Exact Online Premium - Extensions

Table of contents

- Introduction
- Getting Started with Extensions
    - Before you begin
    - Extensions ToolTip
    - Guidelines to create Extension manifest file
    - Uploading extension XML file
- Overview of Extension Elements and Attributes
    - Extensions
        - Entities
        - Mega menu extensions
        - Quick menu extensions
        - Power BI report extensions
            - Encoding example
        - Application Extensions
            - Button
            - Monitor Item
            - Card Section
            - Content Section Row
            - Grid Page Header
            - Grid Column
            - Filter Block
            - Column Group
        - Division Settings
        - Translation Extensions
    - Attributes
        - Common attributes
        - Custom Field
            - Attributes
        - Reference (GUID type) field
- Concepts and development options
    - Mandatory Legislation
    - Mandatory Featuresets
    - Forbidden Featuresets
    - Feature Check
    - Expression Engine
        - Example expressions
        - Built-in functions
        - Example expressions with built-in functions
        - Exact Online Function Library
            - Visible Expression
            - Default Expression

    - Type transformations
- Functional Validations with Azure Functions
    - Create an Azure Function from Template
- Validation Function Rule
- Installing Extension
    - Publishing an extension to the Extensions catalogue
- Extensions catalogue
- Enable or disable an extension
- Uninstall, downgrade or upgrade versions of the extension
- Adding custom data to the layout
- Putting all the pieces together

Introduction

An **extension** for Exact Online Premium is a low-code solution that enables you to make modifications or add extra functionality to the standard Exact Online Premium solution.
If you're new to building Exact Online Premium extensions, we recommend that you read this document to get an understanding of the basics and terms you'll encounter while working.

The extension that you create should be in an XML file. The file is often referred to as an **extension manifest file**. Detailed information on steps in defining an extension and sections of an extension manifest file is explained below with examples.

Extensions implemented with this low-code platform will support:

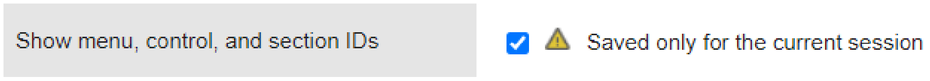
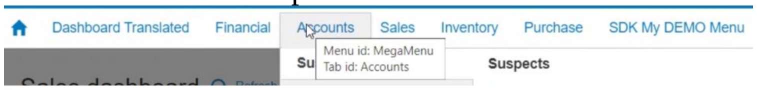
- Extending standard entity and data model, to support custom data fields
- Extending standard business logic, to support custom validation or custom defaults adding custom functional components
- Extending standard user interfaces, adding custom UI elements, like custom buttons or input fields, on standard Aurora pages and menus

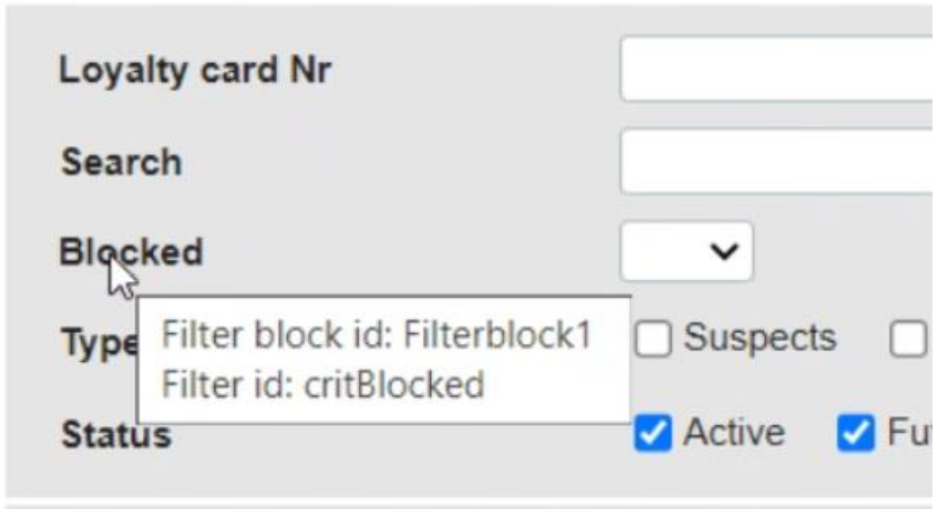
Getting Started with Extensions

Before you begin

- Check for the unique generated extension developer prefix applicable to you. You will need to use this when creating the extension file. To check this, go to **My Exact Online** > **My account** > **Extension developer prefix**.
- The extension developer prefix is auto-generated, but as long as there is no extension created with the generated code, it can be edited from **My Exact Online** > **My account** > **Extension developer prefix**.

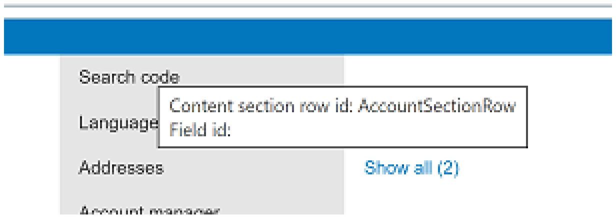
Extensions ToolTip

- A helpful tooltip for extensions developer to get the IDs of MegaMenu, QuickMenu, sections, and controls. (application extensions)
- Here is the menu path: `My Exact Online > User setting tab > Developer section`
  - 
    Show menu, control, and section IDs ✓ Saved only for the current session
  - When this setting is enabled, existing IDs will be shown when you hover over the MegaMenu, QuickMenu, sections, and controls.
    - Snapshots of extensions tooltip:
      - 

        Menu id: MegaMenu
        Tab id: Accounts
      - 

        Filter block id: Filterblock1
        Filter id: critBlocked

      If the id value is blank in the tooltip, then it is not possible to customize that control. For example, it is not possible to hide the search code field shown in the example below.

      Content section row id: AccountSectionRow
      Field id:

Guidelines to create Extension manifest file

The extension that you create should be an XML file. You are free to assign the file any name you like. However, it is recommended that you give it a descriptive and meaningful file name, and preferably use the naming convention **Extension.[Extension developer prefix]_[Short solution description].xml**. For example, if your extension developer prefix is DEVPP, an extension to add a new menu (such as warranties) to the navigation bar at the top of the Exact Online Premium page might be called Extension.DEVPP_AddWarrantiesMenu.xml.

It is recommended that you use an XML validation-enabled editing tool (such as Notepad++ or Visual Studio Code). At the start of your development work the file should contain the lines presented below in the section Default XML file':

XML schema file is helpful for validating element names, attribute names, permissible data types and required values.

XML schema file, Extension.xsd, can be downloaded from the below menu location.

Master Data → Customisation and configuration → Developer | Maintain extensions →Developer resources → Download XSD

Once downloaded, take below steps to use the schema file.

1. Use a xml validation enabled editing tool such as;
   1. Notepad++ with xml validation addon
      - download notepad++
      - how to enable xml tool in notepad++
   2. Visual Studio Code (has a built-in xml validation)
      - download Visual Studio Code
   3. **Default XML file**

```
<?xml version="1.0" encoding="utf-8"?>
<extension code="(your solution code)" version="1.0.0(your
version)" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="(local Path)">
<!--
        Place you extension data here
          -->
</extension>
```

3.Update xsi:noNamespaceSchemaLocation on the manifest xml with the local location of downloaded extension.xsd file.

4. It is important that the sequence of the xml elements should follow the schema definition

There are some common attributes supported by custom UI Elements. For example, caption attribute is used to set the display text for a button. Please see the Common Attributes section for more details.

Uploading extension XML file

Follow the steps below to upload an extension XML file.

- Go to **Company name > Master data > Customisation and configuration** and click **Developer: Maintain extensions**.
- Click **New** to create an extension and enter the following information:
    1. **Code**: The name of the extension (aka **solution code**), using the unique generated prefix. Use the following naming convention: '**[Extension developer prefix]_[Short solution description]**' (e.g. ‘DEVPP_YourOwnShortDescription’). Make sure the solution code is unique. This is what you will use when uploading the file to create a new version of an extension.
    2. **Description**: A brief description of the extension.
    3. **Title**: Title of the extension

  You’re now ready to upload the file. Click the name of the newly created extension to open it.

  You can upload multiple extensions at once, and they can be used to customize the same Exact Online Premium pages. Each extension is uniquely defined by its extension code, the UI elements, business components, and data extensions, and has its own set of data.

- Click the **New** hyperlink. This opens the page for you to upload a new version of the extension XML file. Enter semantic versioning in the **Version** field (see Semantic Versioning for more details) and then click **Upload a new file** to add it.
- Click **Save** to add the new version.

Currently, there is a limitation that the versions are ordered string based, so "2" is greater than "10", which should be worked around.

Overview of Extension Elements and Attributes

Extensions

Entities

Entities are definitions to extend the properties of a business component. Also, the scripts for the extended table are generated and executed when the extension is installed. Attributes and their details are explained below.

`<entity>` - definition of an entity to be extended.

    `table` - Name of the table from exact online to be extended.

Please note only tables having division column are extendable `extensiontable` - Name of the new extended table. The table is automatically created based on the extended configuration.

`businesscomponent` - Name of the business component to be extended.

`<property>` - defines a column in an extended SQL table and the property of a business component extended.

`name` - Name of the property in the business component must be prefixed with code

`columnname` - Name of the column in extended SQL table.

`<listitems>` -When the ‘integer’ type is used, then a list of choice items can be supplied with <listitems>. If not supported with <listitems> then integers are displayed as ranges. The value of the selected item is stored in the `integer` column.

`type` - represents the data type of the property or column. The supported datatypes are Boolean, integer, string, double, date, hyperlink, and GUID.

`integer` - generates a number column in the extended table and creates an integer type property in the business component.

`string` - generates `nvarchar` and the length is from the `length` attribute and creates string property in the business component.

`boolean` - generates `tinyint` and creates Boolean property in the business component.

`guid` - generates `uniqueidentifier` and creates a GUID property in the business component.

`refersto` - the name of the referencing entity. (See Reference (GUID type) field section for more details)

```
<entities>
   <entity name="Account" table="Accounts"
extensiontable="SDK_LoyalPr_Accounts" businesscomponent="Account">
      <property name="SDK_LoyalPrLoyaltyMember" columnname="LoyaltyMember"
type="boolean" caption="Loyalty member" />
      <property name="SDK_LoyalPrLoyaltyCard" columnname="LoyaltyCard"
caption="Loyalty card" type="string" length="18" />
      <property name="SDK_LoyalPrLoyaltyDiscount"
columnname="LoyaltyDiscount" caption="Loyalty discount" type="double" />
      <property name="SDK_LoyalPrLoyaltyLevel" columnname="LoyaltyLevel"
type="integer" caption="Loyalty level">
```

```
                <listitems allowempty="true"
listdefinitionid="LoyaltyLevelList">
                        <listitem value="1" caption="Entry" captionid="0" />
                        <listitem value="3" caption="Gold" captionid="0"
translationid="SDK_LoyalPrGoldTranslated" />
                        <listitem value="2" caption="Silver" captionid="0" />
                        <listitem value="4" caption="Platinum" captionid="0"
translationid="SDK_LoyalPrplatinumTranslated" />
                </listitems>
      </property>
      <property name="SDK_LoyalPrAccount" columnname="Account"
caption="Custom Solution Account" type="guid" refersto="Account" />
      <property name="SDK_LoyalPrProject" columnname="Project"
caption="Custom Solution Project" type="guid" refersto="Project" />
      <property name="SDK_LoyalPrContact" columnname="Contact"
caption="Custom Solution Contact" type="guid" refersto="Contact" />
      <property name="SDK_LoyalPrOpportunity" columnname="Opportunity"
caption="Custom Solution Opportunity" type="guid" refersto="Opportunity" />
      <property name="SDK_LoyalPrSalesOrder" columnname="SalesOrder"
caption="Custom Solution SalesOrder" type="guid" refersto="SalesOrder" />
      <property name="SDK_LoyalPrItem" columnname="Item" caption="Custom
Solution Item" type="guid" refersto="Item" />
    </entity>
</entities>
```

In Entity elements of an XML file, you can only link one business component to a table extension. That works well for most extensions, but extending purchase/sales entry lines gives a problem. For example, if you extend GLTransactions with business component = PurchaseEntryLine, then you can extend FinInvoiceEntry.aspx with the properties defined in the entity.

```
<entity name="GLTransaction" table="GLTransactions"
extensiontable="CTA_XmlValidationIssue_GLTransactions"
businesscomponent="PurchaseEntryLine">
      <property name="CTA_XmlValidationIssue_MyColumn"
columnname="CTA_XmlValidationIssue_MyColumn" type="double" caption="MyColumn"
/>
</entity>
```

But then the sales entry page gives a runtime error. This is because this page dynamically uses either the PurchaseEntryLine or SalesEntryLine business component, and it expects that all used properties exist in both business components. To solve this, you must define the entity twice, as you can see in the following example - Once for the Purchase Entry Line, and once for the SaleEntryLine business component:

```
<entity name="GLTransaction" table="GLTransactions"
extensiontable="CTA_XmlValidationIssue_GLTransactions"
businesscomponent="PurchaseEntryLine">
<property name="CTA_XmlValidationIssue_MyColumn"
columnname="CTA_XmlValidationIssue_MyColumn" type="double" caption="MyColumn"
/>
</entity>
```

```
<entity name="GLTransaction" table="GLTransactions"
extensiontable="CTA_XmlValidationIssue_GLTransactions"
businesscomponent="SalesEntryLine">
<property name="CTA_XmlValidationIssue_MyColumn"
columnname="CTA_XmlValidationIssue_MyColumn" type="double" caption="MyColumn"
/>
</entity>
```

Mega menu extensions

The Mega menu is the main menu that is displayed on the navigation bar located at the top of the Exact Online Premium page. Each menu is defined in `<Tab>` element. `<section>` defines the immediate section under a menu and `<subsection>` are the sections under each menu item. Each subsection can have a link defined with `<link>` element.

`<megamenuextensions>`

- `<tab>` - describes a new menu with the name defined in the `caption` attribute.
  - `<section>` - section in the menu.
    - `<subsection>` - subsections are menu items under a section.
      - `<link>` - Links are clickable menu items with the destination URL set with `href` attribute.

        If you want to open the link in a new tab, you can do so by setting the **showinnewtab**="true" attribute. This attribute must be used for **non-EOL URLs**.

For example please see below code snippet.

```
        <megamenuextensions>
                <megamenuextension >
                        <tab id="Categories" caption="Categories"
existing="false">
                                        <section id="Digital" caption="Digital"
captionid="0" existing="false">
                                                <subsection id="Electronics"
caption="Electronics" captionid="0" existing="false">
                                                        <link id="Televisions"
caption="Televisions" captionid="0" href="https://sample.com/televisions"
existing="false" showinnewtab="true" />
                                                </subsection>
                                                <subsection id="Games"
caption="Games" captionid="0" existing="false">
                                                        <link id="PS4"
caption="PS4 Games" captionid="0" href="sample.aspx/ps4games"
existing="false" showinnewtab="true"  />
                                                </subsection>
                                        </section>
```

```
                              </tab>
                </megamenuextension>
        </megamenuextensions>
```

Quick menu extensions

The Quick menu is accessible easier for faster navigation by going to Create menu located at the top right corner of the Exact online navigation bar.

`<quickmenuextensions>`

`<section>` - section in the menu.

`<link>` - Links are clickable menu items with the destination URL set with `href` attribute.

If you want to open the link in a new tab, you can do so by setting the **showinnewtab**="true" attribute. This attribute must be used for **non-EOL URLs**.

For example please see below code snippet.

```
<quickmenuextensions>
                <quickmenuextension menuid="MyQuickMenu">
                        <subsection id="QuickLinks" caption="Quick Links"
captionid="0" existing="false" >
                                <link id="google" caption="Google"
captionid="0" href="https://www.google.com" existing="false"
showinnewtab="true"/>
                        </subsection>
                        <subsection id="Favourites" caption="Favourites"
captionid="0" existing="false">
                                <link id="youtube" caption="Youtube"
captionid="0" href="https://www.youtube.com" existing="false"
showinnewtab="true"/>
                        </subsection>
                </quickmenuextension>
        </megamenuextensions>
```

Power BI report extensions

The `powerbilink` element appears like a `link` element but has a special purpose. When clicked, the link redirects to a page in Exact Online that renders the report internally.

`<megamenuextensions>`

`<tab>` - describes a new menu with the name defined in the caption attribute.

`<section>` - section in the menu.

`<subsection>` - subsections are menu items under a section.

`<powerbilink>` - the Power BI link to render

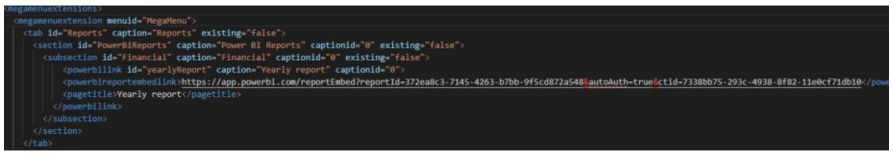
`<powerbireportembedlink>` - The link of the report to embed. Make sure the link you paste in is encoded: all occurrences of `&` should be replaced with `&amp;`.

`<pagetitle>` - (Optional) the title to display on top of the embed page. When omitted, the default title "Power BI Report" will be used.

For example, see the following code snippet:

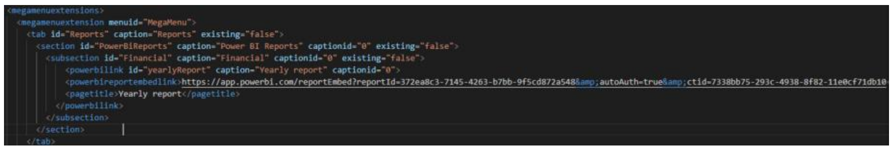
```
<megamenuextensions>
        <megamenuextension>
                <tab id="Reports" caption="Reports" existing="false">
                        <section id="PowerBiReports" caption="Power BI Reports"
captionid="0" existing="false">
                                <subsection id="Financial" caption="Financial"
captionid="0" existing="false">
                                    <powerbilink id="yearlyReport"
caption="Yearly report" captionid="0">

        <powerbireportembedlink>https://app.powerbi.com/reportEmbed?reportId=3
72ea8c3-7145-4263-b7bb-9f5cd872a548&amp;autoAuth=true&amp;ctid=7338bb75-293c-
4938-8f82-11e0cf71db10</powerbireportembedlink>
                                                <pagetitle>Yearly
report</pagetitle>
                                        </powerbilink>
                                </subsection>
                        </section>
                </tab>
        </megamenuextension>
</megamenuextensions>
```

Encoding example



```
<megamenuextensions>
  <megamenuextension menuid="MegaMenu">
    <tab id="Reports" caption="Reports" existing="false">
      <section id="PowerBiReports" caption="Power BI Reports" captionid="0" existing="false">
        <subsection id="Financial" caption="Financial" captionid="0" existing="false">
          <powerbilink id="yearlyReport" caption="Yearly report" captionid="0">
          <powerbireportembedlink>https://app.powerbi.com/reportEmbed?reportId=372ea8c3-7145-4263-b7bb-9f5cd872a548&autoAuth=true&ctid=7338bb75-293c-4938-8f82-11e0cf71db10</powe
          <pagetitle>Yearly report</pagetitle>
        </powerbilink>
      </subsection>
    </section>
  </tab>
```

Should be:



```
<megamenuextensions>
  <megamenuextension menuid="MegaMenu">
    <tab id="Reports" caption="Reports" existing="false">
      <section id="PowerBiReports" caption="Power BI Reports" captionid="0" existing="false">
        <subsection id="Financial" caption="Financial" captionid="0" existing="false">
          <powerbilink id="yearlyReport" caption="Yearly report" captionid="0">
          <powerbireportembedlink>https://app.powerbi.com/reportEmbed?reportId=372ea8c3-7145-4263-b7bb-9f5cd872a548&amp;autoAuth=true&amp;ctid=7338bb75-293c-4938-8f82-11e0cf71db10
          <pagetitle>Yearly report</pagetitle>
        </powerbilink>
      </subsection>
    </section>
  </tab>
```

Application Extensions

Application extensions extend the functionality of a page (.aspx) of Exact Online. UI Elements extended from a page are defined with `<applicationextenssion>` element in the manifest file. Below are the details of supported custom UI elements to extend in a page of Exact Online with examples.

There are some common attributes supported by custom UI Elements. For example, caption attribute is used to set the display text for a button. Please see the Common Attributes section for more details.

Button

A button can be added to a page with href defined to navigate to the destination URL or relative path to an aspx page.

`<button>` - custom button

> `href` - destination URL or relative path to an aspx page.
>
> `showinnewtab` - when `true` the button click will take to a new tab from the browser.

```
<applicationextension application="CRMAccounts.aspx">
    <button captionid="0" caption="Google" id="btnGoogle"
href="https://www.google.com" showinnewtab="true" existing="false"/>
</applicationextension>
```

Monitor Item

`<monitor>` - The container of the monitor item.

> `<monitoritem>`
>
> > `href` - destination URL or relative path to an aspx page.
> >
> > `showinnewtab` - when `true` the button click will take you to a new tab from the browser.
> >
> > `image` - Icon for monitor item (If there is no image given in the attribute, by default the image will use mtr_docs.gif). The images that are available in Exact Online can only be used for the image attribute. Note: A list of available images will be provided in the final document.

Monitor Items are buttons with images that can be added with href navigating to the destination URL or relative path to an aspx page.

```
<applicationextension application="FinGLAccount.aspx">
```

```
        <monitor id="csMonitor" existing="true">
                <monitoritem id="MonitorItem" caption="Accounts" captionid="0"
href="CRMAccounts.aspx" showinnewtab="true" image="icon-a-customer.gif"
existing="false" />
        </monitor>
</applicationextension>
```

It is not possible to add a new monitor. Even existing="false" is given for the <monitor>

Card Section

New fields can be added to an existing section of a card page by referring to its `id`.

`<cardsection>` - custom card section with the possibility to extend an existing card section or add a new one.

`<field>`

`href` - destination URL or relative path to an aspx page.

`showinnewtab` - when `true` the button click will take you to a new tab from the browser.

`datasource` - the business component name used in the `application`.

`datafield` - refer to the `<entity>` element's `property` value.

`type` - represents the datatype of the field. The supported datatypes are Boolean, integer, string, double, date, hyperlink, and GUID.

In the below example, `csGeneral` is an existing card section and a field `LoyaltyCard` is added to the section.

```
<applicationextension application="CRMAccount.aspx">
    <cardsection id="csGeneral" existing="true">
           <field id="LoyaltyCard" type="string" length="18" caption="Loyalty
Card" captionid="0" datafield="SDK_LoyalPrLoyaltyCard" datasource="bc"
existing="false"/>
           <field id="LoyaltDiscount" type="string" length="18"
caption="Loyalty Discount" captionid="0"
datafield="SDK_LoyalPrLoyaltyDiscount" datasource="bc" existing="false"/>
    </cardsection>
</applicationextension>
```

It is not possible to add a custom section. Even existing="false" is given for the <cardsection>, new Card Section would not be created and the element will be ignored.

Content Section Row

Sections within a card page are extended with `contentsectionrow`.

```
<contentsectionrow>

    <field>

<contentsectionrow id="AccountSectionRow" existing="true">
    <mandatorylegislation>Netherlands</mandatorylegislation>
    <field id="LoyaltyLevel" type="integer" caption="Loyalty Level"
controltype="radiobuttonlist" datafield="SDK_LoyalPrLoyaltyLevel"
datasource="bc" existing="false" />
</contentsectionrow>
```

Grid Page Header

Custom fields are added to the header sections of a grid page by making use of `gridpageheader` element. This is very similar to extending a card page with `cardsection` or `contentsectionrow`.

```
<applicationextension application="SlsSalesOrderEntry.aspx">
    <gridpageheader>
        <section id="sfHeader" existing="true">
            <field id="LoyaltyDiscount" type="double" caption="Loyalty
discount" captionid="0" datafield="SDK_LoyalPrLoyaltyDiscount"
datasource="grd" existing="false"/>
        </section>
    </gridpageheader>
</applicationextension>
```

Grid Column

Add new columns to a grid page by defining `gridcolumn` element. The position of the new column is set before the column defined with `positionbefore` the attribute.

```
<applicationextension application="SlsSalesOrderEntry.aspx">
    <gridcolumn id="LoyaltyColor" type="string" length="20" width="8%"
caption="Loyalty Card" captionid="0" datafield="SDK_LoyalPrLoyaltyCard"
datasource="grd" positionbefore="colQuantity"  existing="false"/>
</applicationextension>
```

Filter Block

Filter blocks are sections with filter fields used by reporting framework pages. It is possible to extend an existing `filterblock` or add a new one.

```
<applicationextension application="CRMAccounts.aspx">
    <filterblock id="LoyaltyCardFilterBlock" existing="false">
        <filter id="LoyaltyCardNumberFilter" type="string" length="18"
caption="Loyalty card Number" translationid="LoyaltyCardNuberTranslation"
captionid="0" existing="false"/>
```

```
        <filter id="LoyaltyCardHolderFilter" type="boolean" length="18"
caption="Loyalty card Number" translationid="LoyaltyCardNuberTranslation"
captionid="0" existing="false"/>
        <filter id="LoyaltyLevelFilter" type="integer"
controltype="checkboxlist" listdefinitionid="LoyaltyLevelList"
caption="Loyalty level" captionid="0" existing="false" />
    </filterblock>
</applicationextension>
```

Column Group

Columns from the reporting framework pages can be customized.

`<join>` - specifies the `extesiontable` to use.

`<column>` - new column to be added or an existing column to modify.

`field` - refer to the `<entity>` element's `property` value.

`filterid` - refer to the `<filter>` element's `id` to be linked.

`type` - represents the datatype of the column. The supported datatypes are Boolean, integer, string, double, date, hyperlink, and GUID.

```
<applicationextension application="CRMAccounts.aspx">
    <columngroup id="cgGeneral" existing="false">
        <join extensiontable="SDKLoyalPrAccounts"/>
        <column id="textColumn" field="SDK_LoyalPrLoyaltyCard"
filterid="LoyaltyCardNumberFilter" type="string" caption="Loyalty card"
captionid="0" translationid="SDKLoyaltyCard" existing="false"/>
        <column id="booleanColumn" field="SDK_LoyalPrLoyaltyMember"
filterid="LoyaltyCardHolderFilter" type="boolean" caption="Active Loyalty
card" captionid="0" translationid="SDKActiveLoyaltyCard" existing="false"/>
        <column id="levelColumn" field="SDK_LoyalPrLoyaltyLevel"
filterid="LoyaltyLevelFilter" type="integer"
controltype="checkboxlist"  listdefinitionid="LoyaltyLevelList"
caption="Loyalty Level" captionid="0" existing="false">
            <documentation>List of loyalty levels</documentation>
        </column>
    </columngroup>
</applicationextension>
```

Division Settings

Division settings are extended by defining new settings with `divisionsettingsextension` block. This custom `setting` also supports adding the default value.

`<divisionsettingsextensions>`

`<tab>` - New custom tab control is added to the division settings page of Exact Online.

`<section>` - A section with a caption to group the settings.

`<setting>` - defines a setting with the default value set with `defaultvalue` attribute

`type` - represents the type of the setting. type is defined in primitive datatypes such as boolean, integer, string, double, date, and hyperlink

- Setting id must be prefixed with the extension code.

Please see the below example, `Use Enable Facebook` setting is added under the section `SocialMedia` of a new tab `SocialMediaManagement`. The default value for the setting `Enable Facebook Sharing`is true.

```
<divisionsettingsextensions>
    <tab id="SocialMediaManagement" caption="Social Media Management"
captionid="0" translationid="SocialMediaManagementTranslations">
        <section id="SocialMedia" caption="Social Media" captionid="0"
translationid="SocialMediaTranslation">
            <setting id="SDK_PrLoyalFacebook" caption="Enable facebook
sharing" captionid="0" type="boolean" defaultvalue="true"
translationid="FacebookSharingEnableTranslation" />
            <setting id="SDK_PrWhatsapp" caption="Enable whatsapp sharing"
captionid="0" type="boolean" defaultvalue="false"
translationid="WhatsappSharingEnableTranslation" />
        </section>
    </tab>
</divisionsettingsextensions>
```

Translation Extensions

All the translations on captions used in custom UI elements can be defined with `translationid` attribute. Right translation matching the id is retrieved from the list of translations defined in `<translationextensions>`.

`<translationextensions>`

`<translation>` - `id` is referred to by UI Elements.

`<language>` -

`code` - ISO Language code

Please see the below example, button `New` looks up for a matching translation from `<translationextensions>` based on the Exact Online current language set in the user profile.

```
<translationextensions>
    <translation id="NewButtonTranslation" >
        <language code="en">New</language>
        <language code="nl-BE">Nieuwe</language>
        <language code="nl">Nieuwe</language>
    </translation>
</translationextensions>
<applicationextension application="CRMAccounts.aspx">
    <button captionid="0" caption="New" id="btnNew" href="NewSomething.aspx"
showinnewtab="false" translationid="NewButtonTranslation"/>
</applicationextension>
```

Attributes

Common attributes

`caption` - Sets the text to be displayed for the custom control.

`captionid` - the existing id from Exact Online referring to a predefined caption text.

`translationid` - unique id referring to the translation defined in `<translationextensions>` element in the extension manifest file.

`existing` -For the customization of standard controls, set existing to true and specify the id of the intended standard control.

`positionbefore` - used for placing the newly created control before the specified standard control. Provide the ID of the standard control where the new custom control can be placed before.

`visibleexpression` - a child element supporting the ability to make the UI element visible or invisible with an expression using the expression engine. Please see Visible Expression | Expression Engine.

`controltype` - custom control type to define the type of the control to be used for a custom UI Element. Supported to use with `cardsection`, `contentsectionrow`, `gridpageheader`, `gridcolumn`, `filterblock`, and `columngroup`.

> `default` - uses the default type of the UI control predefined in the code.
>
> `list` - renders a combo box control with the values.
>
> `checkboxlist` - multi-selectable checkbox list.
>
> `radiobuttonlist` - single selectable radio button group.

Custom Field

`field` element defines a custom UI Element of various types. Below are detailed explanations of the attributes supported by a field and its behavior.

Attributes

`type` - a type of control defined using primitive types, for example, type string renders a text field and type number renders a field where only numbers are allowed. Please see Type Transformations for more info.

`length` - length of the field if applicable. For example, you can limit the number of characters allowed in a field.

`width`- width of the field control.

`readonly` - if true, the field is not editable.

`positionbefore` - places the field before an element matching the id set with this attribute.

`controltype` - defines how a list control is rendered. possible options are explained below.

> `default`
>
> `list` - a combo box is rendered with all the list items defined in a `property`. The selected value when saved is stored in the column `ColorCode`.
>
> `checkboxlist` - all the list items defined with the `property` are rendered as a checkbox list. This control is only valid when used for a `FilterBlock`. When used in other elements, the `list` control type is rendered instead.
>
> `radiobuttonlist` - radio buttons within a group are rendered displaying all the list items defined with a `property`. Only one item can be selected and the value is saved in the related column for the list definition.

`refersto` - the name of the referencing entity. It is the same as that of `refersto` attribute value as defined in the corresponding entity element's property

```
<entities>
               <entity name="Account" table="Accounts"
extensiontable="SDK_ZA_Accounts" businesscomponent="Account">
                       <property name="SDK_ColorCodes" columnname="ColorCode"
type="string" caption="Color Code">
                               <listitems allowempty="true"
listdefinitionid="ColorCodeDef">
                                       <listitem value="#000000"
caption="Black" captionid="0" />
                                       <listitem value="#FF0000" caption="Red"
captionid="0"/>
                                       <listitem value="#FFFF00"
caption="Yellow" captionid="0"/>
                               </listitems>
                       </property>
               </entity>
```

```
        </entities>

<applicationextension application="CRMAccounts.aspx">
   <filterblock id="Filterblock" existing="false">
      <filter id="sfColorCode" type="string" controltype="list"
listdefinitionid="ColorCodeDef" caption="Color Code"
captionid="0"  existing="false"/>
   </filterblock>
</applicationextension>
```

As seen in the example below, there is a custom field named 'Loyalty Level'. Its type is determined as integer and its control type as radiobuttonlist. Since we chose this custom field value using radio buttons, we assign the control type as radiobuttonlist. So what would happen if we didn't specify the control type for Loyalty Level? If you do not define the controltype as radiobuttonlist for Loyalty level, the system cannot assign the value you selected via the radio button from UI. If there is no controltype, the system adds a control based on the custom field type. In other words, since the type of the Loyalty Level's special field is an integer, the system will set a check that ‘the value must be in a range between two numbers’.

```
<applicationextension application="CRMAccount.aspx">
                        <cardsection id="csGeneral" existing="true">
                                <field id="LoyaltyLevel"
datafield="SDK_UITLoyaltyLevel" caption="Loyalty Level" type="integer"
controltype="radiobuttonlist" datasource="bc" existing="false" />

                        </cardsection>
</applicationextension>
```

Reference (GUID type) field

Custom field, custom property, custom column, and custom filter can be defined as GUID type. Defining a field (or property or column or filter) as GUID type of an existing entity, makes the field (or property or column or filter) as a reference to that entity so that the field (or property or column or filter) is browsable of that entity.

In order to define a field (or property or column or filter) as GUID type, the attributes `type` and `refersto` are to be defined as "guid" and the name of the referencing entity respectively.

Few examples of defining a GUID (reference) type

Defining a property of GUID type:

`refersto` - the name of the referencing entity. refersto attribute can have one of the values of referstotype element as defined in the schema and currently the supported values are Account, Item, Project, Contact, Opportunity, and SalesOrder.

```
 <property name="SDK_LoyalPrContact" columnname="Contact" caption="Custom
Solution Contact" type="guid" refersto="Contact" />
```

Defining a field of GUID type:

`refersto` - the name of the referencing entity. It is the same as that of `refersto` attribute value as defined in the corresponding entity element's property

```
<field id="Contact" datafield="SDK_LoyalPrContact" type="guid"
refersto="Contact" caption="Custom Solution Contact" readonly="true"
datasource="bc" existing="false" />
```

Defining a filter of GUID type:

`refersto` - the name of the referencing entity.

```
<filter id="sfContact" type="guid" caption="Custom Solution Contact"
refersto="Contact" existing="false" />
```

Defining a column of GUID type:

`refersto` - the name of the referencing entity.

```
<column id="ContactColumn" type="guid" field="Contact" refersto="Contact"
caption="Custom Solution Contact" filterid="sfContact" existing="false" />
```

Supporting GUID type is not applicable for Button, GridColumn, GridPageHeader, Monitor (includes MonitorItem)

Concepts and development options

Some of the features to further extend the behaviour of extending UI Element is as follows. Details with an example of the usage of these elements are described below.

Mandatory Legislation

The `<mandatorylegislation>` restricts visibility of the custom UI Control based on the legislation

For ex:

```
<quickmenuextensions>
        <quickmenuextension menuid="MyQuickMenu">
                        <subsection id="Favourites" caption="Favourites"
captionid="0" existing="false">
                                <link id="youtube" caption="Youtube"
captionid="0" href="https://www.youtube.com" existing="false">

        <mandatorylegislation>Belgium</mandatorylegislation>
                                </link>
                        </subsection>
        </quickmenuextension>
</megamenuextensions>
```

Mandatory Featuresets

You can define which featuresets are required to show the UI controls extended within `<mandatoryfeaturesets>`. Please see the below example, featuresets Account and GeneralPro are required to see the newly added link to `youtube`

```
<quickmenuextensions>
        <quickmenuextension menuid="MyQuickMenu">
                        <mandatoryfeaturesets>
               <featureset>Account</featureset>
               <featureset>GeneralPro</featureset>
             </mandatoryfeaturesets>
                        <subsection id="Favourites" caption="Favourites"
captionid="0" existing="false">
                                <link id="youtube" caption="Youtube"
captionid="0" href="https://www.youtube.com" existing="false"/>
                        </subsection>
        </quickmenuextension>
</megamenuextensions>
```

Forbidden Featuresets

Custom UI Controls are not rendered for users having a featureset that is added in `forbiddenfeaturesets` comma-separated list. Please see the below example, `youtube` link is not rendered for a custom having featureset `Account`.

```
<quickmenuextensions>
        <quickmenuextension menuid="MyQuickMenu">
                        <forbiddenfeaturesets>
               <featureset>Account</featureset>
             </forbiddenfeaturesets>
                        <subsection id="Favourites" caption="Favourites"
captionid="0" existing="false">
                                <link id="youtube" caption="Youtube"
captionid="0" href="https://www.youtube.com" existing="false">

        <mandatorylegislation>Belgium</mandatorylegislation>
                                </link>
                        </subsection>
        </quickmenuextension>
</megamenuextensions>
```

Feature Check

There are 3 options you can set with `featurecheck` and the values set change the behaviour of `<mandatoryfeaturesets>` and `<forbiddenfeaturesets>`.

`All`

- UI controls are rendered only if the user has rights to all of the featuresets defined with `mandatoryfeaturesets` element.
- UI Controls are not rendered only if the user has rights to all of the featuresets defined with `<forbiddenfeaturesets>`.

`Any`

if `featurecheck` is set to `Any`

- UI controls are rendered if the user has rights to at least one of the featuresets defined with `mandatoryfeaturesets` element.
- UI Controls are not rendered if the user has rights to at least one of the featuresets defined with `<forbiddenfeaturesets>`.

`None`

If `featurecheck` is set to None

- No features sets are required

Expression Engine

The expression engine is a framework to parse and evaluate an expression. It supports all common operators, like add, subtract, multiply, divide, comparison operators (==, !=, >, >=, <, <=), and boolean operators (And, Or, Not). Also supports brackets, nesting functions, and all relevant primitive data types, like boolean, int, double, decimal, string, and date.

Example expressions

```
1 + 2           returns 3
1 + 2.0         returns 3.0
1 + 2 * 3       returns 7
(1 + 2) * 3     returns 9
```

Built-in functions

A function library contains functions and/or properties that can be used within the expressions. Below are some of the global functions supported by Expression Engine and their usage

String Functions

Expression	Description
Concat	concatenates one or more strings. Example: Concat('a', 'b', 'c')

Lower(string)	converts the value of a specified string to its lowercase equivalent
Upper(string)	converts the value of a specified string to its uppercase equivalent

Logical Functions

Expression	Description
If(condition, truePart, falsePart)	evaluates a condition and returns either the second parameter or the third parameter, depending on the outcome of the condition (same as the Visual Basic If function)

Date Functions

Expression	Description
Today()	Get the current date
Now()	Get the current date and time
Date(year, month, day)	Date from year, month, and day. NB: the Date function behaves like the Excel Date function. That means that it is allowed to specify negative values for year, month and day, and values that exceed 12 (for month) and 31 (for day)
Year(datetime)	Get the year part in from datetime. Input for this function and similar functions is the output from other date functions as Today(), Now() or Date(year, month, day). Example: Year(Today()).
Month(datetime)	Get the month part in integer from datetime
Day(datetime)	Get the day part from datetime
Hour(datetime)	Get the hour part from datetime
Minute(datetime)	Get the minute part from datetime
Second(datetime)	Get the seconds part from datetime

Example expressions with built-in functions

```
If(1 < 2, 'a', 'b')    returns a
Concat('a', 'b')       returns ab
Lower('Foo')           returns foo
Upper('foo')           returns FOO
Concat(Upper('a'), If(1 > 2, 'b', If(1 > 2, 'c', 'd')))      returns Ad
Date(Year(Today()), Month(Today()), 1) returns the first day of the current
month
Date(Year(Today()), Month(Today()) + 1, 0) returns the last day of the
current month
Date(Year(Today()) + 1, 1, 0) returns the last day of the current year
```

```
Today() + 1                   returns tomorrow
Today() - 1                   returns yesterday
<applicationextension application="CRMAccount.aspx">
     <field id="IsItNewYear" type="string" caption="Happy New Year"
translationid="HappyNewYearTranslationId" captionid="0" existing="false">
               <visibleexpression>Month(Today()) = 1 AND Day(Today()) =
1</visibleexpression>
        </button>
</applicationextension>
```

Exact Online Function Library

Apart from the global functions explained above below are some of the additional functions supported in Exact Online using the powerful Expression Engine.

Expression	Description
DivisionSetting(settingName, defaultValue)	Returns the division setting value if the setting exists, else the default value. Ex.) `<visibleexpression>DivisionSetting('EnableFacebookButton', true)</visibleexpression>`

Please see the below example, `DivisionSetting` function in the expression engine evaluates and sets the visibility based on the setting with the name `EnableFacebookButton`. The default is set with the second parameter. which is `true` in the below case

```
<applicationextension application="CRMAccount.aspx">
    <button caption="Enable Facebook" id="ButtonEnableFacebook"
existing="false">
               <visibleexpression>DivisionSetting('EnableFacebookButton',
true)</visibleexpression>
        </button>
</applicationextension>
```

Visible Expression

The `<visibleexpression>` is supported by all UI elements (`button`, `monitoritem`, `cardsection`, `contentsectionrow`, `field`, `gridcolumn`, `gridpageheader`) of the `<applicationextensions>`.

VisibleExpression is an expression or formula, which is evaluated at runtime. The outcome of the expression is a boolean value, which determines whether the UI element is visible. In this expression you can calculate with values and dates, concate strings, use boolean operators like AND and OR, use brackets, and use functions and properties.

The properties that you can use in an expression, depend on the context. For example, in a VisibleExpression of a UI element in the account card, properties of the account business component can be used in expressions, e.g. IsCustomer or IsSupplier.

The functions that you can use in an expression are global, and include date functions (like Today(), Year(date), Date(year, month, day), etc.) and text functions like Concat. Furthermore you can use a function to retrieve a division setting.

In the below example, Field `FacebookId` is only visible if the setting `EnableFacebookField` is true.

```
<applicationextension application="CRMAccount.aspx">
    <cardsection id="csGeneral" existing="true">
        <field id="FacebookId" type="string" length="18" caption="Facebook
ID" translationid="FacebookIdTranslation" captionid="0" datasource="bc"
existing="false">
            <visibleexpression>DivisionSetting('EnableFacebookField',
true)</visibleexpression>
        </field>
    </cardsection>
</applicationextension>
```

Default Expression

In the `businesscomponentextensions` element, you can specify defaults for business component properties.

With a default expression, you specify a default value for a property of a business component. Default expressions have access to all property values of the business component and to all built-in functions and operators.

Expression defaults can be defined in the `functionalcomponent` element. This element has the following properties:

- `targetproperty`: property name of the business component, for which the default is specified
- `type`: ExpressionDefault
- `propertyStates`: one or more of AfterAddNew, OnValidate, BeforeInsert, BeforeUpdate, AfterEdit, AfterCopy, Overwrite. Use a space as a separator if you specify multiple values.

The property states determine when exactly the default expression is executed in the lifetime of the business component.

Defaults execute only if the property has no value yet, except if the Overwrite property state is specified. In that case, any existing property value is overwritten with the default.

In practice, we advise using the property state AfterAddNew for literal defaults, because they will be immediately visible in the Exact Online UI after the entry form is started.

```
<businesscomponentextensions>
    <businesscomponent name="Account">
               <!--Specify a default for the EMail property of the Account
business component. The default executes After AddNew.-->
               <functionalcomponent targetproperty="EMail"
type="ExpressionDefault" propertyStates="AfterAddNew">
                       <!--You can use literal strings, property names and
built-in functions in default expressions-->
                       <expression>Concat('www.', Name, '.com')</expression>
               </functionalcomponent>
        </businesscomponent>
</businesscomponentextensions>
```

Type transformations

Some of the custom extension definitions in the Extension Manifest XML file have a `type` attribute. Types are transformed into a type of control when the extension is loaded. Below you can use the table to see the mapping for the type transformations

Type	UI Element
integer	Number field
double	Number field supporting precision
string	Text field
boolean	Checkbox
date	Datetime control
Hyperlink (only supported by card fields)	HyperLink

Functional Validations with Azure Functions

Functional validations for a form can be customized by extending the validation logic with your own Azure Functions. Azure Function is a serverless solution that allows you to write less code, maintain less infrastructure, and save on costs. Instead of worrying about deploying and maintaining servers, the cloud infrastructure provides all the up-to-date resources needed to keep your applications running.

`<businesscomponent>` - Add custom validation for a business component.

    `name` - name of the business component the validations are extended.

`<azurevalidations>` - custom functional components validated using azure functions.

> `azurefunctionendpoint` - Endpoint to the azure function hosted in Azure.
>
> `azuresecretref` - API key to make communications with azure function are stored in a keyvault maintained by Exact. `azuresecretref` sets the name of the key in keyvault.
>
> `<functionalcomponent>`
>
> > `type` - Type of the functional component. Supported types are `AzureFunctionValidation` and `ComposedValidation`.
> >
> > `targetproperty` - Name of the property needed to be validated.
> >
> > `azureoperationname` - the name of the function responsible for doing the actual validation.

In the example below, Email field is validated when saving with an azure function.

```
<businesscomponentextensions>
        <businesscomponent name="Account">
            <azurevalidations
azurefunctionendpoint="http://emailvalidatorfunction.azurewebsites.net"
azuresecretref="EmailValidatorFunctionAPIKey">
                <functionalcomponent targetproperty="Email"
azureoperationname="ValidateEmail"/>
            </azurevalidations>
        </businesscomponent>
    </businesscomponentextensions>
```

Behind the scene when you hit the save on the page :

- POST request is performed on `http://emailvalidatorfunction.azurewebsites.net/api/ValidateEmail?code=<api key retrieved from the keyvault matching the reference defined in azuresecretref attribute>`.
- Body of the POST request will have the list of all the target properties and their values to validate.

```
[
    {
      "name":"Email",
      "displayValue":"joe@sampleemail.COM"
    },
    {
      "name":"Name",
      "displayValue":"Joe"
    }
]
```

Create an Azure Function from Template

You can find how to create an azure function from the template with this link : How to: Make custom validations with Azure Function

---

Validation Function Rule

Extension Manifest files when uploaded are validated to ensure compliance with Exact Online. Functional rules for the validation of an extension are divided into two main areas for each of the configuration sections.

- `Unique attributes validation` - ensures that identifier attributes for a given configuration element are not repeated in a specific way (e.g. the ids from all the tabs element in the same level, cannot be repeated) *****.
- `Content attributes validation` - ensures that an attribute is validated according to a specific rule set. (e.g. the id of a given element needs to be prefixed)

These rule sets are different per global type (main configuration element).

Entities	Unique Attributes (Cannot be repeated in the same level)	Attributes Content
Entity		• `ExtensionTable` needs to be prefixed with the solution code.
Property	• Name • ColumnName	• `Name` needs to be prefixed with the solution code.

Mega Menu	Unique Attributes (Cannot be repeated in the same level)	Attributes Content
MegaMenuExtension	• MenuId	N/A

Tab/Section/Subsection/Link	• Id	When existing='true', there are two validations 1. There must be a matching standard node (id value, hierarchy matching, node type). 2. Parent node, if any, should also specify existing='true'. Subsequent nodes will not be evaluated if the parent node validation fails.

Quick Menu	**Unique Attributes (Cannot be repeated in the same level)**	**Attributes Content**
Quick Menu	**Unique Attributes (Cannot be repeated in the same level)**	**Attributes Content**
QuickMenuExtension	• MenuId	N/A
Subsection/Link	• Id	When existing='true', there are two validations 1. There must be a matching standard node (id value, hierarchy matching, node type). 2. Parent node, if any, should also specify existing='true'. Subsequent nodes will not be evaluated if the parent node validation fails.

Applications	**Unique Attributes (Cannot be repeated in the**	**Attributes Conten**

	same level)	
ApplicationExtension	N/A (currently inactive)	N/A (currently inactive)
All		When existing='true then parent node, if any, should also specify existing='true'.
Button/CardSection/FilterBlock/ColumnGroup/MonitorItem/GridPageHeader	N/A (currently inactive)	N/A (currently inactive)
GridColumn	N/A (currently inactive)	All GridColumns that have an datasource attribute with !string.isnulloremp and not existing="true" mus be validated against an existing set of entities/properties elements and there needs to be at least **ONE** matching property name.
Field	N/A (currently inactive)	All Fields that have an datasource attribute with !string.isnulloremp and not existing="true" mus be validated against an existing set of entities/properties elements and there needs to be at least **ONE** matching property name. The name needs to be prefixed with the solution code.

Division Settings	Unique Attributes (Cannot be repeated in the same level)	Attributes Content
Tab	N/A	N/A
Section	• Id	When existing='true' then parent node should also specify existing='true'.
Setting	• Id	Id needs to be prefixed with the solution code.

Installing Extension

An extension created by following the steps in the earlier sections can be tested immediately when the extension file is uploaded and installed.
To install an extension do the following steps:

- Navigate to the extension version overview by going to Company → Master data → Developer: Maintain extensions
- Open the extension and the version to install
- Click on the `Install` button
- When the solution is installed, refresh the page and the extension is active

Publishing an extension to the Extensions catalogue

When an extension is created and verified, and tested by the extension developer it can be published so that other Exact Online Premium customers can install it from the Extensions catalogue. The best way to test the extension is to first Install it in your own environment.

Publishing extension functionality is available only in the packages with the feature set **PremiumExtensionsPublishing**.

Publish/Unpublish extension feature is only allowed for the users with the roles '**Manage extensions**' and '**Publish extensions**'.

To publish an extension:

- Navigate to the extension version overview by going to Company → Master data → Developer: Maintain extensions
- Open the extension and the version to publish
- Click on the `Publish` button

When the extension is uploaded and published, it is visible to all Exact Online Premium customers from the Extensions catalogue so that they can install it in their own Exact Online Premium environment.

The **Unpublish** button is used to unpublish the extension and it is enabled only if the extension is published.

Extensions catalogue

The extensions catalogue provides a list of available extensions, published by different Exact Online Premium customers, and ready to be installed in any Exact Online Premium environment.

`Master data > Catalogue | Extensions`

Enable or disable an extension

The extension is by default enabled, once installed. It is possible to disable the extension while keeping custom data and the extension installed. When an extension is disabled it is not rendered the Exact Online Premium. Later it is possible to enable the extension that was disabled.

To enable or disable an extension user must have the right '**Manage extensions**'.

Menu Path → Navigate to `Master data > Overview | Installed extensions`

Uninstall, downgrade or upgrade versions of the extension

- Uninstall extension

  This is used to uninstall the extension and it is no longer available in Exact Online Premium. While uninstalling we have the option to **Remove extension data** when the user selects this it will completely remove the extension data from Exact online. If the user unselects this option and tries to reinstall the same version next time the extension data still exists.

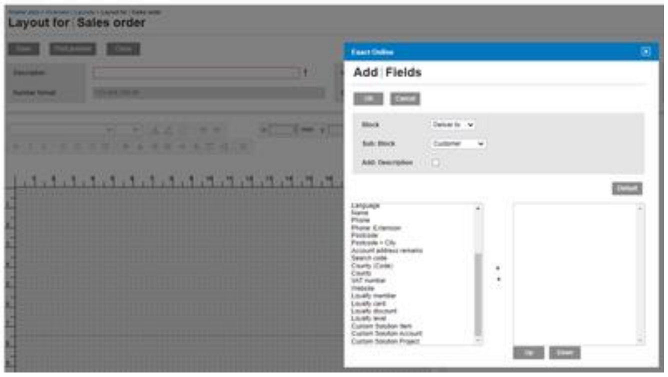
- Downgrade

  Installing the extension to an older version. The user can select and install the required old version.

- Upgrade

If the install version is not the latest version, the upgrade button is enabled which allows the user to upgrade to the latest version.

Adding custom data to the layout



When there is a new custom data field it can be easily added to the layout. For example, you have a new custom field ‘Custom Solution Project’ for Sales Order and you want to add it to the layout. First, open the layout for the Sales Order. Then drag and drop the area section from the Design tools and a section like the one below will appear. The new custom field is accessible at the bottom of the list with fields on the left since custom fields are always presented at the bottom of the first section.

Putting all the pieces together

Below is a sample extension making use of most of the supported features. This can be a starting point to play around and get yourself familiarized with extensions.

```
<?xml version="1.0" encoding="utf-8"?>

<!--
        - Extension code begins with the unique 3 letter code for the
extension developer. For ex : AUROR_TEST
        - Format for the code is <extension developer unique code>_<Name of
the extension>
        - Extension developer code ( for ex. AUROR_TEST ) must be in
uppercase
        - Version represents the version of the extension in semantic
versioning format. Please ref https://semver.org/ for more information
    -->
<extension code="AUROR_TEST" version="1.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

        xsi:noNamespaceSchemaLocation="C:\playground\exactonline\WebApp\Source
s\xml\Extensions.xsd">
    <entities>
```

```
        <!--
        - Adds properties and property relations for business components
        - Extension table and its columns are automatically created with the
defenition in XML
        - TwitterId column and LoyaltyLevel columns are added to newly
extending table SDK_ZA_Accounts
        - TwitterId and LoyaltyLevel properties are added to the properties
list of business component Account
        - listitems define possible list of enumerations for loyaltylevel and
internally the value is stored when saving in table
    -->
        <entity name="Account" table="Accounts"
extensiontable="AUROR_TEST_Social_Accounts" businesscomponent="Account">
            <property name="AUROR_TEST_CS1_TwitterID" columnname="TwitterId"
caption="Twitter Id" type="string" mandatoryfeatureset="VAT" />
            <property name="AUROR_TEST_CS1_LoyaltyLevel"
columnname="LoyaltyLevel" type="integer" caption="Loyalty level">
                <listitems allowempty="true"
listdefinitionid="LoyaltyLevelList">
                    <listitem value="1" caption="Bronze" captionid="0" />
                    <listitem value="3" caption="Gold" captionid="0"/>
                    <listitem value="2" caption="Silver" captionid="0"/>
                    <listitem value="4" caption="Platinum" captionid="0"/>
                </listitems>
            </property>
        </entity>
    </entities>

    <!--
        - Extends functional validations for business components
        - Validation logic is written in Azure function exposed with the
endpoint specified in azurefunctionendpoint
        - API key for the HTTP Triggered azure function is kept in Keyvault
maintained by Exact and a reference to the key is set here with
azuresecretref attribute
               - azuresecretref can only contain alphanumeric charecters and
'-'
        - Multiple functional components can use a same azure function
        - targetproperty for a functionalcomponent refers to the name of the
field to be validated
        - azureoperationname refers to the operation within the azure
function
    -->
    <!-- <businesscomponentextensions>
        <businesscomponent name="Account">
            <azurevalidations
azurefunctionendpoint="https://someazurefunction.microsoft.net"
azuresecretref="SecretApiKeyRef">
                <functionalcomponent targetproperty="Email"
azureoperationname="ValidateEmail">
                    <documentation>Validates email is from
gmail</documentation>
                </functionalcomponent>
            </azurevalidations>
        </businesscomponent>
    </businesscomponentextensions> -->
    <!--
        - Can add a new tab, section, subsection or links for a megamenu
        - Can also extend an existing tab,section or subsections
        - In the example below, a new link Twitter is created in Dashboards
menu. This link is only available for users under Netherlands negislation (
mandatorylegislation )
    -->
    <megamenuextensions>
        <megamenuextension menuid="MegaMenu">
            <tab id="Dashboards" existing="true">
                <section id="Dashboards" existing="true">
                    <subsection id="General" existing="true">
                        <link id="Twitter" caption="Twitter" captionid="0"
href="https://twitter.com" existing="false"showinnewtab="true" >

<mandatorylegislation>Netherlands</mandatorylegislation>
                        </link>
                    </subsection>
                </section>
            </tab>
        </megamenuextension>
    </megamenuextensions>
    <!--
        - Can add a new subsection or links for a quickmenu
        - Can also extend existing subsections
        - In the example below, a new link Twitter is created in Accounts
subsection. This link is only available for users under Netherlands
negislation ( mandatorylegislation )
    -->
    <quickmenuextensions>
        <quickmenuextension menuid="Wholesale">
            <subsection id="Accounts" existing="true">
                <link id="Twitter" caption="Twitter" captionid="0"
href="https://twitter.com"  existing="false" showinnewtab="true"/>
            </subsection>
            <!-- href also supports relative path. For example to an aspx
page within Exact online docs -->
            <subsection id="Entry" caption="Entry" captionid="0"
existing="false">
                <link id="FinInvoiceEntryLink" caption="Financial Invoice
Entry" captionid="0" href="FinInvoiceEntry.aspx"  existing="false"/>
            </subsection>
        </quickmenuextension>
    </quickmenuextensions>
    <!--
        - Add new or extend existing UI controls for aspx pages
    -->
    <applicationextensions>
        <applicationextension application="CRMAccountCard.aspx">
            <!--
                - New buttons in card page
                - Twitter button appears for Netherlands legislation and
hides facebook
                - Facebook button appears for belgium legislation and hides
Twitter
             -->
            <button href="https://twitter.com" showinnewtab="true"
captionid="0" caption="Twitter" id="btnTwitter" existing="false">
                <mandatorylegislation>Netherlands</mandatorylegislation>
            </button>
            <button href="https://www.facebook.com" showinnewtab="true"
captionid="0" caption="Facebook" id="btnFacebook" existing="false">
                <mandatorylegislation>Belgium</mandatorylegislation>
            </button>

            <!-- New MonitorItem -->
            <monitor id="QuickActionsMonitor" existing="true">
                <monitoritem id="MonitorId" caption="Loyalty History"
captionid="0" href="CflMatch.aspx" image="icon-customer-30.png"
existing="false"/>
            </monitor>

            <cardsection id="AccountInfoSection" existing="true">
                <!--Customize Standard Card Field-->
                <field existing="true" id="PhoneField">
                    <visibleexpression>false</visibleexpression>
                </field>

                <!--New Card Field-->
                <field id="MollieUrl" type="hyperlink"
href="www.mollie.com/en" showinnewtab="true" existing="false" />
            </cardsection>

            <contentsectionrow id="AccountSectionRow" existing="true">
                <!--New ContentSections -->
                <field id="LoyaltyLevel"
datafield="AUROR_TEST_CS1_LoyaltyLevel" type="integer" caption="Loyalty
Level"  controltype="radiobuttonlist"  readonly="true" datasource="bc"
existing="false"/>

                <!--
                 - In below example, VanumberCardField is an already existing
field. Caption of the field is overriden with the caption set here
                -->
                <field id="VatNumberCardField" existing="true" caption="Value
Tax" captionid="0"/>

                <!--
                 - In below example, CocNumberCardField is an already
existing field. Field is made invisible with the use of visible expression
                -->
                <field id="CocNumberCardField" existing="true" >
                    <visibleexpression>false</visibleexpression>
                </field>
            </contentsectionrow>
        </applicationextension>
        <applicationextension application="FinEntryBankCash.aspx">
            <!--
                - Add or modify existing fields in a grid page header section
             -->
                       <gridpageheader>
                               <section id="CustomHeader" existing="false">
```

```
                                      <field id="TwitterId"
datafield="AUROR_TEST_CS1_TwitterID" type="string" length="80"
caption="Twitter Profile"  captionid="0" datasource="grd" existing="false" />
                              </section>
                       </gridpageheader>

            <!--
                - Add or modify existing columns in a grid page
                - Possible to position the column by setting positionbefore
attribute
             -->
                       <gridcolumn id="TwitterId"
datafield="AUROR_TEST_CS1_TwitterID" type="string" length="34" width="15%"
caption="Twitter Profile" captionid="0" datasource="grd"
positionbefore="colAccount" existing="false"/>
               </applicationextension>
    </applicationextensions>

    <!--
        Extends division specific settings
    -->
    <divisionsettingsextensions>
        <!--
            Add new tab or extend existing tab
         -->
               <tab id="CustomSettings" caption="Custom Settings"
captionid="0" existing="false">
                       <section id="Social" caption="Social Media"
captionid="0" existing="false">
                              <!--
                   - Add new setting field with default value. Supported
types are boolean,string, integer, double, date and hyperlink
                -->

                <setting id="AUROR_TEST_CS1_EnableTwitter" caption="Enable
Twitter Field" captionid="0" type="boolean" defaultvalue="true"
translationid="AUROR_TEST_CS1_EnableTwitterFieldTranslation" />

                       </section>
               </tab>
        </divisionsettingsextensions>


    <translationextensions>
        <translation id="AUROR_TEST_CS1_EnableTwitterFieldTranslation">
            <language code="en-EN">Enable twitter field</language>
            <language code="nl-NL">Twitter veld inschakelen</language>
        </translation>
    </translationextensions>
</extension>
```