Id>Letter</Id>
                  <Name>Brief</Name>
                  <Filename>Letter.owt</Filename>
               </Template>
               <Template>
                  <Id>Note</Id>
                  <Name>Notiz</Name>
                  <Filename>Note.owt</Filename>
               </Template>
            </Templates>
         </Folder>
         <Folder>
            <Name>02 Sales</Name>
            <Templates>
               <Template>
                  <Id>Offer</Id>
                  <Name>Offerte</Name>
                  <Filename>Offer.owt</Filename>
               </Template>
               <Template>
                  <Id>Invoice</Id>
                  <Name>Rechnung</Name>
                  <Filename>Invoice.owt</Filename>
               </Template>
            </Templates>
            <Folders>
               <Folder>
                  <Name>Presales</Name>
                  <Templates>
                     <Template>
                        <Id>ShortOffer</Id>
                        <Name>Richtofferte</Name>
                        <Filename>ShortOffer.owt</Filename>
                     </Template>
                  </Templates>
               </Folder>
            </Folders>
         </Folder>
      </Folders>
   </Structure>
</TemplatesGetResult>
```

JSON Example

```
{
  "Structure": {
    "Folders": {
      "Folder": [
        {
          "Name": "01 Corporate Templates",
          "Templates": {
            "Template": [
              {
                "Id": "Letter",
                "Name": "Brief",
                "Filename": "Letter.owt"
              },
              {
                "Id": "Note",
                "Name": "Notiz",
                "Filename": "Note.owt"
              }
            ]
          }
        },
        {
          "Name": "02 Sales",
          "Templates": {
            "Template": [
              {
                "Id": "Offer",
                "Name": "Offerte",
                "Filename": "Offer.owt"
              },
              {
                "Id": "Invoice",
                "Name": "Rechnung",
                "Filename": "Invoice.owt"
              }
            ]
```

```
          },
          "Folders": {
            "Folder": {
              "Name": "Presales",
              "Templates": {
                "Template": {
                  "Id": "ShortOffer",
                  "Name": "Richtofferte",
                  "Filename": "ShortOffer.owt"
                }
              }
            }
          }
        }
      ]
    }
  }
}
```


Status codes

200: OK
400: General error
500: Internal server error
501: Provided request file format not supported

Name

List/Cache/Clear

HTTP method

Post

URL

http://{serviceurl}/EDCServer/{responseformat}/officeatworkWizard.svc/Templates/List/Cache/Clear

URL Example

```
http://intern.sample.net/officeatwork/samplenet/samplenet/EDCServer/json/officeatworkWizard.svc/Templates/List/Cache/Clear
```

Description

Cleares and recreats cache of the template list method

Request

None

Response

None

Status codes

200: OK
400: General error
500: Internal server error
501: Provided request file format not supported

MasterProperties

Name

List

HTTP method

Get

URL

http://{serviceurl}/EDCServer/{responseformat}/officeatworkWizard.svc/Template/MasterProperties/List

URL Example

```
http://intern.sample.net/officeatwork/samplenet/samplenet/EDCServer/json/officeatworkWizard.svc/Template/MasterProperties/List
```

Description

Receives a list of the available «Master-Properties» and values.

Request

«TemplateId», optional «Language», optional «FieldValues» as POST parameter in Request Body as XML string [Content-Type: text/xml] or as JSON object [Content-Type: text/json].

If no «Language» is defined or the language is not supported, the solution default language will be choosen. If no solution default language is defined, the language «en-us» will be choosen.

If no «FieldValues» is defined or empty or unknown for each data record will be returned the «Id» and «IDName».

For the Master-Property «CustomFields» only the structure but no data will be returend.

XML Example

```
<Template>
 <Id>Letter</Id>
 <Language>en-us</Language>
 <FieldValues>Default (Empty), None, All</FieldValues>
</Template>
```

JSON Example

```
{
  "Template": {
    "Id": "Note",
    "Language": "fr-fr",
    "FieldValues": "None"
  }
}
```

Response

A list of the available Master-Properties and the values either as XML string or as JSON object.

XML Example

```
<MasterPropertiesGetResult xmlns:i="http://www.w3.org/2001/XMLSchema-instance">
   <MasterProperties>
      <MasterProeprty>
         <Id>Author</Id>
         <Name>Autor</Name>
         <SourceId>Employee</SourceId>
         <Fields>
            <Field>
               <Id>Id</Id>
               <Name>Identifikation</Name>
```

```
        </Field>
        <Field>
            <Id>IDName</Id>
            <Name>Eindeutige Bezeichnung</Name>
        </Field>
        <Field>
            <Id>LastName</Id>
            <Name>Nachname</Name>
        </Field>
        <Field>
            <Id>FirstName</Id>
            <Name>Vorname</Name>
        </Field>
        <Field>
            <Id>EMail</Id>
            <Name>Email-Adresse</Name>
        </Field>
    </Fields>
</MasterProeprty>
<MasterProeprty>
    <Id>ContactPerson</Id>
    <Name>Kontaktperson</Name>
    <SourceId>Employee</SourceId>
    <Fields>
        <Field>
            <Id>Id</Id>
            <Name>Identifikation</Name>
        </Field>
        <Field>
            <Id>IDName</Id>
            <Name>Eindeutige Bezeichnung</Name>
        </Field>
        <Field>
            <Id>LastName</Id>
            <Name>Nachname</Name>
        </Field>
        <Field>
            <Id>FirstName</Id>
            <Name>Vorname</Name>
        </Field>
        <Field>
            <Id>EMail</Id>
            <Name>Email-Adresse</Name>
        </Field>
    </Fields>
</MasterProeprty>
<MasterProeprty>
    <Id>Organisation</Id>
    <Name>Abteilung</Name>
    <SourceId>Organisation</SourceId>
    <Fields>
        <Field>
            <Id>Id</Id>
            <Name>Identifikation</Name>
        </Field>
        <Field>
            <Id>IDName</Id>
            <Name>Eindeutige Bezeichnung</Name>
        </Field>
        <Field>
            <Id>Phone</Id>
            <Name>Telefonnummer</Name>
        </Field>
        <Field>
            <Id>Fax</Id>
            <Name>Faxnummer</Name>
        </Field>
        <Field>
            <Id>Address</Id>
            <Name>Adresse</Name>
        </Field>
    </Fields>
</MasterProeprty>
<MasterProeprty>
    <Id>CustomFields</Id>
    <Name>Spezial Felder</Name>
    <SourceId/>
    <Fields>
        <Field>
            <Id>Date</Id>
            <Name>Erstellungsdatum</Name>
        </Field>
        <Field>
            <Id>Classification</Id>
            <Name>Dokumenten-Klassifizierung</Name>
        </Field>
        <Field>
            <Id>Enclosures</Id>
            <Name>Beilage</Name>
        </Field>
```

```
         </Fields>
      </MasterProeprty>
   </MasterProperties>
   <Data>
      <Data>
         <Id>Employee</Id>
         <Values>
            <Value>
               <Id>1</Id>
               <IDName>Jean Clark, Management, Geschäftsführer</IDName>
            </Value>
            <Value>
               <Id>2</Id>
               <IDName>Melissa Sadiku, Personalabteilung, HR Spezialistin</IDName>
            </Value>
         </Values>
      </Data>
      <Data>
         <Id>Organisation</Id>
         <Values>
            <Value>
               <Id>1</Id>
               <IDName>Personalabteilung</IDName>
            </Value>
            <Value>
               <Id>2</Id>
               <IDName>Finanzen</IDName>
            </Value>
         </Values>
      </Data>
   </Data>
</MasterPropertiesGetResult>
```

JSON Example


```
{
  "MasterProperties": [
    {
      "Id": "Recipient",
      "Name": "Empfänger",
      "SourceId": "",
      "Fields": [
        {
          "Id": "IDName",
          "Name": "Anzeige Name"
        },
        {
          "Id": "DeliveryOption",
          "Name": "DeliveryOption"
        },
        {
          "Id": "Company",
          "Name": "Company"
        },
        {
          "Id": "Department",
          "Name": "Department"
        },
        {
          "Id": "CompleteAddressImported",
          "Name": "Complete Address Imported"
        }
      ]
    },
    {
      "Id": "CustomField",
      "Name": "Zusatzinformationen",
      "SourceId": "",
      "Fields": [
        {
          "Id": "Classification",
          "Name": "Klassifizierung"
        },
        {
          "Id": "DocumentDate",
          "Name": "Datum"
        },
        {
          "Id": "DocumentType",
          "Name": "Dokument Typ"
        }
      ]
    },
    {
      "Id": "Organisation",
      "Name": "Organisation",
      "SourceId": "Organisation",
      "Fields": [
        {
          "Id": "ID",
          "Name": "ID"
        },
        {
          "Id": "UID",
          "Name": "UID"
        },
        {
          "Id": "IDName",
          "Name": "Such-Name"
        },
        {
          "Id": "Organisation",
          "Name": "Organisation"
        },
        {
          "Id": "AdressSingleLine",
          "Name": "Adresse einzeilig"
        }
      ]
    },
    {
      "Id": "Contactperson",
      "Name": "Kontaktperson",
      "SourceId": "Employee",
      "Fields": [
        {
          "Id": "ID",
          "Name": "ID"
        },
        {
          "Id": "UID",
          "Name": "UID"
        },
```

```
      {
        "Id": "IDName",
        "Name": "Such-Name"
      },
      {
        "Id": "Name",
        "Name": "Name"
      },
      {
        "Id": "PersonalNumber",
        "Name": "Personal-Nr."
      }
    ]
  },
  {
    "Id": "Signature1",
    "Name": "Unterschrift 1",
    "SourceId": "Employee",
    "Fields": [
      {
        "Id": "ID",
        "Name": "ID"
      },
      {
        "Id": "UID",
        "Name": "UID"
      },
      {
        "Id": "IDName",
        "Name": "Such-Name"
      },
      {
        "Id": "Name",
        "Name": "Name"
      },
      {
        "Id": "PersonalNumber",
        "Name": "Personal-Nr."
      }
    ]
  },
  {
    "Id": "Signature2",
    "Name": "Unterschrift 2",
    "SourceId": "Employee",
    "Fields": [
      {
        "Id": "ID",
        "Name": "ID"
      },
      {
        "Id": "UID",
        "Name": "UID"
      },
      {
        "Id": "IDName",
        "Name": "Such-Name"
      },
      {
        "Id": "Name",
        "Name": "Name"
      },
      {
        "Id": "PersonalNumber",
        "Name": "Personal-Nr."
      }
    ]
  },
  {
    "Id": "Author",
    "Name": "Autor",
    "SourceId": "Employee",
    "Fields": [
      {
        "Id": "ID",
        "Name": "ID"
      },
      {
        "Id": "UID",
        "Name": "UID"
      },
      {
        "Id": "IDName",
        "Name": "Such-Name"
      },
      {
        "Id": "Name",
        "Name": "Name"
      },
      {
```

```
          "Id": "PersonalNumber",
          "Name": "Personal-Nr."
        }
      ]
    }
  ]
}
```

Status codes

200: OK
400: General error
404: Template does not exists
500: Internal server error
501: Provided request file format not supported

New Order

Name

New

HTTP method

POST

URL

http://{serviceurl}/EDCServer/{responseformat}/officeatworkOrders.svc/Orders/New

URL Example

```
http://intern.sample.net/officeatwork/samplenet/samplenet/EDCServer/json/officeatworkOrders.svc/Orders/New
```

Description

Receives the «order specification» as POST parameter in the Request Body. It creates a new «order» and checks the main server settings. If successful the «order token» is returned to the service user.

Request

«instructions format» as POST parameter in Request Body as XML string [Content-Type: text/xml] or as JSON object [Content-Type: text/json].

XML Example

```
<orderSpecification>
 <instructionsFormat>dcml</instructionsFormat>
</orderSpecification>
```

JSON Example

```
{"instructionsFormat":"dcml"}
```

Response

The «order token» either as XML string or as JSON object.

XML Example

```
<ordersNewResult xmlns:i="http://www.w3.org/2001/XMLSchema-instance">
 <orderToken>de74e21b-6afb-4cc0-b896-0fdc49b89e1b</orderToken>
</ordersNewResult>
```

JSON Example

```
{"orderToken":"de74e21b-6afb-4cc0-b896-0fdc49b89e1b"}
```

Status codes

200: OK
400: General error
500: Internal server error
501: Provided instruction file format not supported

File

Name

File

HTTP method

POST

URL

http://{serviceurl}/EDCServer/{responseformat}/officeatworkOrders.svc/Orders /{orderToken}/File

URL Example

```
http://intern.sample.net/officeatwork/samplenet/samplenet/EDCServer/xml/officeatworkOrders.svc/Orders/de74e21b-6afb-4cc0-b896-0fdc49b89e1b/File
```

Description

This service is optional and can be used to upload an existing document that can be edited using a corresponding edit instruction. Receives the file `id` and the document file `content` as POST parameter in the Request Body. Puts the whole document file `content` in an internal queue.

Request

orderToken	Required. Order token is part of the URI
Content-Type	Required. text/xml or text/json
id	Required. Unique id
content	Required. Depends ont the

This id is the same as in your DCML for the <DocumentID> element.

Response

There is no response value.

XML Example

```
<file xmlns:i="http://www.w3.org/2001/XMLSchema-instance">
    <id>sample.docx</id>
    <content>G4J45XQghgoPP8+BXYRtHOfN245Hc=...</content>
</file>
```

JSON Example

```
{"id":"sample.docx","content":[23,12,03,28,12,42,21,…,123]}
```

Status codes

200: OK
400: General error (hint: check Content-Type)
404: Order token invalid
500: Internal server error

New Instruction

Name

New

HTTP method

POST

URL

http://{serviceurl}/EDCServer/{responseformat}/officeatworkOrders.svc/Orders
/{orderToken}/Instructions/New

URL Example

```
http://intern.sample.net/samplenet/samplenet/officeatwork/EDCServer/xml/officeatworkOrders.svc/Orders/de74e
21b-6afb-4cc0-b896-0fdc49b89e1b/Instructions/New
```

Description

Receives «instructions file» as POST parameter in the Request Body. Creates a new «instruction token» and puts the whole «instructions file» content in an internal queue. If successful the «instruction token» is returned to the service user.

Request

Order token in the URI.

String according to officeatwork DCML API in the Request Body content. The request body content type must be defined as [Content-Type: text/plain].

DCML request example

```
<DCML>
    <Instruction>
        <CreateDocument>
            <Language>2055</Language>
            <TemplateID>letter</TemplateID>
            <Output>
                <Save Filename="Letter for customer.docx"
                      Fileformat="DOCX"
                      DeliveryMethod="Download"
                />
            </Output>
        </CreateDocument>
    </Instruction>
</DCML>
```

Response

The «instruction token» in the defined response format.

XML Example

```
<instructionsNewResult xmlns:i="http://www.w3.org/2001/XMLSchema-instance">
   <instructionToken>
      a0b62d55-de34-4168-9256-d9bff294eb3c
   </instructionToken>
</instructionsNewResult>
```

JSON Example

```
{"instructionToken":"a0b62d55-de34-4168-9256-d9bff294eb3c"}
```

Status codes

200: OK
400: General error (hint: check Content-Type: text/plain)
404: Order token invalid
500: Internal server error

Instruction state

Name

State

HTTP method

GET

URL

http://{serviceurl}EDCServer/{responseformat}/officeatworkOrders.svc/Orders/{orderToken}/Instructions/{instructionToken}/State

URL Example

```
http://intern.sample.net/officeatwork/samplenet/samplenet/EDCServer/xml/officeatworkOrders.svc/Orders/de74e21b-6afb-4cc0-b896-0fdc49b89e1b/Instructions/a0b62d55-de34-4168-9256-d9bff294eb3c/State
```

Description

Checks whether the execution of a previous uploaded «instructions file» has finished. Returns the process state and a token list of the result files. Possible states are:

- processing
- finished

Reqeust

«Order token» and «instruction token» in the URI.

Response

The execution progress containing the process state. If the instruction passed to the server orders files for download, the corresponding file token list is also returned.

XML Example

```
<instructionState xmlns:i="http://www.w3.org/2001/XMLSchema-instance">
   <state>processing</state>
   <fileTokens/>
</instructionState>
```

Or

```
<instructionState xmlns:i="http://www.w3.org/2001/XMLSchema-instance">
   <state>processing</state>
   <fileTokens>
      <fileToken>a9ca69a4-ec49-451d-a245-0464df2677cd</fileToken>
   </fileTokens>
</instructionState>
```

Or

```
<instructionState xmlns:i="http://www.w3.org/2001/XMLSchema-instance">
   <state>finished</state>
   <fileTokens>
      <fileToken>a9ca69a4-ec49-451d-a245-0464df2677cd</fileToken>
      <fileToken>97d66812-87b9-4b54-a1ae-270e0cf8457e</fileToken>
   </fileTokens>
</instructionState>
```

JSON Examples

```
{"state":"processing","fileTokens":[]}
```

Or

{"state":"finished","fileTokens":["a9ca69a4-ec49-451d-a245-0464df2677cd","97d66812-87b9-4b54-a1ae-270e0cf8457e"]}

Status codes

200: OK
400: General error
404: Order or instruction token invalid
500: Internal server error

Download file

Name

Download

HTTP method

GET

URL

http://{serviceurl}/EDCServer/{responseformat}/officeatworkOrders.svc/Orders/{orderToken}/File/{fileToken}

URL Example

```
http://intern.sample.net/officeatwork/samplenet/samplenet/EDCServer/json/officeatworkOrders.svc/Orders/de74e21b-6afb-4cc0-b896-0fdc49b89e1b/File/a9ca69a4-ec49-451d-a245-0464df2677cd
```

Description

Downloads a file according to the associate file token.

Request

«Order token» and «file token» in the URI.

Response

The file name and the binary file content. In the XML response the file content is a Base64 encoded string. In the JSON response the content is an ordinary byte array.

XML Example

```
<file xmlns:i="http://www.w3.org/2001/XMLSchema-instance">
    <name>sample.docx</name>
    <content>G4J/XQghgoPP8+BXYRtHOfN2/Hc=...</content>
</file>
```

JSON Example

```
{"name":"sample.docx","content":[23,12,03,28,12,42,21,…,123]}
```

Status codes

200: OK
400: General error
404: Order or file token invalid
500: Internal server error

Download Report

Name

Report

HTTP method

GET

URL

http://{serviceurl}/EDCServer/{responseformat}/officeatworkOrders.svc/Orders/{orderToken}/instructions/{instructionToken}/Report

URL Example

```
http://intern.sample.net/officeatwork/samplenet/samplenet/EDCServer/json/officeatworkOrders.svc/Orders/de74e21b-6afb-4cc0-b896-0fdc49b89e1b/instructions/a9ca69a4-ec49-451d-a245-0464df2677cd/Report
```

Description

Downloads a report file according to the associate instruction token.

Reports are only available for orders in dcml format.

Request

«Order token» and «instruction token» in the URI.

Response

The report file name and the binary file content. In the XML response the file content is a Base64 encoded string. In the JSON response the content is an ordinary byte array.

XML Example

```
<file xmlns:i="http://www.w3.org/2001/XMLSchema-instance">
    <name>processing.report</name>
    <content>G4J/XQghgoPP8+BXYRtHOfN2/Hc=...</content>
</file>
```

JSON Example

```
{"name":"processing.report","content":[23,12,03,28,12,42,21,…,123]}
```

Status codes

200: OK
400: General error
404: Order or instruction token invalid or the report creation is not initialized
500: Internal server error

Download Template

Name

Download

HTTP method

POST

URL

http://{serviceurl}/EDCServer/{responseformat}/officeatworkWizard.svc/Template/Download

URL Example

```
http://intern.sample.net/officeatwork/samplenet/samplenet/EDCServer/json/officeatworkWizard.svc/Template/MasterProperties/List
```

Description

Downloads a template file according to the associate filename.

Request

«Filename» as POST parameter in Request Body as XML string [Content-Type: text/xml] or as JSON object [Content-Type: text/json].

XML Example

```
<templateDownload xmlns:i="http://www.w3.org/2001/XMLSchema-instance">
  <templateFileName>Blank.owt</templateFileName>
</templateDownload>
```

JSON Example

```
{
  "templateFileName": "Blank.owt"
}
```

Response

The template file name and the binary file content. In the XML response the file content is a Base64 encoded string. In the JSON response the content is an ordinary byte array.

XML Example

```
<file xmlns:i="http://www.w3.org/2001/XMLSchema-instance">
    <name>Blank.owt</name>
    <content>G4J/XQghgoPP8+BXYRtHOfN2/Hc=...</content>
</file>
```

JSON Example

```
{"name":"Blank.owt","content":[23,12,03,28,12,42,21,…,123]}
```

Status codes

200: OK
400: General error
404: Order or instruction token invalid or the report creation is not initialized
500: Internal server error

Clean up

Name

CleanUp

HTTP method

POST

URL

http://{serviceurl}/EDCServer/{responseformat}/
officeatworkOrders.svc/Orders/{orderToken}/CleanUp

URL Example

```
http://intern.sample.net/officeatwork/samplenet/samplenet/EDCServer/json/officeatworkOrders.svc/Orders/de74e21b-6afb-4cc0-b896-0fdc49b89e1b/CleanUp
```

Description

Performs the cleanup task for an existing order.

Request

«Order token» in the URI.

Response

None

Status codes

200: OK
202: Order does not exists or has already been cleaned up
400: General error
500: Internal server error

Appendix

Examples

The following example shows you an example how to process an order in xml format with C#.

```
public class Simple_Order_Processing_In_XML
{
  static void Main(string[] args)
  {
    // step 1: request new order
    const string ServerXmlPrefix =
    "http://intern.sample.net/officeatwork/EDCServer/xml/officeatworkOrders.svc/Orders/";
    const string CreateOrderRequestUri = ServerXmlPrefix + "New";
    const string CreateOrderRequest =
    @"<orderSpecification>
        <instructionsFormat>dcml</instructionsFormat>
      </orderSpecification>";

    XDocument createOrderResponse = PostData(CreateOrderRequestUri, CreateOrderRequest, "text/xml");
    string orderToken = createOrderResponse.XPathSelectElement("/ordersNewResult/orderToken").Value;

    // step 2: request new instruction
    string createInstructionRequestUri = ServerXmlPrefix + orderToken + "/Instructions/New";
    const string InstructionFileContent =
    @"<DCML>
        <Instruction>
            <CreateDocument> ... </CreateDocument>
        </Instruction>
      </DCML>";

    XDocument createInstructionResponse =
    PostData(createInstructionRequestUri, InstructionFileContent, "text/plain");
    string instructionToken = createInstructionResponse
    .XPathSelectElement("/instructionsNewResult/instructionToken").Value;

    // step 3: request instruction state
    string instructionStateRequestUri =
    ServerXmlPrefix + orderToken + "/Instructions/" + instructionToken + "/State";

    XDocument instructionState;
    string processingState;

    do
    {
      instructionState = GetData(instructionStateRequestUri);
      processingState = instructionState.XPathSelectElement("/instructionState/state").Value;
    }
    while (processingState == "processing");

    // step 4: download document
    IEnumerable<string> fileTokens = instructionState
    .XPathSelectElements("/instructionState/fileTokens/fileToken")
    .Select(x => x.Value);

    foreach (string fileToken in fileTokens)
    {
      string downloadFileRequestUri = ServerXmlPrefix + orderToken + "/File/" + fileToken;
      XDocument fileResponse = GetData(downloadFileRequestUri);

      string fileName = fileResponse.XPathSelectElement("/file/name").Value;
      byte[] fileContent =
     Convert.FromBase64String(fileResponse.XPathSelectElement("/file/content").Value);

      File.WriteAllBytes(@"c:\temp\" + fileName, fileContent);
    }

    // step 5: clean up
    string cleanUpRequestUri = ServerXmlPrefix + orderToken + "/CleanUp";
    PostData(cleanUpRequestUri, string.Empty, string.Empty);
  }
```

```
  static XDocument PostData(string requestLocation, string textObjectToPost, string contentType)
  {
    byte[] dataToPost = Encoding.UTF8.GetBytes(textObjectToPost);

    var request = (HttpWebRequest)WebRequest.Create(requestLocation);
    request.Method = "POST";
    request.ContentLength = dataToPost.Length;
    request.ContentType = contentType;

    using (Stream postStream = request.GetRequestStream())
    {
      postStream.Write(dataToPost, 0, dataToPost.Length);
    }

    if (string.IsNullOrEmpty(contentType))
    {
      return null;
    }

    using (var response = (HttpWebResponse)request.GetResponse())
    {
      return XDocument.Load(response.GetResponseStream());
    }
  }

  static XDocument GetData(string requestLocation)
  {
    var client = new WebClient();

    using (var stream = new MemoryStream(client.DownloadData(requestLocation)))
    {
      return XDocument.Load(new StreamReader(stream));
    }
  }

}
```

HTTP Status codes

The following table gives an overview of some basic HTTP Result codes and their meaning.

Status code	Meaning	Description
Success 2xx		
200	OK	The request was fulfilled.
201	Created	Following a POST command, this indicates success.
202	Accepted	The request has been accepted for processing, but the processing has not been completed.
204	No Response	Server has received the request but there is no information to send back, and the client should stay in the same document view.
Redirection 3xx		
301	Moved	The data requested has been assigned a new URI, the change is permanent.
Error 4xx – Client side		
400	Bad Request	The request had bad syntax or was inherently impossible to be satisfied.
401	Unauthorized	The parameter to this message gives a specification of authorization schemes which are acceptable.
403	Forbidden	The request is for something forbidden. Authorization will not help.
404	Not found	The server has not found anything matching the URI given.
Error 5xx – Server side		
500	Internal Error	The server encountered an unexpected condition which prevented it from fulfilling the request.
501	Not implemented	The server does not support the facility required.

CHAPTER 2

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

officeatwork AG
Bundesplatz 12
6300 Zug, Switzerland

T +41 41 544 7100

www.officeatwork.com
mail@officeatwork.com