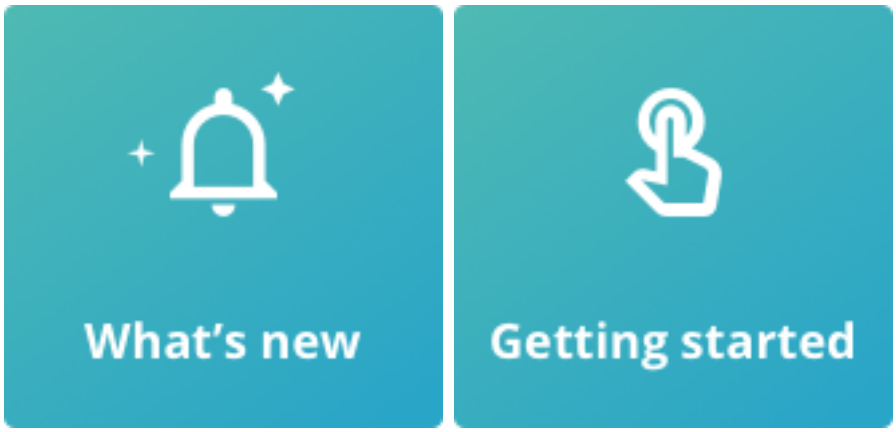

2021.2

# Developer Guide

Contents

# Developer Guide for WorkZone for Office 2021.2




- View PDF version

## Most used topics

- Customize view and dialog boxes
- Install customizations
- Create macro-enabled templates

## Useful links

- XAML Syntax In Detail
- XAML Overview (WPF)

## Related product documentation

- WorkZone for Office User Guide
- Installation Guide for WorkZone

## WorkZone links

- WorkZone documentation
- WorkZone support
- WorkZone website
- WorkZone portal

# What's new

## 2021.2

No changes in this release.

### 2021.1

No changes in this release.

### 2021.0

No changes in this release.

### 2020.3

- The `AccessToken` and `UseOAuth` parameters have been added to the `BuilderParameters` to support the OAuth2 authentication. If you get `AuthenticationException` when using the `BuildAsync` method in the OAuth2 mode, it means that the current access token has expired and another one should be obtained. See Merge API.
- `Ready`, `startDocument`, and `exportToExcel` methods are no longer used as the available API methods. The new `initialize` method has been introduced. The `openDocument` method has been replaced by the `openEmail` one. See Available API methods.

### 2020.2

No changes in this release.

### 2020.1

No changes in this release.

### 2020.0

No changes in this release.

### 2019.3

New macro interface members have been introduced:

- `Before_Merge`
- `After_Merge`
- `WorkZone_BeforeMerge`
- `WorkZone_AfterMerge`

Use them to notify custom methods when the merging starts and ends, and, optionally, to cancel the merging. See Macro interface members.

### 2019.2

No changes in this release.

### 2019.1

- The **ICustomXmlPartBuilder** interface has been extended with another `BuildAsync` method. The new method contains `caseID` as a parameter and allows to retrieve case ID directly from the method. See Merge API.

### 2019.0

No changes in this release.

### 2018.2

No changes in this release.

### 2018.1

No changes in this release.

### 2018

- Merge API methods have been added and described. Now you can customize content controls and merge them with information by using public API.
- New settings have been added to configure content controls. The settings for the merge are the following: <linkTemplate> with attributes `href` and `queryForLocalData`; `internal` in <property>; `addressContext` in <staticData> and <link>. The settings for repeating section are the following: `repeatable` in <dynamicMenu>, `group` in <staticData> and `additionalKey` in <dynamicData>.
- Number of run-time dependencies has been decreased from 6 to 4.

### 2017

- REST API methods have been added and described. Use them to integrate your system with the WorkZone for Office functionality.
- You can customize content controls for your WorkZone for Office installation. To do this, you need to learn the structure of the XML files and follow the example.
- You can make the Role field on the Document registration pane required.

### 2016 R2

- The standard value set for WorkZone for Office Server has been updated. See the Standard value set in WorkZone Office Server installer.

# Get started

The forms of WorkZone for Office can be customized in several ways. You can bind controls to WorkZone Content Server fields or to the properties in WorkZone Content Server items.

Basic knowledge of XAML and of the WorkZone Content Server data model are the pre-requisites for performing customization.

The following sections introduce you to the customization of WorkZone for Office:

- An overview of components used for customization.
- The location of files and how to add and remove fields.
- Instructions on how to install customized XAML and load ColumnSetDefinitions.

# Customizing WorkZone for Office

## Component overview

The registration panes and dialog boxes in WorkZone for Office are implemented using XAML.

A dialog box or a registration pane is defined by two types of components:

- The **XAML** file that describes the layout of the form and data bindings for each field.
- The **ColumnSetDefinitions** file that contains all data presented in the form.

To facilitate the binding between the XAML controls and the data columns, a set of 'Sj' styles exists. The SjStyles include styles for most WorkZone Content Server field types. Furthermore, the SjStyles set up bindings between the attributes of the XAML control and data dictionary attributes for the corresponding data column.

> Important: XAML and SjStyles are carefully constructed to provide the desired functionality and appearance. So, even though it is technically possible to replace the XAML completely, KMD recommends that you use the existing XAML and SjStyles to the greatest extent possible.

## About SJStyles

The simplest way to work with standard bindings is to use styles. SJStyles are used to customize XAML controls. There are defined styles for the most common control types, but you can also define your own style, if needed. The predefined controls start with "Sj".

"SJ" controls describe binding rules for the most commonly used controls. These are:

- `SjTextBox`
- `SjComboBox`
- `SjTextBlock`

- `SjLabel`
- `SjDatePicker`.

For the search forms, the following styles have been defined:

- `SjTextBoxSearch`
- `SjComboBoxSearch`
- `SjAutoCompleteControl` (also used in `DocumentRegistrationPane`).

The major difference between them is that the search form styles do not validate the input.

---

## Generalized SJStyles

The generalized SjStyles listed below are presented only in XAML forms. The SjStyles are not available in XAML dialog boxes. Changing these styles in dialog boxes is possible only by inheritance. However, you can add new controls with these styles.

- `SjOfficeStyleTextBox` - a text box that inherits the Microsoft Office color scheme and has no binding. It is used as a basis for `SjTextBox`. It can be used directly if no default binding is needed.
- `SjTextBox` - a text field. Attribute values are inherited from the data column.
- `SjTextBoxSearch` - is used on the search forms for regular text input. Does not validate the input.
- `SjComboBox` - is used on the search forms for regular text input. Does not validate the input.
- `SjComboBoxSearch`, `SjFilterComboBox` - a drop down field. Attributes values inherited from the data column.
- `SjGridComboBox` - is used to select items inside the grid control.
- `SjDatePicker` - a date field with the date picker control attached. Attributes values inherited from the data column.

  You can configure date picker content control to use the long or short date format, so that it uses the same format as the SjDatePicker. See Configurable elements.

- `SjCaseClassAutoCompleteBox` - is used for case class auto-complete text box.

- `SjAutoCompleteControl` - is used on the search forms and in the **Document Registration** pane to enable auto-complete behavior for Case Handler (`officer`) and Responsible Unit (`responsible_ou`) fields.

## Customizable SJStyles

Customizable SjStyles are presented in each XAML file. You can configure them according to your needs.

- `SjLabel` - static text, used primarily for labels.
- `SjTextBlock` - a read-only text field.

### Properties for SjAutoCompleteControl

- `MinimumSearchLengthByCode` - defines the minimum number of typed characters needed to start the search. If this property is used, search is performed only by initials. The default value is 2.

  > **Important:** This value should be lower than the value you assign to `MinimumSearchLength`.

- `MinimumSearchLength` - defines the minimum number of typed characters needed to start the search. If this property is used, search is performed on any text type (that is, in full text and initials). The default value is 3.
- `LoadAllDataCommand` - displays all items that match the typed text.

  > **Important:** This property must not be used together with `LoadNextDataCommand`.

- `LoadNextDataCommand` - displays the first 50 items thatwhich match the typed text. If there are more than 50 items, a scrollbar appears. If you scroll down, additional items are displayed.

- `ResolveItemValueAsPrimary` - the value can be `True` or `False`.
    - `True`: when the drop-down list is closed, the item containing initials that match the typed text is selected automatically.
    - `False`: when only one item is found, this item is selected automatically. When several items are found, the selected item is then resolved as empty (that is, the control is cleared).

## Customize view and dialog boxes

The **View** settings of the registration pane and various dialog boxes in WorkZone for Office are stored in the database as an XAML configuration.

---

### View and change settings for dialog boxes and views

To see the views available in the database and (or) change your selected settings, you need to execute the following query in SCANSQL or SQLPLUS:

```
SELECT*
FROM services_configuration
WHERE Module_name = 'Scanjour.Services.Office'
```

Below is the full list of customizable dialog boxes and views in WorkZone for Office:

- AddressSelectDialog
- CaseConfirmDialog
- CaseSearchDialog
- CreateCaseDialog
- DocumentRegistrationPane
- OutlookItemRegistrationDialog
- MultipleSavingCommonMetadataDialog
- FacetsSelectDialog
- FileClassSelectDialog
- FixedListDialog

- OutlookRecordSearchDialog
- PartiesSuggestionDialog
- RecipientSelectDialog
- RecordSearchDialog
- RecordSelectDialog

As views can contain language specific texts, separate versions of the views for each of the supported languages are stored in the database. See About SJStyles.

ColumnSetDefinitions is an xml structure that defines which fields are available for the Office Clients. This configuration is merged into the WorkZone Content Server data dictionary.

> **Important:** You need to load the standard ColumnSet configuration used by WorkZone for Office into the `sd_datadict` table manually. See Installation Guide for WorkZone.

## Make a field required or non-required during the registration of the item

You can add values to the dialog boxes and views and make them required or non-required during the item registration. This is relevant for the following dialog boxes and views:

- CreateCaseDialog
- DocumentRegistrationPane
- OutlookItemRegistrationDialog
- MultipleSavingCommonMetadataDialog.

1. Open the relevant XAML file.
2. Find the control with a field value that you want to make required or non-required.
3. To make a field required, you must add the following property to the control: `prop:ControlBehaviour.IsRequired="True".`
   -or-
   To make a field non-required, you must remove its control from the XAML file.
4. Save your changes.

> **Important:** You can make a field required if it is defined as non-required on the server, but you cannot make a field non-required if it is set as required on the server.

The procedure of making the **Role** field on the Document registration pane required differs. You can find an example below:

### How to make the Role field on the Document registration pane required

If you want users to always select a role when they add a new party on a case, you need to complete the following changes in XAML:

1. Remove the following code:

```
<GridViewColumn Width="80" DisplayMemberBinding="{Binding
custom_label.Elab}"
                        prop:RangeColumnBehaviour.MinWidth="
{x:Static
                        consts:LayoutConstants.GridViewColumnMinWid
```

2. Instead, insert the following code:

```
<GridViewColumn  Width="80"
        prop:RangeColumnBehaviour.MinWidth="{x:Static
        consts:LayoutConstants.GridViewColumnMinWidth}">
              <GridViewColumn.CellTemplate>
                    <DataTemplate>
                       <Grid>
                                <ComboBox DataContext="{Binding cus
tom_label}"
                                Style="{DynamicResource ODataComboB
                                IsEnabled="False" prop:Con-
trolBehaviour.IsRequired="True"/>
                       </Grid>
                    </DataTemplate>
              </GridViewColumn.CellTemplate>
 </GridViewColumn>
```

## Configure the "ColumnSetDefinitions" file

To open and edit your ColumnSetDefinitions file, you must execute the following query in SCANSQL or SQLPLUS:

```
SELECT*
FROM sd_datadict>
WHERE name = 'mwrp_columnsetdefinitions'
```

Customized versions of ColumnSetDefinitions and XAML are stored in a local project structure together with the scripts for loading the configuration to database.

## Add a text field to the registration pane

You can add an extra field to the **Registration** pane.

1. Add the data column for the new field to ColumnSetDefinitions (if it is not present already).
2. Add the XAML control for the field using the appropriate SjStyle, and bind it to the relevant data column.

### A text field example

In the example below, the **Record Title** field is defined as a simple text field.

The field name in data dictionary: `record:title`

ColumnSetDefinitions

In the ColumnSetDefinitions file, add the `"title"` column under the register `"record"` and ColumnSet `"Single"`:

```
<Scanjour>
    <Settings>
       <Services>
          <Clients>
            <Client name="OfficeService">
```

```
                <Registers>
                    <Register name="record">
                        <ColumnSets>
                            <ColumnSet name="Single">
                                <Column name="title"/>
                            </ColumnSet>
                        </ColumnSets>
                    </Register>
                </Registers>
            </Client>
        </Clients>
    </Services>
</Settings>
</Scanjour>
```

XAML

In **Registration Pane**.<language>.xaml, you can define a field using the SjTextBox styling and binding to the column Record.title:

```
<TextBox x:Name="Title_field" DataContext="{Binding Record.-
title}"
                Style="{DynamicResource SjTextBox}"
                Margin="0" Width="Auto" Height="20" Tex-
tWrapping="NoWrap"
                VerticalAlignment="Stretch"
HorizontalAlignment="Stretch" TabIndex="0"
                HorizontalAlignment="Stretch" TabIndex="0"/>
```

In the example above, extra layout attributes have been added to the XAML control to specify alignment, tab order, and text wrapping.

## Add other field types

You can add all common ScanJour WorkZone Content Server field types using the pre-defined SjStyles.

The example below shows how to add a drop-down list for a domain controlled field. In the example, 'Record Type' is defined as a domain bound field.

Example

**ColunmSetDef.**

In ColumnSetDefinitions, the field is added like this:

```
<ColumnSet name="Single">
                <Column name="record_type"/>
                </ColumnSet>
```

**XAML**

In XAML, the field can be defined as follows:

```
<ComboBox x:Name="Doctype_combo" DataContext="{Binding Record.re-
cord_type}"
                Style="{DynamicResource SjComboBox}"
                Margin="0" Height="20" TabIndex="1" />
```

## Remove a field from the registration pane

You can remove any field from the standard **Registration** pane. You only need to remove the field from XAML; it is not necessary to remove the field from ColumnSetDefinitions.

> Important: When you decide to remove a field from the **Registration** pane, you must make sure that this field is not defined as a required field in the data dictionary. A required field must always be present for input, unless it has a default value defined in the data dictionary.

## Use add-in customization

To write and use the customized add-ins, the following requirements must be met:

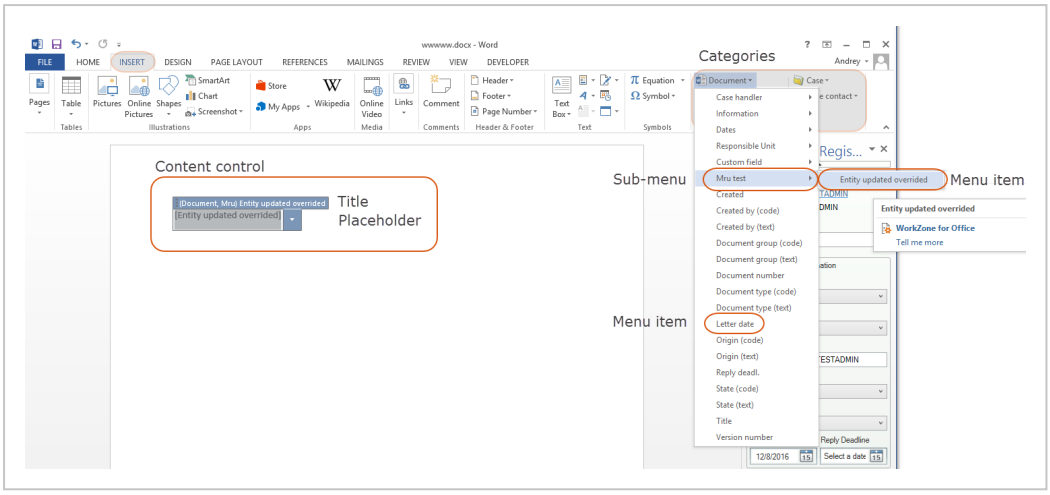
- The AddIn assembly must have a reference to the WorkZone for Office assembly `Scanjour.Office.CustomizationContracts`.
- The AddIn class must be inherited from the `IAddIn` interface.
- The AddIn class must have an `AddIn` attribute.
- All AddIns have the `SetContext(HostContext hostContext)` method. This HostContext has WorkZone for Office methods that are accessible from the AddIn.
- All AddIns have the `GetViewModel` method, which is used to get a view model (that is like a code behind the UI customization on XAML). The access to a view model from XAML uses the `XamlViewModelKey` attribute from the `AddInConfiguration.xml` file.
- You can create an AddIn without a UI (a view model). To do this, you must return null from `'AddIn.GetViewModel()'` and not provide an `XamlViewModelKey` attribute for the specific AddIn in the `AddInConfiguration.xml` file.

## Customize content controls

### About content controls

Content controls help users create templates in Microsoft Word. From a WorkZone point of view, a content control represents a specific property of an entry. An entry is an object in OData that refers to the database. Entries in OData are used to retrieve specific values from the database.

When a user creates a template, the user inserts content controls by selecting menu items from the **Content properties** group on the **Insert** tab in Word. The user then selects a document in the **Registration pane** and clicks **Merge**. WorkZone substitutes each content control with values from the database.




You can create sub-menus and define which menu items to be displayed in each category. Content properties are divided into four categories:

- Document
- Case
- Document contact
- Case contact

## Create a content control

This section describes step by step how to create a new content control called **Entity updated overrided**. The content control represents the **Entity updated** property of the **MruRecords** entry. To complete the creation, you need to create a reference to the database through OData and add a new menu item that will appear in Word.

### Define the content control in OData

1. Open the **merge_odata_description** XML file. To request a specific entry, create a query to OData. Use the following example:

> **Example**:

```
 <staticData id="E34FE056-129F-46D2-B7E7-
A20D91F9F47C" query="MruRecords?$filter=Value eq '
{recordId}' and User_Value eq '{record.officer}'">
```

where:

- `staticData` - use this element to request an entry.
- `E34FE056-129F-46D2-B7E7-A20D91F9F47C` - insert a unique ID that will be used internally as a reference to the OData entry. You may use any unique ID, however, it is recommended to use a GUID generator to generate the ID.
- `query` - an OData request that retrieves the entry that you need. In this example, it is a call to the `MruRecords` entry.

2. Within the `staticData` element, bind a content control with the required property. To do this, you need to add and define the **property** element.

   You may add as many properties as you need.

```
<staticData id="E34FE056-129F-46D2-B7E7-A20D91F9F47C"
query="MruRecords?$filter=Value eq
       '{recordId}' and User_Value eq '{record.officer}'">
       <property id="C5D4237E-2DEF-4D5A-B714-1F5CA88518F5" name-
e="EntityUpdated " />
</staticData>
```

   where:

   - `property` - use this element to request a property.
   - `E34FE056-129F-46D2-B7E7-A20D91F9F47C` - insert a unique ID that will be used internally as a reference to the property of the entry.
   - `name` - specify property's name in OData.

## Add content control to the user interface

3. Add a new menu item to the Word application. You can do this in the **menu_description** XML file. Open the file and find a proper place for the new menu item. The current example represents how to search for the **Document** category.

   Expand to find a proper category for your menu item:

   - `<menu id="Case">` refers to **Case**.
   - `<menu id="Record">` refers to **Document**.
   - `<menu id="CaseParty">` refers to **Case contacts**.
   - `<menu id="RecordParty">` refers to **Document contacts**.

   In the code source, the **Document** category refers to **Record**. Therefore, the new sub-menu must be added under the `<menu id="Record">` element:

```
<menu id="Mru">
       <displayName xml:lang="en-GB">Mru test</displayName>
       <displayName xml:lang="da-DK">Mru test</displayName>
              <dynamicMenu id="Mru_Items">
                     <contentRefs>
                            <ref>E34FE056-129F-46D2-B7E7-A20D91
                     </contentRefs>
              </dynamicMenu>
</menu>;
```

   where:

   - `menu` - use this element to create a sub-menu.
   - `displayName` - define a menu item name to be displayed in Word.
   - `dynamicMenu` - use this element to create new menu items.
   - `ref` - specify reference to the unique ID in the `merge_odata_description` XML file.

4. Optionally, define customized names for the new menu item and content control:

```
<label dataItemRef="C5D4237E-2DEF-4D5A-B714-
1F5CA88518F5">
       <value xml:lang="en-GB">Entity updated overrided</value>
        <value xml:lang="da-DK">Entity updated
```

```
overrided</value>
</label>
```

where:

- `label` - specifies a customized name.
- dataItemRef - specifies a reference to the property of the entry from the merge_odata_description XML file.
- `xml:lang` - is a localization parameter.

5. You can define how the content control must behave in Word when there is no value. You have two options:
    a. Keep the content control with no changes
    b. Hide the content control

For option *a*, specify the unique ID of the entry within the `<Rule rule="Unchanged">` section. For option *b*, specify the unique ID of the entry within the `<Rule rule="Empty">` section.

**Example**:

```
<Rule rule="Unchanged"> <menuDataRefs>
<ref>E34FE056-129F-46D2-B7E7-A20D91F9F47C</ref>
</menuDataRefs> </Rule> <Rule rule="Empty">
<menuDataRefs> <ref>17EDE3E5-43D3-4B88-8E6E-
F336D916B916</ref> </menuDataRefs> </Rule>
```

where:

- `ref` - a reference to the unique ID in the **merge_odata_description** XML file.

Optionally, you can define which information should be included in the content control title.

**Example**:

```
<cc dynamicMenuRef="Mru_Items"> <titleFormat xml:lang-
g="en-GB">(Document, Mru) {displayName}</titleFormat>
 <titleFormat xml:lang="da-DK">(Document, Mru) {dis-
playName}</titleFormat> </cc>
```

where:

- `dynamicMenuRef` - specifies a reference to a menu item.
- `TitleFormat` - defines the content control title for each language.
- `{displayName}` - displays the name of the menu item that a user can click.

## XML files structure

There are two files that define the Merge-related functionality:

1. `merge_odata_description.xml`
2. `menu_description.xml`

If you want to edit an existing content control, you only need to customize the `merge_odata_description.xml` file. To add a completely new content control, you need to customize both files.

You can define the following:

- Content controls and their order in the menus in Word. See About content controls.
- Titles and placeholders of the content controls
- Merge functionality

# merge_odata_description.xml

This file describes data that is used for merging content controls and values from the database. Merge of multiple entities is not supported, so make sure that you only select single items in your queries. To edit the file, you must have knowledge about OData.

## Elements and attributes:

- `property` - establishes mapping `atom:content -> metadata:properties`.
    - `internal` - attribute for internal usage only. You can use it to display or hide a content control in menu on UI. If the value is `false`, then content control is displayed. If the value is `true`, then content control is hided.
- `link` - establishes mapping to `atom:link`. Only `atom:link` with `type=entry` is supported. The structure of `link` is the same as the structure of `staticData`. This element is optional.
    - `query` - relative OData request that contains any allowed OData system query options. The only exceptions are `select` and `expand` options that are built based on the `property` and `link` elements.
    - `addressContext` - defines which rule must be applied to handle protected addresses locally and from Merge API. Available values for `addressContext`are the following: *CaseOfficer*, *CaseResponsibleUnit*, *RecordOfficer*, *RecordResponsibleUnit*, *RecordParty*, *CaseParty*.
- `dynamicData` - builds content control menus with items based on queries.
    - `key` - use together with the `query` attribute to create an OData query. The value of `key` is used as an ID of a menu item.
    - `additionalKey` - use additionally to `key` if OData entities have complex ID.
    - `value` - use together with the `query` attribute to create an OData query. The value of `value` is used as a label of a menu item.
- `staticData` - builds content control menus with items based on static properties. The element reflects the OData atom payload for a specific entity. See also Naming conventions and Atom format.

    - `href` - mapping to `atom:link`.
    - `query` - relative OData request that contains any allowed OData system query options.
    - `group` - groups related data for the repeating content, for example, **Information** or **Dates** in the **Contact** menu.
    - `addressContext` - defines which rule must be applied to handle protected addresses locally and from Merge API. Available values for `addressContext`are the following: *CaseOfficer*, *CaseResponsibleUnit*, *RecordOfficer*, *RecordResponsibleUnit*, *RecordParty*, *CaseParty*.
- `linkTemplate` - child element of the `staticData` element and also reflects the OData atom payload for a specific entity. This element is only used for ID 81E576F0-FFAA-4229-8B46-2BEC4972B474 (Record staticData) and has an additional attribute.
    - `href` - mapping to `atom:link`.
    - `queryForLocalData` - describes OData query based on 'local' Record data. The attribute refers to run-time dependency {record.<SOM field name>}.

## Run-time dependencies

During the merge, all run-time dependencies will be replaced with the correspondent values. All dependencies are wrapped in '{}'. There are 4 types of dependencies:

1. `{recordId}` - ID of the document that a user merges.
2. `{addressId}` - ID of an address that belongs to the corresponding case or document party.
3. `{guid}` - reference to a `dynamicData` ID or `property` ID in `merge_odata_description.xml`.
4. `{record.<SOM field name>}` - where '<SOM field name>' is a direct document field name. This dependency was created for the scenarios when a document is not yet saved on a server.

## menu_description.xml

This file describes the UI elements. Use the file to make a new content control available to users in Microsoft Word.

There are three main sections in the file:

1. **Menus** – customizes structure of menus, supertips, and screentips.
2. **Labels** – overrides the default menu item names and content control titles.
3. **Content controls** – customizes content controls.

### Menus

Elements and attributes:

- `menu` – describes the static menu. You can create menu items by adding `dynamicMenu` elements. If you want to create a new sub-menu, you need to add another `menu` element within this element.
- `displayName` – defines a menu name in Word.
- `supertip` and `screentip` – display additional information when users move the mouse over the menu.
- `dynamicMenu` – adds menu items based on data from `merge_odata_description.xml`.
    - `contentRefs` – a reference to the entries in `merge_odata_description.xml`.
    - `repeatable` – identifies item in menu that must be used to repeat content control. When repeat section is used around the content control, the menu item with `repeatable="true"` is used.

### Labels

Elements and attributes:

- `label` - overrides the default menu item label for a specific property that is defined in `merge_odata_description.xml`.
    - `dataItemRef` - specifies the reference to the property of an entry.
- `value` - specifies a language-specific label. It will also be used as a content control placeholder.
    - `format` - use this to format a `value` (optionally).

### Content controls

Elements and attributes:

- `NoEntityRules` - defines the behavior of a content control if a user clicks **Merge**, but no value exists for the content control.
    - `<Rule rule="Unchanged">` - adds the content control into this element if you want to remain the content control in the user interface without changes.
    - `<Rule rule="Empty">` - adds the content control into this element if you want to hide the content control from the user interface.
        - `menuDataRefs` - specifies a reference to a `staticData` and `link` in the `merge_odata_description.xml` file.
- `cc` - defines which information to include in the content control title.
- `dynamicMenuRef` - specifies the name of a sub-menu or a menu item for which you want to apply this title.
- `titleFormat` - specifies the content control title.
    - `{displayName}` - displays the name of the menu item that a user has clicked.
- `holderFormat` - optionally, specify the content control placeholder.

## Install customizations

## About XAML files

The XAML files are loaded using the LoadData tool which is located in :

- `Program Files (x86)\ScanJour\Captia\Modules\Services`

  -Or-
- `Program Files\ScanJour\Captia\Modules\Services`

For example, to load a modified Danish-language XAML for the task pane, execute the following command:

```
loaddata.exe /module_name="Scanjour.Services.Office" /name=WordTaskPane /culture_name=da-DK /rank=50 /default=N /file=<your file> /b=sjsysadm /a=<password for sjsysadm> /database=<dnsname for your database>
```

> **Important:**
>
> - The default value must be set to `J` for an English-language file and to `N` for a non-English language file.
> - For the new XAML to take effect, you need to clear the client XAML cache.

## Example of Resource String Used as XAML Label

Example from DocumentRegistrationPane.en-GB.xaml:

```
<UserControl.Resources>
                <system:String x:Key=" LabelTitle">Title</system:String>
                <system:String x:Key="LabelDocumentType">Document type</system:String>
```

The resource strings may then be used for labeling the XAML controls:

```
<Label x:Name="Title_label" Content="{StaticResource LabelTitle}" Style="{DynamicResource LabelStyle}" />
```

In this way, all terms are kept together in the **Resources** section, and they are therefore easier to translate.

The English version of the XAML file (for example, DocumentRegistrationPane.en-GB.xaml), including the terms resources, is considered the master XAML.

## Clear the XAML Cache

When WorkZone for Office is upgraded to a new version, all cached XAML files for current user are automatically cleared upon the first start of a Microsoft Office application.

Cached XAML files are locally stored in:

```
%appdata%\Scanjour\Office\Xaml
```

To clear the cached XAML files manually, delete the `Xaml` folder.

## Load ColumnSetDefinitions

ColumnSetDefinitions are ordinary WorkZone configuration deltas and should be loaded in the usual way by using the loaddatadict.

```
loaddatadict /d:<dsn name for your database> /file:<your file>
/name:custom_ColumnSetDef /type_rank:80 /module_rank:1020
/rank:10
```

> **Note:** This type is not suitable for very large domains (larger than 10.000 items).

## Add a new field to ColumnSetDefinitions

The column set definitions for all standard WorkZone fields are defined by the server setup in the configuration data ColumnSetDefinitions.

If you need to bind extra fields to XAML controls, you must add them to the ColumnSetDefinitions setup.

If you need the standard data dictionary setup for a field, you only need to list the new field in the module specific `ColumnSetDefinitions.xml` file.

> **Example**:
>
> `<Column name="medium" />`

# Working with macro-enabled templates



## Create macro-enabled templates

You can create your own macro-enabled templates to set default values and to perform merge and other important functions in WorkZone for Word.

> **Prerequisite:** To use macro-enabled templates, you need to enable macros first.

### Create a macro-enabled template

1. Open a new Microsoft Word document.
2. On the **Developer** tab in the **Code group**, click **Visual Basic**.

   -Or-

   Press **Alt+F11**.

   A new Visual Basic document opens.

### Example of an advanced document template

In this example, we are creating a macro-enabled template for a standard letter that prompts the user to select a case for a document and then recipients. Finally, it merges all the information into the document, after which the user can write the content of the letter.

```
Public WithEvents App As Word.Application
     Public IsScriptExecuted As Boolean
     Private Sub App_DocumentChange()
        If Not IsScriptExecuted Then
               Dim addIn As COMAddIn
               Set addIn = Application.COMAddIns("Scan-
jour.Office.WordAddIn")
               IsScriptExecuted = True
                       If Not addIn.Object.IsMergedDocument Then
                              ' <- Place your code here (START)
                              addIn.Object.SetDocumentMetadata "title",
                              addIn.Object.SetDocumentMetadata "record_
                              addIn.Object.SetDocumentMetadata "record_
                              addIn.Object.RegistrationPaneVisible = Tr
                              addIn.Object.ShowDocumentRecipientsDialog
                              addIn.Object.Merge
                              ' <- Place your code here (END)
                        End If
        End If
     End Sub
```

## Display the Developer tab

By default, the **Developer** tab is not displayed in the ribbon. To display it, you need to perform the following steps:

1. Start Microsoft Word.
2. On the **File** tab, select **Options**. The **Word Options** dialog box displays.
3. In the categories pane, select **Customize Ribbon**.
4. In the **Main Tabs** list, select the **Developer** check box.
5. Click **OK**.
6. On the **Developer** tab that has just appeared, click **Visual Basic**.
7. In the **Project** pane, right-click the **Microsoft Word Objects** folder, and select

**Insert > Class Module**. A new **Class1** class is added to the project tree.

8. Rename the Class1 module to ThisApplication.

### Rename

1. Select the **Class1** module in the project tree. A **Properties** pane is displayed under the project tree.
2. In the **(Name)** field, set **ThisApplication**.

9. In the **Project** pane, double-click **ThisApplication** in the project tree. A new ThisApplication document opens.
10. Enter a code in the ThisApplication document.
11. Right-click your code project. Then click the **Insert** menu, and click **Module**.
12. In the **Properties** pane (lower left corner of screen), click the **Name** field, and type **AppEventHandler**.
13. In the **Code** pane (right side of screen), type the following:
    Dim wordApp As New ThisApplication.

```
Sub AutoNew()
        Set wordApp.App = Word.Application
        wordApp.IsScriptExecuted = False
End Sub
```

14. Save the document template in .dotm format.
15. To use your template, just copy it to a target machine, and open it. As a result, a new document will be created based on your template (that is, with your template script executed).

**Tip:**

- To modify your template, right-click it from **Windows Explorer**, and select **Edit** from a context menu.

- You can have more than one macro template, however only one template will be used at the same time.

## Macro interface members

The following interface members can be used to build macro-enabled templates for WorkZone for Word:

| Macro interface member | Description |
|---|---|
| String `ShowCaseCreateDialog()` | Displays the **Create Case** dialog box. Returns the key of a created case or null. |
| String `ShowCaseSearchDialog()` | Displays the **Search Case** dialog box. Returns the key of a selected case or null. |
| String `ShowDocumentOpenDialog()` | Displays the **Document Open** dialog box. Returns the key of an open document or null. |
| Bool `RegistrationPaneVisible { get; set; }` | `True` in order to display the **Registration** pane; otherwise `False`. |
| Void `ShowDocumentPartiesDialog()` | Displays the **Add Document References** dialog box. You can manage the document references interactively. All changes made in the dialog are saved to the current document. |
| Void `ShowDocu-mentRecipientsDialog()` | Displays the **Add Recipients** dialog box. You can manage the document recipients interactively. All changes made in the dialog box are saved to the current document. |
| Void `SetDocumentMetadata` (string `key`, object `value`) | Sets document meta data (for example, Title, Document Type, Document Group, Case Handler, Parties, Document References, and so on). |
| Object `GetDocumentMetadata (string key)` | Gets current document meta data (including document number, parties document references, and so on). |

| Macro interface member | Description |
|---|---|
| Void `AddDocumentParty(string addressKey, string partyRole)` | Adds the corresponding contact of specified addressKey as a document party. |
| Void `AddDocumentReference(string documentKey, string documentRole)` | Adds the document reference to the document, identified by `documentKey` to the current document. |
| Void `Merge()` | Perform merge to content controls. |
| String `SaveDocument()` | Saves the current document. Returns ID of the saved document. |
| Void `AttachDocumentToCase(string caseKey)` | Attaches the document to a specific case, and show the **Registration** pane if it is not visible. |
| String `GetODataEndpointAddress()` | Turns current endpoint address for OData services. |
| Bool `IsMergedDocument { get; set; }` | `True` if this document is created in the process of merging to multiple recipients; otherwise `False`. |
| `Before_Merge` (bool `Cancel`) | The member notifies that the merging starts. Set the `Cancel` parameter to `True` if you want to cancel the merging. The member is used for a specific document only.<br>To use the member, VBA must subscribe to it by using this method: `RegisterCallbackOnActiveDocument(string callbackMethod, object callbackOwner)`, where `callbackMethod` stands for `Before_Merge` and `callbackOwner` stands for a VBA module where `Before_Merge` is declared.<br>To unsubscribe, use this method: `UnregisterCallbackOnActiveDocument(string callbackMethod, object callbackOwner)`. These methods are applied |

| Macro interface member | Description |
|---|---|
| | to the active document. |
| `After_Merge` (int `RecordKey`, int `FileKey`) | The member notifies that the merging ends and provides case and document IDs of the merged document.<br>To use the member, VBA must subscribe to it by using this method: `Register-CallbackOnActiveDocument(string callbackMethod, object call-backOwner)`, where `callbackMethod` stands for `After_Merge` and `callbackOwner` stands for VBA module where `After_Merge` is declared.<br>To unsubscribe, use this method: `Unre-gisterCallbackOnActiveDocument (string callbackMethod, object callbackOwner)`. These methods are applied to the active document. |
| bool `WorkZone_BeforeMerge` (string `id`) | Use it as the alternative to `Before_Merge` or use both members simultaneously. In contrast to `Before_Merge`, `WorkZone_BeforeMerge` applies globally within the WorkZone for Office add-in and doesn't require registration. If `True`, merging is cancelled. If `False`, merging starts. |
| `WorkZone_AfterMerge`(string `id`) | Use it as the alternative to `After_Merge` or use both members simultaneously. In contrast to `After_Merge`, `WorkZone_BeforeMerge` applies globally within the WorkZone for Office add-in and doesn't require registration. |

All possible key values for your documents are defined in the ColumnSetDefinitions file. You can edit them according to your needs. See Customize view and dialog boxes.

# API

## API integration

### About integration

You can integrate your system with the WorkZone for Office add-in, access the existing files, and create new documents via the API. To do this, you must use the `wzfo.integration.js` javascript module. The module includes asynchronous methods that return `jQuery` promises. Use the promises to track the status of asynchronous operations.

When you install the WorkZone for Office Server, the installer adds the `wzfo.integration.js` file to the WorkZone Server.

### Available API methods

| Method | Description |
|---|---|
| `initialize` | Optional method. Determines language settings of the integration module. It receives the `cultureName` argument in the format like `da-DK` or `da`. |
| `isWordEnabled` | Returns `true` if Microsoft Word and WorkZone for Office add-in for Word are installed. Otherwise, the method returns `false`. |
| `isExcelEnabled` | Returns `true` if Microsoft Excel and WorkZone for Office add-in for Excel are installed. Otherwise, the method returns `false`. |
| `isPowerpointEnabled` | Returns `true` if Microsoft PowerPoint and WorkZone for Office add-in for PowerPoint are installed. Otherwise, returns `false`. |
| `isOutlookEnabled` | Returns `true` if Microsoft Outlook and WorkZone for Office |

| Method | Description |
| --- | --- |
| | add-in for Outlook are installed. Otherwise, returns `false`. |
| `openEmail` | Opens a requested email. This method requires WorkZone record Id.<br>`{recordId:<id>}` |
| `replyDocument` | Creates a new Microsoft Word document that is linked to an existing document. This method requires a document ID of the existing document.<br>`{recordId:<id>}` |
| `createEmail` | Creates a new email with the following parameters:<br>• **Subject**<br>• **Body**<br>• **Recipients:**<br>  • **To**<br>  • **Cc**<br>  • **Bcc.**<br>`{Subject:"<Test sub-ject>",Body:"<test body>", Recipients:{To:"<to@lmdom.local>", Cc:"<c-c@lm-dom.-local>",Bcc:"<bcc@lmdom.local>"}}` |
| `attachDocuments` | Creates a new email with an attachment. This method requires the following parameters:<br>• **ids** - Specify the IDs of the documents that must be attached to this email.<br>• **asPdf** - This defines the file format of the attached files. Set this parameter to `false` to attach the doc- |

| Method | Description |
| --- | --- |
| | uments in the original file format. Set this parameter to `true` to attach the documents as PDF files.<br><br>**Example**: {ids:[<id>,<id>],asPdf:false}<br><br>**Note:** If there is no PDF file of the requested document on the server, neither the PDF nor the original file will be attached. |
| `createProcessOverviewFolder` | Creates a new folder in Microsoft Outlook. This folder is located in **Process Views -> My Views**. This method requires the following parameters:<br><br>• **name** - Specify a name for the new folder.<br>• **folderUrl** - Specify an address for the content to be displayed in the folder.<br><br>`{name:"<name>",folderUrl:"<URL>"}` |

## Merge API

WorkZone for Office enables you to customize content controls and merge them with information by using public API. Find more information here:

- Walkthrough: Binding Content Controls to Custom XML Parts
- Custom XML Parts Overview

To use public API, install the compiled version of WorkZone for Office and the following library assemblies:

`Kmd.Wzfo.CustomXmlParts.Bootstrapper.dll`

`Kmd.Wzfo.CustomXmlParts.dll`

`Kmd.Wzfo.CustomXmlParts.Workzone.dll`

`Scanjour.Office.Services.Interfaces.dll`

`Scanjour.Office.ODataClient.dll`

`Scanjour.OData.Client.dll`

`Scanjour.Utils.dll`

`Kmd.Wzfo.CustomXmlParts.Bootstrapper.dll` is the entry point library assembly. It contains `WorkzoneCustomXmlPartBuilderFactory` with method `'ICustomXmlPartBuilder Create(BuilderParameters parameters, IProtectedAddressesAcceptor protectedAddressesAcceptor)'`

To generate data source for WorkZone for Office content controls, proceed with the following steps:

1. Create instance of `ICustomXmlPartBuilder` by running `WorkzoneCustomXmlPartBuilderFactory.Create` with appropriate arguments;
2. Build custom xml part

## BuilderParameters

```
public class BuilderParameters
	{
		public BuilderParameters(Uri oDataEndpoint)
		{
			ODataEndpoint = oDataEndpoint;
			CredentialsProvider = new DefaultNetworkCredentia
			CacheDuration = new TimeSpan(0, 0, 10, 0);
			CultureProvider = new ApplicationLocalizer();
		}
		public Uri ODataEndpoint { get; }
		public ICredentialsProvider CredentialsProvider { get; se
		public TimeSpan CacheDuration { get; set; }
		public ICultureProvider CultureProvider { get; set; }
		public bool UseOAuth { get; set; }
		public string AccessToken { get; set; }
	}
```

where:

- `ODataEndpoint` - OData endpoint for building xml part, for example, *https://d-b01/OData;*
- `CredentialsProvider` - Interface for getting OData credentials;
- `CacheDuration` - Duration of the cache that will be used by WorkZone for Office caching mechanism;
- `CultureProvider` - Extracts data of specific language for OData.
- `AccessToken` - Used in token-based authentication to allow an application to access an API if your organization uses OAuth2 for user authentication.
- `UseOAuth` - Enables using an access token for OAuth2 authentication if your organization uses such authentication.

## IProtectedAddressesAcceptor

This interface is used to set rules for handling protected addresses. By default, its values are set to false, so that protected address fields are hidden on UI.

```
public interface IProtectedAddressesAcceptor
        {
                bool AcceptCasePartyAddress(IParty party);
                bool AcceptRecordPartyAddress(IParty party);
                bool AcceptCaseOfficerAddress();
                        bool AcceptCaseResponsibleOuAddress();
                        bool AcceptRecordOfficerAddress();
                        bool AcceptRecordResponsibleOuAddress();
        }
```

## ICustomXmlPartBuilder

You can use the build custom xml part as data source for WorkZone for Office content controls.

```
public interface ICustomXmlPartBuilder
        {
                Task<string> BuildAsync(string recordId, IEnu-
merable<IParty> recordParties, IEnumerable<IReadCustomXmlMapping>
```

```
contentControls);
                Task<string> BuildAsync(string recordId, string caseId,
IEnumerable<IParty> recordParties, IEnu-
merable<IReadCustomXmlMapping> contentControls);
        }
```

where:

- `recordId` - document ID that is used as a data context for building custom xml part.
- `caseId` - case ID that is used as a data context for building custom xml part.

In the upper `BuildAsync` method, case ID is retrieved from primary case metadata by using `recordID`. In the lower `BuildAsync` method, case ID is retrieved directly from `caseID`.

If you get `AuthenticationException` when using the `BuildAsync` method in the OAuth2 mode, it means that the current access token has expired and another one should be obtained.

## IParty

For Merge API, this interface is used to describe case party properties, for example, case party role:

```
new PartyRole ("Afsender", "AP")
       public interface IParty
        {
               string AddressKey { get; }
               PartyRole Role { get; }
        }
```

## IReadCustomXmlMapping

This interface contains information about Content Control for which we build custom xml data.

```
public interface IReadCustomXmlMapping
        {
        string XmlMappingPrefix { get; }
        string XmlMappingXPath { get; }
        string Tag { get; }
```

```
        ContentControlType Type { get; }
        }
```

Ensure that your client extracts correct data from content controls via Merge API and then merge the data with the instance of `IReadCustomXmlMapping`.

# Terms and conditions

## Intellectual property rights

This document is the property of KMD. The data contained herein, in whole or in part, may not be duplicated, used or disclosed outside the recipient for any purpose other than to conduct business and technical evaluation provided that this is approved by KMD according to the agreement between KMD and the recipient. This restriction does not limit the recipient's right to use information contained in the data if it is obtained from another source without restriction set out in the agreement between KMD and the recipient or by law.

## Disclaimer

This document is intended for informational purposes only. Any information herein is believed to be reliable. However, KMD assumes no responsibility for the accuracy of the information. KMD reserves the right to change the document and the products described without notice. KMD and the authors disclaim any and all liabilities.