

Model-driven apps developer documentation

Model-driven apps are primarily a no-code or low-code component focused approach to app development. Extend model-driven apps by applying business logic using client scripting, web resources. Customize entity forms, views, dashboards, chart, commands and ribbons.

Get started

OVERVIEW

[Developer overview](#)

[Developers: Get started](#)

TRAINING

[Extend model-driven apps using code](#)

Customize

OVERVIEW

[Customize entity forms](#)

[Customize entity views](#)

[Customize visualizations and dashboards](#)

[Customize commands and ribbons](#)

Apply business logic

OVERVIEW

[Client scripting](#)

[Understand the Client API object model](#)

TUTORIAL

[Write your first client script](#)

Work with Web resources

OVERVIEW

[Web resources](#)

[Script web resource development using Fiddler AutoResponder](#)

[Web resource dependencies](#)

SAMPLE

[Pass multiple values to a web resource](#)

[Import files as web resources](#)

Build custom code components

OVERVIEW

[Power Apps component framework](#)

TUTORIAL

[Create your first code component](#)

TRAINING

[Create components with Power Apps Component Framework](#)

Reference

REFERENCE

[Client API reference](#)

[Customization XML reference](#)

[Form XML schema](#)

[Power Apps component framework API reference](#)

[Power Apps component framework Manifest schema reference](#)

Related resources

HOW-TO GUIDE

[Model-driven apps documentation](#)

[Microsoft Dataverse developer documentation](#)

[Canvas apps developer documentation](#)

Model-driven apps Developer Guide

Article • 11/30/2022

Model-driven apps are primarily a no-code or low-code component focused approach to app development. The value developers can provide is by extending the application. Before you start writing code, begin with [learning how to build model-driven apps](#) and what options can be applied without code.

Note

Model-driven apps connect to Microsoft Dataverse. For information about how developers can add value at the service level, see [Dataverse Developer Overview](#).

Content in this section will refer only to extensions developers can do that apply to the experience for users of model-driven apps.

If you are new to the Dataverse applications, the topics in this section provides a high-level overview of the important concepts to help developers get started working with model-driven apps.

[Get started](#)

Related topics

[Understand model-driven apps components](#)

[Create your first model-driven app](#)

Get started with Model-driven apps customization using code

Article • 11/30/2022

You can customize model-driven apps by using tools that are available in Power Apps or that are described in the documentation. These customizations are supported and can be upgraded.

Customizations made using methods other than those described here are unsupported and could cause problems during updates and upgrades to model-driven apps. For more information, see [Unsupported customizations](#) later in this topic.

Topics covered in technical articles published on Microsoft sites such as this one are supported, but might not be upgradable.

Customizations using Power Apps

There are a variety of tools included with model-driven apps that you can use to customize them. Customizations made using the model-driven apps tools are fully supported and fully upgradeable.

The following customization methods can be used to produce fully supported customizations:

- Customization in Power Apps or solution explorer. For more information, see [Overview of building model-driven apps](#)
- Settings in the web application. For more information, see [Administer model-driven apps](#).
- Reporting Services. For more information, see [Reporting and analytics guide for model-driven apps](#).

Note

The behavior of model-driven apps depend on customizations applied to the associated Microsoft Dataverse. More information: [Supported customizations for Dataverse](#) *Fully supported* means that developer support can provide assistance for customizations and that application support can help customers running those modifications.

Customizations applied using code

The documentation on this site for developers, technical articles, sample code published on this site, and information released by the Dataverse Developer Support Team are included in the area of customizations applied using code. The specific actions and levels of supportability and upgradeability are described later in this topic.

Client-side JavaScript

You can use JavaScript within model-driven apps in three areas:

- **Form Script event handlers:** You can configure form event handlers to call functions defined in JavaScript web resources.
- **Command bar (ribbon) commands:** You can use the `<CustomRule>` or `<JavaScriptFunction>` elements to define actions that call functions defined within JavaScript web resources.
- **Web resources and IFRAMES:** You can use JavaScript web resources within HTML web resources. IFRAMES configured to allow cross-site scripting, or scripts within HTML web resources included in a form may interact with the documented `Xrm.Page` or `Xrm.Utility` methods within the form via the parent reference.

All interaction with the application pages must only be performed through the methods documented in the [Client API Reference for model-driven apps](#). Directly accessing the Document Object Model (DOM) of any Model-driven apps page is not supported. The use of jQuery in form scripts and commands is not recommended. More information: [Client scripting in model-driven apps using JavaScript](#).

You can open model-driven apps forms, views, dialogs, and reports using the methods documented in [Open forms, views, dialogs, and reports with a URL](#).

Ribbon customization

Use of `RibbonDiffXml` to add, remove, or hide ribbon elements is supported. Reuse of ribbon commands defined by Model-driven apps is supported; however, we reserve the right to change or deprecate the available commands. Reuse of JavaScript functions defined within ribbon commands is not supported.

Unsupported customizations

Modifications to Model-driven apps that are made without using either the methods described in this documentation or Power Apps tools are not supported and are not preserved during updates or upgrades of model-driven apps. Anything that is not documented in this documentation and supporting documents is not supported. Additionally, unsupported modifications could cause problems when you update through the addition of hot fixes or service packs or upgrade model-driven apps.

The following is a list of unsupported action types that are frequently asked about:

- The reuse of any model-driven apps JavaScript code. This code may change or be overwritten during an upgrade.
- Editing a solutions file to edit any solution components other than ribbons, forms, SiteMap, or saved queries is not supported. For more information, see [When to edit the customizations file](#).
 - Defining new solution components by editing the solutions file is not supported.
 - Editing web resource files exported with a solution is not supported.
 - Except for the steps documented in [Maintain managed solutions](#), editing the contents of a managed solution is not supported.
- Displaying a form within an IFrame embedded in another form is not supported.

See also

[Supported customizations for Dataverse](#)

[Apply business logic using client scripting in model-driven apps using JavaScript](#)

[Customize commands and the ribbon](#)

[Web resources in model-driven apps](#)

Customize forms

Article • 11/30/2022

Forms provide the user interface (UI) that people use to create, view, or edit table records. Use the form designer in the customization tools to create and edit forms. More information: [Create and design forms](#) for information about tasks related to working with forms in the application.

This topic will provide information necessary to create or edit forms programmatically.

Access form definitions

Forms are stored in the `SystemForm` table along with dashboards and visualizations.

There are two ways that you can inspect the form definitions for a table:

- Include the table in an unmanaged solution and export the solution.
- Query the `SystemForm` table

View FormXML from a exported table

Only definitions of system forms that have been customized are included in exported managed solution. To view the definition of a system form, you must either change it in some way, or create a new form by saving the existing form with a new name.

After you export the solution, extract the contents and view the `customizations.xml` file. You'll find the definition of the forms in `ImportExportXml > Entities > Entity > FormXml1`. In the `<FormXml1>` node, you'll find each type of form is grouped in a `<forms>` element with the `type` parameter specifying the type of form.

Form properties

The following table describes key `SystemForm` table columns and the corresponding data included in the XML elements exported with the solution.

<code>SystemForm</code> property	<code>FormXML</code> element	Description
-------------------------------------	------------------------------	-------------

SystemForm property	FormXML element	Description
AncestorFormId	<ancestor>	Unique identifier of the parent form. This is set when you create a new form by using Save As for an existing form or by using CopySystemFormRequest .
CanBeDeleted	<CanBeDeleted>	Information that specifies whether this component can be deleted. This managed property is only applied if the form was created by importing a managed solution.
Description	<Descriptions>	<p><code>Description</code> is a string and <code><Descriptions></code> contains any localized labels for the description of the form.</p> <p>The localized labels can be retrieved using the RetrieveLocLabelsRequest.</p>
FormActivationState	<FormActivationState>	<p>Specifies the state of the form.</p> <p>Only forms of type "main" can be deactivated.</p> <p>Valid Values:</p> <ul style="list-style-type: none"> - 0: Inactive - 1: Active
FormId	<formid>	Unique identifier of the form
FormPresentation	<FormPresentation>	Specifies whether this form is in the updated UI layout in Microsoft Dataverse.
FormXml	<form>	XML representation of the form layout.
IntroducedVersion	<IntroducedVersion>	Version of the solution that the form was added in.
IsAIRMerged	N/A	Specifies whether this form is merged with the updated UI layout in Dataverse.
IsCustomizable	<IsCustomizable>	<p>Information that specifies whether this component can be customized.</p> <p>This managed property is only applied if the form was created by importing a managed solution.</p>
IsDefault	N/A	Information that specifies whether the form or the dashboard is the system default.

SystemForm property	FormXML element	Description
Name	<LocalizedNames>	<p><code>Name</code> is a string and <code><LocalizedNames></code> contains any localized labels for the name of the form.</p> <p>The localized labels can be retrieved using the RetrieveLocLabelsRequest.</p>
ObjectTypeCode	The form is a decedent of the <code>Entity</code> element.	The <code>ObjectTypeCode</code> value is the table logical name.
Type	<forms> element <code>type</code> parameter	<p>Valid values for forms are:</p> <ul style="list-style-type: none"> - 2: <code>main</code> - 5: <code>mobile</code> - 6: <code>quick</code> - 7: <code>quickCreate</code>

Create and edit forms

You can only create new forms for a table where [EntityMetadata.CanCreateForms](#) allows it.

You can create new forms using either a [CreateRequest](#) or the [CopySystemFormRequest](#). When using [CopySystemFormRequest](#) or using **Save As** in the form editor, note that there is no inheritance between forms. Therefore, changes to the base form aren't automatically applied to any forms created from it.

Editing the form definitions from an exported managed solution and then re-importing the solution is a supported method to edit forms. When manually editing forms we strongly recommend you use an XML editor that allows for schema validation. More information: [Edit the customizations XML file with schema validation](#)

Open main form in a dialog using client API

To open the main form in a dialog using client API, you need to invoke the call using the `Xrm.Navigation.navigateTo` method. The `Xrm.Navigation.navigateTo` API method allows you to open the dialog with several options, including the size and position.

Note

Xrm.Navigation.openForm method is not supported to open a main form as a dialog.

Examples

Open a new record

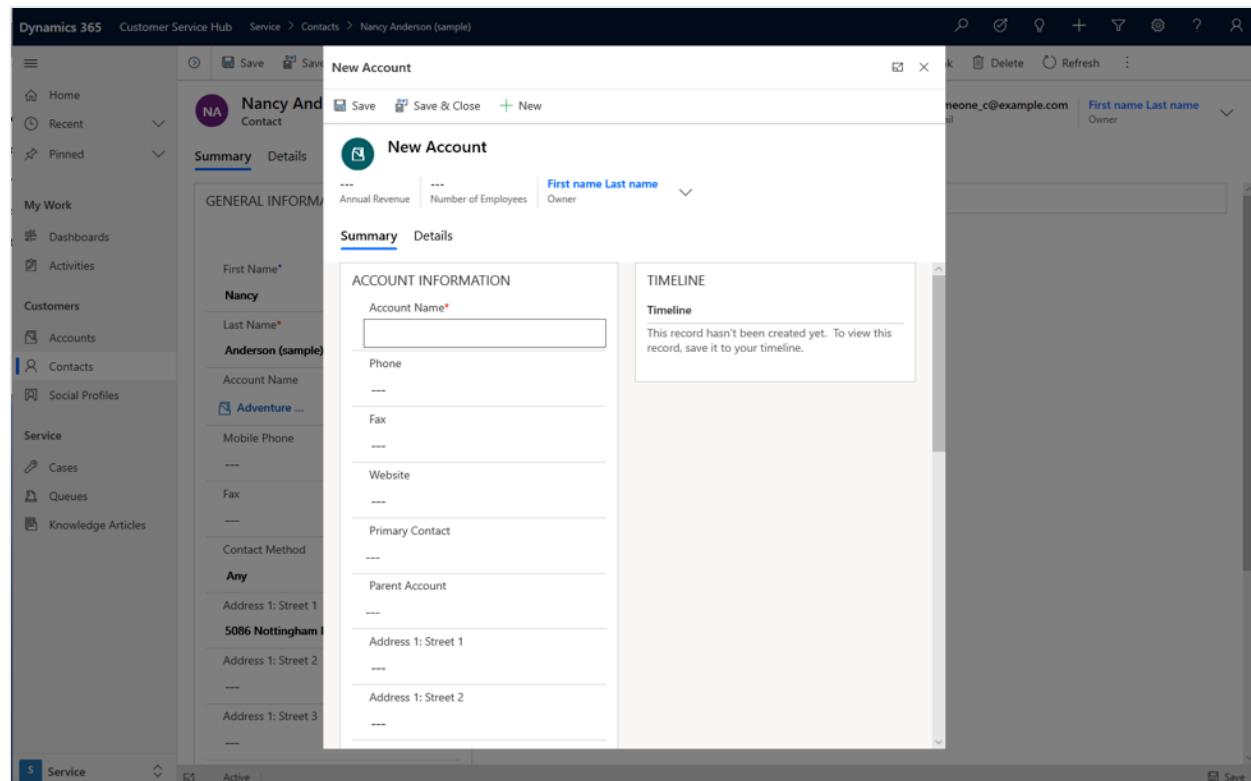
In this example, the dialog opens a new account form for creating a new record. The dialog pops up in the center using up to 50% of the available window as a modal on top of the form it was invoked or called from.

JavaScript

```
var pageInput = {
    pageType: "entityrecord",
    entityName: "account",
    formType: 2,
};

var navigationOptions = {
    target: 2,
    width: {value: 50, unit:"%"},
    position: 1
};

Xrm.Navigation.navigateTo(pageInput, navigationOptions);
```

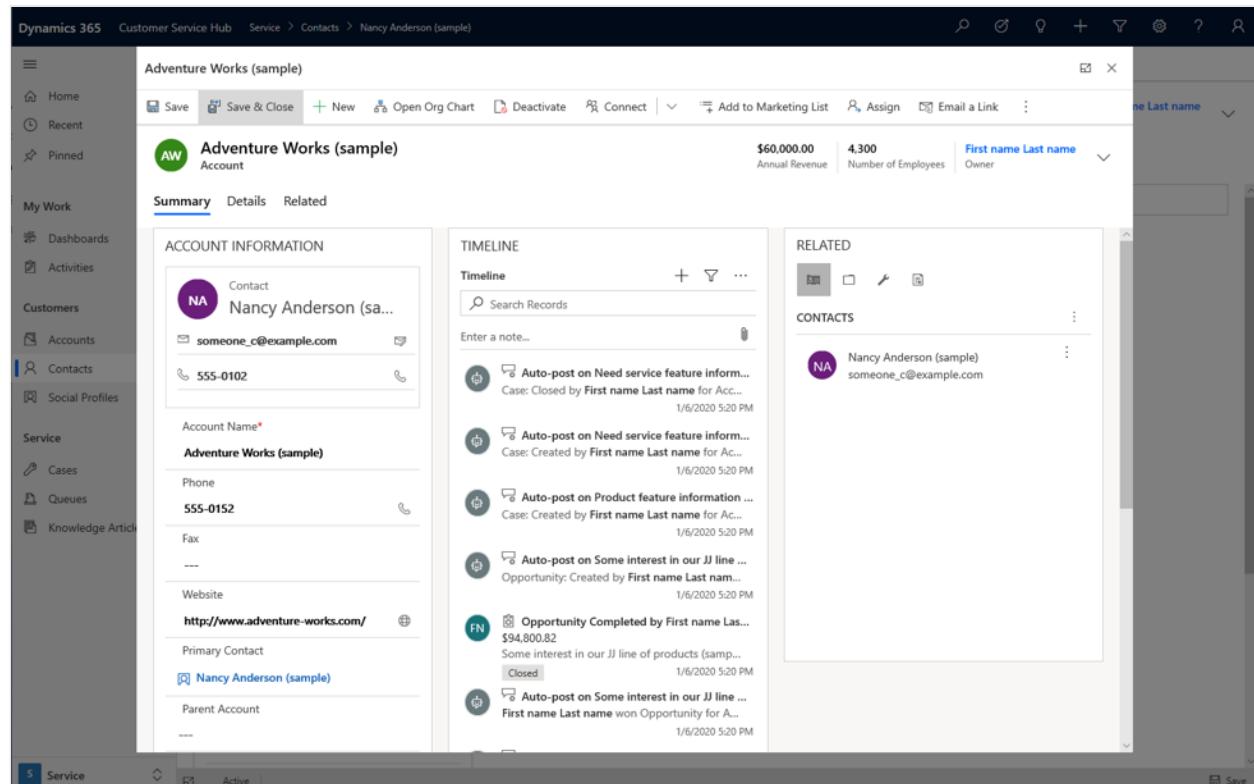


Open an existing record

In this example, the dialog opens an existing account record using the account id value over the contact form. Replace the id with any record id value you want to open the record in the dialog.

JavaScript

```
var pageInput = {
    pageType: "entityrecord",
    entityName: "account",
    formType: 2,
    entityId: "5a57f2c3-5672-ea11-a812-000d3a339706" //replace with actual
ID
};
var navigationOptions = {
    target: 2,
    width: {value: 80, unit:"%"},
    position: 1
};
Xrm.Navigation.navigateTo(pageInput, navigationOptions);
```



Open a new record on the side pane

In this example, the dialog opens a new record in the right corner of the window. This can be achieved by using the pixel options.

JavaScript

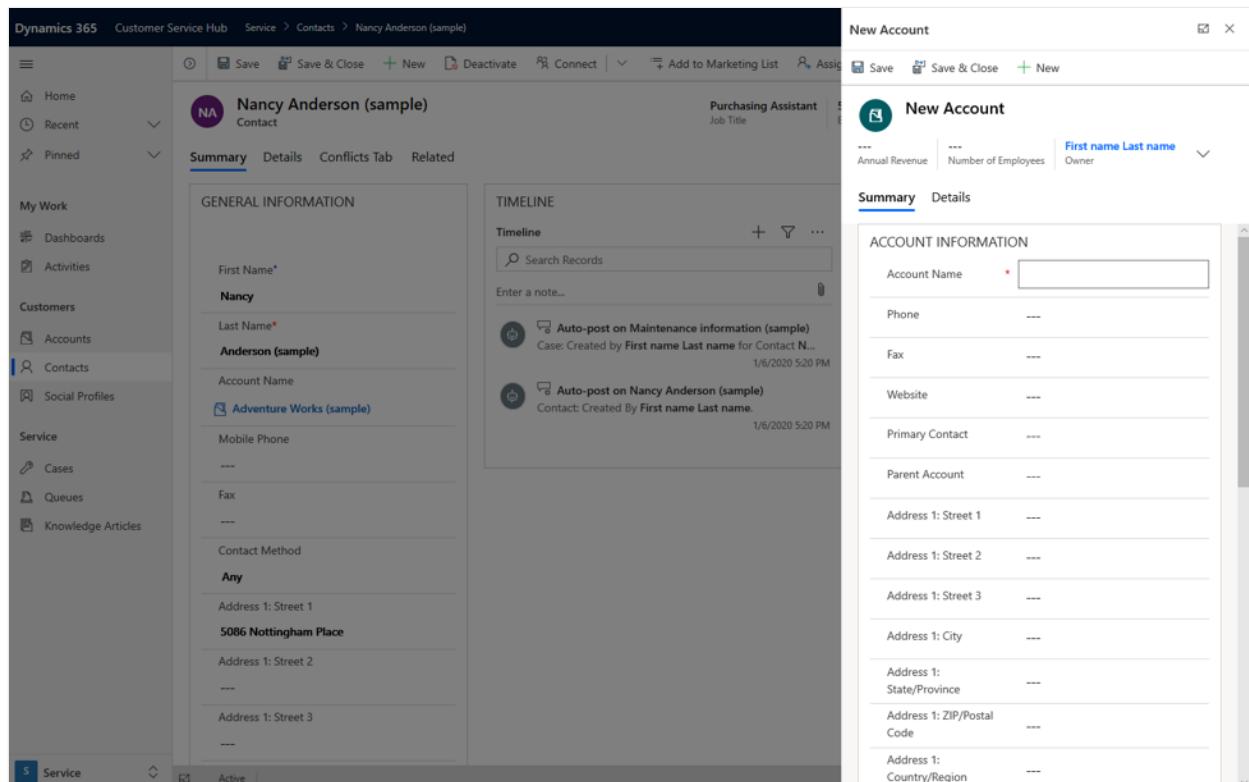
```

var pageInput = {
    pageType: "entityrecord",
    entityName: "account",
    formType: 2,
};

var navigationOptions = {
    target: 2,
    width: {value: 500, unit:"px"},
    position: 2
};

Xrm.Navigation.navigateTo(pageInput, navigationOptions);

```



Open main form in a dialog with callback method

This example shows how a main form dialog is invoked with a callback method after saving a record and closing the dialog.

Javascript

```

var pageInput = {
    pageType: "entityrecord",
    entityName: "account",
    formType: 2
};

var navigationOptions = {
    target: 2,
    width: {value: 80, unit:"%"},
    position: 2
};

```

```
Xrm.Navigation.navigateTo(pageInput, navigationOptions).then(
    function success(result) {
        console.log("Record created with ID: " +
result.savedEntityReference[0].id +
            " Name: " + result.savedEntityReference[0].name)
        // Handle dialog closed
    },
    function error() {
        // Handle errors
    }
);
```

See also

[Create and design forms](#)

[SystemForm table](#)

[Form XML schema](#)

[Xrm.Navigation.navigateTo](#)

Use IFRAME and web resource controls on a form

Article • 12/16/2022

IFRAME and web resource controls embed content from another location in pages by using an HTML IFRAME element.

Note

The designs you choose for the form are also used for the Dynamics 365 for Outlook reading pane and forms used by Dynamics 365 tablets. Web resources and IFRAMES aren't displayed using the Dynamics 365 for Outlook reading pane, however, they are supported in Dynamics 365 for tablets. If your IFRAME depends on access to the `Xrm` object of the page or any form event handlers, you should configure the IFRAME so that it's not visible by default.

You can use an IFRAME to display the contents from another website in a form, for example, in an ASP.NET page.

It is recommended to use [Power Apps component framework components](#) if you're considering to use a web resource to show content that users will interact with.

Displaying a form within an IFrame embedded in another form is not supported.

You can use one of the following web resources to display the contents of web resources in a form:

- [Web Page \(HTML\) web resources](#)
- [Image \(JPG, PNG, GIF, ICO\) web resources](#)

The following sections describe your options if you want these controls to show more than static content.

Select whether to restrict cross-frame scripting

Use the **Restrict cross-frame scripting, where supported** option when you don't fully trust the content displayed in an IFRAME. When this option is selected, the IFRAME has the parameters set that are listed in the following table.

Parameter	Description
<code>security="restricted"</code>	This parameter is no longer supported.
<code>sandbox=""</code>	<p>For browsers that support this parameter, the content in the IFRAME is essentially limited to only displaying information. The following restrictions could be applied:</p> <ul style="list-style-type: none"> - Browser plug-ins are disabled. - Forms and scripts are disabled. - Links to other browsing contexts are disabled. - Content is treated as from a different domain even if the domain is the same. <p>This parameter is defined by W3C and is supported by the following browsers:</p> <ul style="list-style-type: none"> - Microsoft Edge - Google Chrome - Apple Safari - Mozilla Firefox <p>For more information about the sandbox parameter see:</p> <ul style="list-style-type: none"> - How to safeguard your site with HTML5 sandbox - Sandbox

Enabling IFrame communication across domains

There are times when you want to enable communication for an IFRAME that contains content on a different domain. `Window.postMessage` is a browser method that provides this capability for Google Chrome, Mozilla Firefox, and Apple Safari. For more information about using `postMessage`, see the following blog posts:

- [Cross domain calls to the parent form ↗](#)
- [Cross-Document messaging and RPC](#)

Pass contextual information about the record

You can provide contextual information by passing parameters to the URL defined in the control. The page that is displayed in the frame must be able to process parameters passed to it. All the parameters in the following table are passed if the IFRAME or web resource is configured by using the **Pass record object-type code and unique identifier as parameters** option.

You can specify whether all the parameters in the following table will be passed.

Parameter	Name	Description
typename	Table Name	The name of the table.
type	Table Type Code	The integer that uniquely identifies the table in a specific organization.
id	Object GUID	A GUID that represents a record.
orgname	Organization Name	The unique name of the organization.
userlcid	User Language Code	The language code identifier that is being used by the current user.

Language codes are four-digit or five-digit locale IDs. Valid locale ID values can be found at [Locale ID \(LCID\) Chart](#).

① Note

We suggest that you use the table name instead of the type code because the table type code for custom tables may be different between Microsoft Dataverse organizations.

Example

The following sample shows the URL without parameters.

```
https://myserver/mypage.aspx
```

The following sample shows the URL with parameters.

```
https://myserver/mypage.aspx?id=%7bB2232821-A775-DF11-8DD1-00155DBA3809%7d&orglcid=1033&orgname=adventureworkscycle&type=1&typename=account&userlcid=1033
```

Read passed parameters

Passed parameters are typically read in the target .aspx page by using the `HttpRequest.QueryString` property. In an HTML page, the parameters can be accessed by using the `window.location.search` property in JavaScript. For more information, see [HttpRequest.QueryString Property](#) and [search Property](#).

Pass form data

Use the `getValue` method on the columns that contain the data that you want to pass to the other website, and compose a string of the query string arguments the other page will be able to use. Then use a [Column OnChange event](#), [IFRAME OnReadyStateComplete event](#), or [Tab TabStateChange event](#) and the `setSrc` method to append your parameters to the `src` property of the IFRAME or web resource.

If you're using the `data` parameter to pass data to a Silverlight web resource, you can use the `getData` and `setData` methods to manipulate the value passed via the `data` parameter. For webpage (HTML) web resources, use the `setSrc` method to manipulate the `queryString` parameter directly.

Avoid using the [OnLoad event](#). IFRAMES and web resources load asynchronously and the frame may not have finished loading before the `onload` event script finishes. This can cause the `src` property of the IFRAME or web resource you have changed to be overwritten by the default value of the IFRAME or web resource URL property.

Change the URL

You may want to change the target of the IFRAME based on such considerations as the data in the form or whether the user is working offline. You can set the target of the IFRAME dynamically.

Note

When you change the target page for the IFRAME, parameters aren't passed to the new URL automatically. You must append the query string parameters to the URL before you use the `setSrc` method.

Example

The following sample shows you how to set the `src` property for the IFRAME and any parameters by using the `onChange` event of a choice column.

JavaScript

```
//Get the value of an option set attribute
var formContext = executionContext.getFormContext();
var value = formContext.getAttribute("new_pagechooser").getValue();
var newTarget = "";
//Set the target based on the value of the option set
switch (value) {
    case 100000001:
        newTarget = https://myServer/test/pageOne.aspx;
        break;
    default:
        newTarget = https://myServer/test/pageTwo.aspx;
        break;
}
//Get the default URL for the IFRAME, which includes the
// query string parameters
var IFrame = formContext.ui.controls.get("IFRAME_test");
var Url = IFrame.getSrc();
// Capture the parameters
var params = Url.substr(Url.indexOf("?"));
//Append the parameters to the new page URL
newTarget = newTarget + params;
// Use the setSrc method so that the IFRAME uses the
// new page with the existing parameters
IFrame.setSrc(newTarget);
```

See Also

[Client scripting using JavaScript](#)

Customize views

Article • 12/16/2022

Views are special saved queries that retrieve data by using a specific filter. They also contain information about how the data in the view should be displayed in the application. Views are `SavedQuery` records that you can create programmatically. You can also define them as XML, and import them with an unmanaged solution.

A `SavedQuery` view is different from a `UserQuery`. A user query, called a Saved view in the application, is owned by an individual user, can be assigned and shared with other users, and can be viewed by other users depending on the query's access privileges. This is appropriate for frequently used queries that span table types and queries that perform aggregation. More information: [Saved queries](#)

You can also use the customization tool to customize views. More information: [Create and edit views](#)

Types of views

The following table lists the five types of views that are supported for customization. The type code of a view is stored in the `SavedQuery.QueryType` parameter.

When views are defined for a specific table, the `SavedQuery.ReturnedTypeCode` parameter returns the table logical name.

View Type Code	Type Code	Description
Public	0	<ul style="list-style-type: none">- Occurrence: Many- Actions: Create, Update, Delete- Comments: You can set one of these views as the default public view by setting <code>SavedQuery.IsDefault</code> to true.
Advanced Find	1	<ul style="list-style-type: none">- Occurrence: 1- Actions: Update only.- Comments: By default, this view is displayed when results are shown in Advanced Find.
Associated	2	<ul style="list-style-type: none">- Occurrence: 1- Actions: Update only,- Comments: By default, this view is displayed when a grid of related records appears in the navigation pane of a record.

View Type	Type	Description
Code		
Quick Find	4	<ul style="list-style-type: none"> - Occurrence: 1 - Actions: Update only. - Comments: This view defines the columns that will be searched when a user searches for records by using the search column in a list view.
Lookup	64	<ul style="list-style-type: none"> - Occurrence: 1 - Actions: Update only. - Comments: This is the default view that will be used to look up a record when no other view has been configured for the lookup column.

Create views

To create a public view, specify the following properties:

- `SavedQuery.Name`: A unique identifier for the saved query.
- `SavedQuery.ReturnedTypeCode`: Matches the logical name of the table.
- `SavedQuery.FetchXml`: See [Use FetchXML to construct a query](#).
- `SavedQuery.LayoutXml`: See the `layoutxml` element in the [Customization solutions file schema](#) for the valid elements.
- `SavedQuery.QueryType`: Must always be zero (0).

The following sample creates a new public view for the opportunity:

C#

```
System.String layoutXml =
@"
<grid name='resultset' object='3' jump='name' select='1'
      preview='1' icon='1'>
  <row name='result' id='opportunityid'>
    <cell name='name' width='150' />
    <cell name='customerid' width='150' />
    <cell name='estimatedclosedate' width='150' />
    <cell name='estimatedvalue' width='150' />
    <cell name='closeprobability' width='150' />
    <cell name='opportunityratingcode' width='150' />
    <cell name='opportunitycustomeridcontactcontactid.emailaddress1'
          width='150' disableSorting='1' />
  </row>
</grid>";

System.String fetchXml =
@"
<fetch version='1.0' output-format='xml-platform'
      >
```

```

mapping='logical' distinct='false'
<entity name='opportunity'>
<order attribute='estimatedvalue' descending='false' />
<filter type='and'>
    <condition attribute='statecode' operator='eq'
        value='0' />
</filter>
<attribute name='name' />
<attribute name='estimatedvalue' />
<attribute name='estimatedclosedate' />
<attribute name='customerid' />
<attribute name='opportunityratingcode' />
<attribute name='closeprobability' />
<link-entity alias='opportunitycustomeridcontactcontactid'
    name='contact' from='contactid' to='customerid'
    link-type='outer' visible='false'>
    <attribute name='emailaddress1' />
</link-entity>
<attribute name='opportunityid' />
</entity>
</fetch>";

var sq = new SavedQuery
{
    Name = "A New Custom Public View",
    Description = "A Saved Query created in code",
    ReturnedTypeCode = "opportunity",
    FetchXml = fetchXml,
    LayoutXml = layoutXml,
    QueryType = 0
};
_customViewId = service.Create(sq);
Console.WriteLine("A new view with the name {0} was created.", sq.Name);

```

Update views

If the `SavedQuery.IsCustomizable` managed property allows the view to be updated, you can use the `IOrganizationService.Update` method or the `UpdateRequest` message to update the view.

Delete views

You should only delete saved queries that you have created. A solution component or part of the application may depend on a specific saved query. If there are queries you do not want to appear in the application, you should deactivate them.

Retrieve views

Use a [RetrieveMultipleRequest](#) or [IOrganizationService.RetrieveMultiple](#) to retrieve saved query records.

The following sample retrieves all the public views for the opportunity:

C#

```
var mySavedQuery = new QueryExpression
{
    ColumnSet = new ColumnSet("savedqueryid", "name", "querytype",
    "isdefault", "returnedtypecode", "isquickfindquery"),
    EntityName = SavedQuery.EntityLogicalName,
    Criteria = new FilterExpression
    {
        Conditions =
    {
        new ConditionExpression
        {
            AttributeName = "querytype",
            Operator = ConditionOperator.Equal,
            Values = {0}
        },
        new ConditionExpression
        {
            AttributeName = "returnedtypecode",
            Operator = ConditionOperator.Equal,
            Values = {Opportunity.EntityTypeCode}
        }
    }
    };
    RetrieveMultipleRequest retrieveSavedQueriesRequest = new
    RetrieveMultipleRequest { Query = mySavedQuery };

    RetrieveMultipleResponse retrieveSavedQueriesResponse =
    (RetrieveMultipleResponse)service.Execute(retrieveSavedQueriesRequest);

    DataCollection<Entity> savedQueries =
    retrieveSavedQueriesResponse.EntityCollection.Entities;

    //Display the Retrieved views
    foreach (Entity ent in savedQueries)
    {
        SavedQuery rsq = (SavedQuery)ent;
        Console.WriteLine("{0} : {1} : {2} : {3} : {4} : {5}",
rsq.SavedQueryId, rsq.Name, rsq.QueryType, rsq.IsDefault,
rsq.ReturnedTypeCode, rsq.IsQuickFindQuery);
    }
}
```

Deactivate views

If you do not want a public view to appear in the application, you can deactivate it. You cannot deactivate a public view that is set as the default view. The following sample deactivates the **Closed Opportunities in Current Fiscal Year** view for the Opportunity:

C#

```
System.String SavedQueryName = "Closed Opportunities in Current Fiscal
Year";
QueryExpression ClosedOpportunitiesViewQuery = new QueryExpression
{
    ColumnSet = new ColumnSet("savedqueryid", "statecode", "statuscode"),
    EntityName = SavedQuery.EntityLogicalName,
    Criteria = new FilterExpression
    {
        Conditions =
        {
            new ConditionExpression
            {
                AttributeName = "querytype",
                Operator = ConditionOperator.Equal,
                Values = {0}
            },
            new ConditionExpression
            {
                AttributeName = "returnedtypecode",
                Operator = ConditionOperator.Equal,
                Values = {Opportunity.EntityTypeCode}
            },
            new ConditionExpression
            {
                AttributeName = "name",
                Operator = ConditionOperator.Equal,
                Values = {SavedQueryName}
            }
        }
    }
};

RetrieveMultipleRequest retrieveOpportunitiesViewRequest = new
RetrieveMultipleRequest { Query = ClosedOpportunitiesViewQuery };

RetrieveMultipleResponse retrieveOpportunitiesViewResponse =
(RetrieveMultipleResponse)service.Execute(retrieveOpportunitiesViewRequest);

SavedQuery OpportunityView =
(SavedQuery)retrieveOpportunitiesViewResponse.EntityCollection.Entities[0];
_viewOriginalState = (SavedQueryState)OpportunityView.StateCode;
_viewOriginalStatus = OpportunityView.StatusCode;
```

```
SetStateRequest ssreq = new SetStateRequest
{
    EntityMoniker = new EntityReference(SavedQuery.EntityLogicalName,
(Guid)OpportunityView.SavedQueryId),
    State = new OptionSetValue((int)SavedQueryState.Inactive),
    Status = new OptionSetValue(2)
};
service.Execute(ssreq);
```

ⓘ Note

The view state, active or inactive, is not included with the view when it is added to a solution. Therefore, when the solution is imported into a target organization, the status will be set to active by default.

Edit filter criteria or configure sorting

To edit the filter or edit how the data is sorted, you must set the `SavedQuery.FetchXml` parameter. More information: [Use FetchXML to query data](#).

💡 Tip

If you are not familiar with FetchXML the following messages can be used to convert between QueryExpression and `FetchXML:QueryExpressionToFetchXmlRequest` and `FetchXmlToQueryExpressionRequest`.

Edit columns

The columns that you want to display in views can be taken from the table or related tables. For more information about how to specify the columns to display, see the `layoutxml` element in the [Customization solutions file schema](#).

Add custom icons with tooltip for a column

You can add custom icon with tooltip text to display in a column depending on the column value; you can also specify localized tooltip text. This can be done by adding the custom icons as image web resources in your instance and then using a JavaScript web resource to add JavaScript code for a column to display the icons depending on the column value.

Note

Adding custom icons with tooltip is supported only for the read-only grids; this feature isn't supported for the editable grids. For more information about editable grids, see [Use editable grids](#).

Two new parameters, `imageproviderwebresource` and `imageproviderfunctionname`, are added to the `cell` element of the `layoutxml` of `savedquery` that lets you specify the name of a web resource and a JavaScript function name to display custom icons and tooltip text for a column. The JavaScript code gets executed when the page loads.

You can also use the new **Web Resource** and **Function Name** in the **Column Properties** page while modifying the property of a column in a view definition to specify the web resource name and JavaScript function name.

The following sample code demonstrates how you can programmatically specify a web resource and a JavaScript function name for adding custom icons and tooltip for the `opportunityratingcode` column in `layoutxml`:

C#

```
System.String layoutXml =
@"
<grid name='resultset' object='3' jump='name' select='1'
      preview='1' icon='1'>
  <row name='result' id='opportunityid'>
    <cell name='name' width='150' />
    <cell name='customerid' width='150' />
    <cell name='estimatedclosedate' width='150' />
    <cell name='estimatedvalue' width='150' />
    <cell name='closeprobability' width='150' />
    <cell name='opportunityratingcode' width='150'
          imageproviderwebresource='new_SampleWebResource'
          imageproviderfunctionname='displayIconTooltip' />
    <cell name='opportunitycustomeridcontactcontactid.emailaddress1'
          width='150' disableSorting='1' />
  </row>
</grid>";
```

The JavaScript function for displaying custom icons and tooltip text expects the following two arguments: the entire row object specified in `layoutxml` and the calling user's Locale ID (LCID). The LCID parameter enables you to specify tooltip text for the icon in multiple languages. For more information about the languages supported, see [Enable additional languages](#) and [Install or upgrade language packs](#). For a list of locale ID (LCID) values that you can use in your code, see [Locale IDs assigned by Microsoft](#).

Assuming you will most likely be adding custom icons for a choice type of column as it has a limited set of predefined options, make sure you use the integer value of the options instead of label to avoid breaking the code due to changes in the localized label string. Also, in your JavaScript function, specify just the name of an image web resource that you want to use as an icon for a value in the column. The image should be of 16x16 pixels size; larger images will be automatically scaled down to 16x16 pixels size.

The following sample code displays different icons and tooltip text based on one of the values (1: Hot, 2: Warm, 3: Cold) in the `opportunityratingcode` (Rating) column. The sample code also shows how to display localized tooltip text. For this sample to work, you must create three image web resources each with 16x16 images (red, yellow, and blue) in your instance with the following names respectively: `new_Hot`, `new_Warm`, and `new_Cold`.

JavaScript

```
function displayIconTooltip(rowData, userLCID) {
    var str = JSON.parse(rowData);
    var coldata = str.opportunityratingcode_Value;
    var imgName = "";
    var tooltip = "";
    switch (coldata) {
        case 1:
            imgName = "new_Hot";
            switch (userLCID) {
                case 1036:
                    tooltip = "French: Opportunity is Hot";
                    break;
                default:
                    tooltip = "Opportunity is Hot";
                    break;
            }
            break;
        case 2:
            imgName = "new_Warm";
            switch (userLCID) {
                case 1036:
                    tooltip = "French: Opportunity is Warm";
                    break;
                default:
                    tooltip = "Opportunity is Warm";
                    break;
            }
            break;
        case 3:
            imgName = "new_Cold";
            switch (userLCID) {
                case 1036:
                    tooltip = "French: Opportunity is Cold";
                    break;
                default:
```

```

        tooltip = "Opportunity is Cold";
        break;
    }
    break;
default:
    imgName = "";
    tooltip = "";
    break;
}
var resultarray = [imgName, tooltip];
return resultarray;
}

```

This results in displaying the values in the `Rating` column with appropriate icons depending on the value, and icon tooltip text when you hover over the icons.

► Sample: My Open Opportunities ▾

Topic	Actual Revenue	Rating	Contact	Est. Close Date	Est. Revenue
Very likely will order 73 Pr...	\$153,385.37	Cold	Rene Valdes (sa...	4/9/2016	\$16,000.00
Some interest in our JJ line...	\$94,800.82	Warm	Nancy Anderson...	7/29/2016	\$95,000.00
Very interested in our prod...	\$40,201.49	Hot	Scott Konersma...	7/6/2016	\$40,000.00
10 orders of Product SKU J...	\$22,469.39	Hot	Yvonne McKay (...	7/27/2016	\$22,000.00
10 orders or Product SKU ...	\$14,589.45	Warm	Susanna Stubbe...	6/9/2016	\$15,000.00

Set as default

Only one active public view can be set as the default view. To make a view the default view, set the `IsDefault` property to true.

Open apps, forms, views, dialogs, and reports with a URL

Article • 11/30/2022

URL addressable elements enable you to include links to apps, forms, views, dialogs, and reports in other applications.

ⓘ Note

URL addressable apps, forms, views, dialogs, and reports cannot bypass the security. Only licensed users, based on their security roles, can access the data and the records they see.

AppUrls

ⓘ Note

Embedding a model-driven application within an IFrame in another application is not supported.

You can open any model-driven application using the [AppModule.UniqueName](#) or [AppModule.AppModuleId](#) values.

You can retrieve these values using Web API using the following query:

HTTP

```
GET [Organization URI]/api/data/v9.1/appmodules?  
$select=appmoduleid,uniqueName
```

More information: [Query data using the Web API](#)

You can use either the `appname` or `appid` query parameters with the Unique Name or `AppModuleId` values respectively, but you cannot use both at the same time.

Using Unique Name

Append the `appname` query parameter to the `main.aspx` page to open the app using the Unique Name.

```
https://myorg.crm.dynamics.com/main.aspx?appname={UniqueName}
```

For example, if the Unique Name is `msdyn_SolutionHealthHub`, you can open this app using this URL:

```
https://myorg.crm.dynamics.com/main.aspx?appname=msdyn_SolutionHealthHub
```

Using AppModuleId

Append the `appid` query parameter to the `main.aspx` page to open the app using the AppModuleId.

```
https://myorg.crm.dynamics.com/main.aspx?appid={AppModuleId}
```

For example:

```
https://myorg.crm.dynamics.com/main.aspx?appid=12fd1cf3-e06e-e911-a95f-000d3a13c42a
```

URL addressable forms and views

All forms and views are displayed in the `main.aspx` page. Query string parameters passed to this page control what will be displayed. For example:

To open an account record form for where the id is `{91330924-802A-4B0D-A900-34FD9D790829}`:

```
https://myorg.crm.dynamics.com/main.aspx?etn=account&pagetype=entityrecord&id=%7B91330924-802A-4B0D-A900-34FD9D790829%
```

To open the **Closed Opportunities** view:

```
https://myorg.crm.dynamics.com/main.aspx?
etn=opportunity&pagetype=entitylist&viewid=%7b00000000-0000-0000-00AA-
000010003006%7d&viewtype=1039
```

To open the **Active Contacts** view with no navigation bar or command bar

```
https://myorg.crm.dynamics.com/main.aspx?
etn=contact&pagetype=entitylist&viewid={00000000-0000-0000-00AA-
000010001004}&viewtype=1039&navbar=off&cmdbar=false
```

ⓘ Note

- Use `Xrm.Navigation.navigateTo` or `Xrm.Navigation.openForm` when you open forms programmatically within the application by using web resources. Do not use `window.open`.
- Outside the application, where pages do not have access to the `Xrm.Navigation.openForm` or `Xrm.Navigation.navigateTo` functions, use `window.open` or a link to open a specific record or form for a table. Displaying a form within an IFrame embedded in another form is not supported.

You will typically use the `getClientUrl` method to retrieve the organization root Url for Model-driven apps.

Query string parameters for the Main.aspx page

💡 Tip

To get the id value for any record, use the **Send a Link** button the command bar. The following is an example of what will be opened in your email application:

```
<https://mycrm/myOrg/main.aspx?etc=4&id=%7b899D4FCF-F4D3-E011-9D26-
00155DBA3819%7d&pagetype=entityrecord>.
```

The id parameter passed to the URL is the encoded id value for the record. In this example the id value is `{899D4FCF-F4D3-E011-9D26-00155DBA3819}`. The encoded version of the GUID substitutes opening and closing brackets `{` and `}` with `%7B` and `%7D`, respectively,

The following are the query string parameters used with the main.aspx page to open forms or views:

Parameter	Description
<code>etn</code>	The logical name of the table. Important: Do not use the <code>etc</code> (table type code) parameter that contains an integer code for the table. This integer code varies for custom tables in different organizations.
<code>extraqs</code>	Optional for forms. This parameter contains encoded parameters within this parameter. Use this parameter to pass values to a form. For more information, see Set column values using parameters passed to a form . When a table has more than one form defined, you can use this parameter to specify which form to open by passing the encoded parameter <code>formid</code> with the value equal to the ID value of the form. For example, to open a form with the ID of '6009c1fe-ae99-4a41-a59f-a6f1cf8b9daf', include this value in the <code>extraqs</code> parameter: <code>formid%3D6009c1fe-ae99-4a41-a59f-a6f1cf8b9daf%0D%0A</code> .
<code>pagetype</code>	The type of page. There are two possible values: <ul style="list-style-type: none">- entityrecord Displays a record form.- entitylist Displays a view.
<code>id</code>	Optional for forms. Use this when you open a specific table record. Pass in the encoded GUID identifier for the table. The encoded version of the GUID substitutes opening and closing brackets <code>"</code> and <code>"</code> with <code>%7B</code> and <code>%7D</code> , respectively, for example <code>{91330924-802A-4B0D-A900-34FD9D790829}</code> is <code>%7B91330924-802A-4B0D-A900-34FD9D790829%7D</code> .
<code>viewid</code>	Required for views. This is the ID of the <code>savedquery</code> or <code>userquery</code> table record that defines the view. The easiest way to get the URL for a view is to copy it. For more information, see Copy the URL for a View .

Parameter	Description
viewtype	<p>Defines the view type. Possible values are as follows:</p> <ul style="list-style-type: none"> - 1039 Use for a system view. The <code>viewid</code> represents the Id of a <code>savedquery</code> record. - 4230 Use for a personal view. The <code>viewid</code> represents the Id of a <code>userquery</code> record.
navbar	<p>Controls whether the navigation bar is displayed and whether application navigation is available using the areas and subareas defined in the sitemap.</p> <ul style="list-style-type: none"> - <code>on</code> The navigation bar is displayed. This is the default behavior if the <code>navbar</code> parameter is not used. - <code>off</code> The navigation bar is not displayed. People can navigate using other user interface elements or the back and forward buttons. - <code>entity</code> On a form, only the navigation options for related tables are available. After navigating to a related table, a back button is displayed in the navigation bar to allow returning to the original record.
cmdbar	<p>Controls whether the command bar is displayed. Note: This capability supports requirements for the Unified Service Desk application. Using this to display a form within an IFrame embedded in another form is not supported.</p> <ul style="list-style-type: none"> - <code>true</code> The command bar is displayed. This is the default. - <code>false</code> The command bar is hidden.

Copy the URL for a View

Many views in model-driven apps let a user copy the URL for a particular view or send an email with the URL for a particular view embedded in the message. This feature makes communication between users easier, and exposes a way for you to gain access to a URL for a view that users can include in another application, such as a SharePoint site.

Note

Do not use this URL to include the view in application navigation using the site map. For more information, see [Display a view in the application navigation using the Site Map](#).

The page displayed by the URL includes the full view. This includes the ribbon, but does not include the application navigation.

Get the URL for a View

1. Open the view you want to use.
2. On the command bar, click on **Actions** and then click **Email a Link**.
3. Paste the link into Notepad and edit it to extract only the URL part of the text that you want.

① Note

- Views that use the user context as a parameter, such as **My Accounts**, cannot be copied.
 - The GUID that represents system views for system tables will be the same for each installation. The GUID for custom tables and custom views will be unique for each installation.

Display a view in the application navigation using the Site Map

When you customize the application navigation using the site map, do not use the view URL that you copied from the application using the steps in [Copy the URL for a view](#) to set as the URL. That URL displays a page that includes the ribbon and produces undesirable results if used in a `<SubArea>` Url parameter.

To display a list of table records within the application for a SubArea set the table column value. This displays the default view for that table and provides the correct title and icon.

However, if you want to have a SubArea element that uses a specific initial default view, use the following Url pattern.

XML

```
Url="/main.aspx?appid=e2bc1066-488f-eb11-b1ac-  
000d3a56ead9&pagetype=entitylist&etn=account&viewid=%7b<GUID value of view  
id>%7d"
```

When you use this URL, you must also specify appropriate values for <Titles> and <Descriptions>, and specify an icon for the table.

ⓘ Note

If you specify the view using the /main.aspx page, the view selector will still be shown. If the user changes the view, Model-driven apps remembers the user's most recent selection and the initial default view displays after they close and re-open their browser.

Opening a dialog process by using a URL

ⓘ Important

Dialogs are deprecated. You should replace dialogs with business process flows or canvas apps. More information: [Replace dialogs with business process flows or canvas apps](#)

A common customization is to enable a user to open a specific dialog process in the context of a specific record. For example, you might want to add a custom button to the ribbon for a specific table using the id value for current record as an input parameter for the dialog process.

To open a dialog you need the following:

- The unique identifier for the dialog.
- The logical name for the table the dialog is created for.
- The unique identifier for the record you want to have the dialog run against.

ⓘ Tip

To get the unique identifier for the dialog, navigate to **Settings**, in the default solution select **Processes**. Select a process and then in the **Actions** options on the command bar, click **Copy a Link**. This will copy a link to edit the dialog to your clipboard, for example, `[organization url]/sfa/workflow/edit.aspx?id=%7b6A6E93C9-1FE6-4C07-91A9-E0E2A7C70976%7d`.

The following sample shows the URL and query string parameters to open a dialog:

```
[organization url]/cs/dialog/rundialog.aspx?DialogId=[dialog unique identifier]&EntityName=[table logical name]&ObjectId=[unique identifier for the record]
```

For example, to open the dialog with id = {6A6E93C9-1FE6-4C07-91A9-E0E2A7C70976} with the account record id = {40C9ADFD-90A8-DF11-840E-00155DBA380F}, use the URL in the following example.

```
[organization url]/cs/dialog/rundialog.aspx?DialogId=%7b6A6E93C9-1FE6-4C07-91A9-E0E2A7C70976%7d&EntityName=account&objectId=%7b40C9ADFD-90A8-DF11-840E-00155DBA380F%7d
```

Tip

If a dialog process is opened from a link, the **Finish** button may not work. The data will be saved, but the user will need to click the **Close** button on the window to close it. This is because other browsers do not provide a `window.close` method if the window is not opened using JavaScript from another window. When possible, use JavaScript and the `window.open` method to open dialog processes rather than simply providing links.

You can create a JavaScript function to open the dialog as shown in the following example:

JavaScript

```
function openDialogProcess(dialogId, entityName, objectId)
{
    var url = Xrm.Page.context.getClientUrl() +
        "/cs/dialog/rundialog.aspx?DialogId=" +
        dialogId + "&EntityName=" +
        entityName + "&ObjectId=" +
        objectId;
    window.open(url);
}
```

Opening a Report by using a URL

You can open a report by passing appropriate parameter values to the following URL:

```
[organization url]/crmreports/viewer/viewer.aspx.
```

This URL accepts the following parameters:

action

Two possible values for this parameter are `run` or `filter`. When `run` is used, the report will be displayed using the default filters. When `filter` is used, the report will display a filter that the user can edit before choosing the **Run Report** button to view the report.

helpID

This parameter is optional. For reports that are included with model-driven apps the value in this parameter allows the **Help** button to display appropriate content about this report when **Help on This Page** is chosen. The value should correspond to the report `FileName` value.

id

This parameter is the report `ReportId` value.

The following examples show URLs that can be used to open reports in model-driven apps.

Open the **Neglected Cases** report using the default filter:

```
[organization url]/crmreports/viewer/viewer.aspx?  
action=run&helpID=Neglected%20Cases.rdl&id=%7b8c9f3e6f-7839-e211-831e-  
00155db7d98f%7d
```

Open the **Top Knowledge Base Articles** report and prompt the user to set filter values:

```
[organization url]/crmreports/viewer/viewer.aspx?  
action=filter&helpID=Top%20Knowledge%20Base%20Articles.rdl&id=%7bd84ec390-  
7839-e211-831e-00155db7d98f%7d
```

The following function shows how to properly encode values in the URL:

```
JavaScript
```

```
function getReportURL(action,fileName,id) {  
    var orgUrl = GetGlobalContext().getClientUrl();  
    var reportUrl = orgUrl +  
        "/crmreports/viewer/viewer.aspx?action=" +
```

```
encodeURIComponent(action) +  
"&helpID=" +  
encodeURIComponent(fileName) +  
"&id=%7b" +  
encodeURIComponent(id) +  
"%7d";  
return reportUrl;  
}
```

See also

[Set column values using parameters passed to a form](#)

[Xrm.Navigation.openUrl](#)

[Configure a form to accept custom querystring parameters](#)

[Customize the ribbon](#)

[Client scripting using JavaScript](#)

[Web resources](#)

[Change application navigation using the SiteMap](#)

Set column values using parameters passed to a form

Article • 11/30/2022

You can set default values for new records created by users by specifying values in the URL that is used to open the form. By default, these values are set in the form, but can be changed by users before they save the record.

Pass parameters to set column record values

ⓘ Note

You can pass parameter values to the form to set column values using the `Xrm.Navigation.openForm` function. For example, see [Example: Use Xrm.Navigation.openForm to Open a new window](#).

When you open a new form by using the URL address, you can include arguments in the `extraqs` parameter to set column values. The following requirements must be met:

- You must encode the parameters passed in the `extraqs` parameter. To encode the parameters, use [encodeURIComponent](#). To use special characters like "=" or "&" in the parameter values, you must double encode (e.g. to set `name` to `A=B&C`, it would be `extraqs=name%3DA%253DB%2526C`).
- The names of the query string arguments must match or include the names of columns for the table.
- The values passed must be valid.
- The value can't be a script.
- Any attempt to pass an invalid parameter or value will result in an error.
- For Boolean columns, use either an integer value of `0` or `1`, or a text value of `true` or `false` to set the value.
- For DateTime columns, use the text value of the date.

Example: Set the value for string columns

The following sample sets the value for the **Name** column of a new account record to "New Account".

The unencoded value for the `extraqs` parameter is "name=New Account".

```
/main.aspx?etn=account&extraqs=name%3DNew%20Account&pagetype=entityrecord
```

Set values for lookup columns

The following table describes five types of lookup columns. For examples using lookup columns, see [Example: Set the value for lookup columns](#) and [Example: Use Xrm.Navigation.openForm to open a new window](#).

Lookup Type	Description
simple lookup	Allows for a single reference to one type of table.
customer lookup	Allows for a single reference to either an account or a contact record.
owner lookup	Allows for a single reference to either a team or a system user record.
partylist lookup	Allows for multiple references to multiple tables.
regarding lookup	Allows for a single reference to multiple tables.

The following guidelines apply when setting the value of a lookup on a form using a query string argument:

- For simple lookups you must set the value and the text to display in the lookup. Use the suffix "name" with the name of the column to set the value for the text. Don't use any other arguments.
- For customer and owner lookups you must set the value and the name in the same way you set them for simple lookups. In addition you must use the suffix "type" to specify the type of table. Allowable values are account, contact, systemuser, and team.
- You can't set the values for partylist or regarding lookups.

Example: Set the value for lookup columns

To set values for lookup columns, use the data value, the name value, and for customer or owner lookups only, specify the type value for the respective column. The following sample sets the owner column to a user named "Mark Folkerts".

The unencoded value for the `extraqs` parameter is "`ownerid={B8C6E040-656E-DF11-B414-00155DB1891A}&owneridname=Mark Folkerts&owneridtype=systemuser`".

```
/main.aspx?etn=lead&pagetype=entityrecord&extraqs=ownerid%3D%7bB8C6E040-656E-DF11-B414-00155DB1891A%7d%26owneridname%3DMark%20Folkerts%26owneridtype%3Dsystemuser
```

The following sample sets the primary contact column to a user named "Yvonne McKay (sample)". The unencoded value for the `extraqs` parameter is "`primarycontactid={43b58571-eefa-e311-80c1-00155d2a68c4}&primarycontactidname=Yvonne McKay (sample)`".

```
/main.aspx?  
etn=account&pagetype=entityrecord&extraqs=primarycontactid%3D%7B43b58571-eefa-e311-80c1-00155d2a68c4%7D%26primarycontactidname%3DYvonne%20McKay%20(sample)
```

ⓘ Note

For a simple lookup like this, you don't have to set a type value.

Example: Set the value for date columns

The following sample sets the **Est. Close Date** column for a new opportunity to January 31, 2011. The unencoded value for the `extraqs` parameter is "`estimatedclosedate=01/31/11`".

```
/main.aspx?  
etn=opportunity&extraqs=estimatedclosedate%3D01%2F31%2F11&pagetype=entityrec  
ord
```

Example: Set the value for choice columns

To set the value for a **Choice** column, set the integer value for the option. The following sample sets the **Role** column value to "Decision Maker" in a new contact record.

The unencoded value for the `extraqs` parameter is "accountrolecode=1".

```
/main.aspx?etn=contact&extraqs=accountrolecode%3D1&pagetype=entityrecord
```

Example: Set the value for choices columns

To set the value for **Choices** column, Specify integer values for the options in the URL that is used to open the form. For example, to set the options for the **Hobbies** column, the unencoded value for the `extraqs` parameter will be "hobbies=[1,3,4]".

```
/main.aspx?  
etn=contact&extraqs=hobbies%3D%5B1%2C3%2C4%5D&pagetype=entityrecord
```

Example: Use Xrm.Navigation.openForm to open a new window

The following sample sets default values on several different columns and shows how to use the `Xrm.Navigation.openForm` function. It is equivalent to the previous example that used the `window.open` method.

Javascript

```
function OpenNewContact() {  
    var parameters = {};  
    //Set the Parent Customer column value to "Contoso".  
    parameters["parentcustomerid"] = "2878282E-94D6-E111-9B1D-00155D9D700B";  
    parameters["parentcustomeridname"] = "Contoso";  
    parameters["parentcustomeridtype"] = "account";  
    //Set the Address Type to "Primary".  
    parameters["address1_addressstypecode"] = "3";  
    //Set text in the Description column.  
    parameters["description"] = "Default values for this record were set  
    programmatically.";  
    //Set Do not allow E-mails to "Do Not Allow".
```

```

parameters["donotemail"] = "1";

// Define the table name to open the form
var entityFormOptions = {};
entityFormOptions["entityName"] = "contact";

// Open the form
Xrm.Navigation.openForm(entityFormOptions, parameters).then(
    function (success) {
        console.log(success);
    },
    function (error) {
        console.log(error);
    });
}

```

Example: Use window.open to open a new window

The following sample sets default values on several different columns and shows how to use [encodeURIComponent](#) to encode the value of the `extraqs` parameter. If you use the [window.open](#) method, you can control the features of the window that is opened.

Javascript

```

function OpenNewContact() {
    //Set the Parent Customer column value to "Contoso".
    var extraqs = "parentcustomerid={F01F3F6D-896E-DF11-B414-00155DB1891A}";
    extraqs += "&parentcustomeridname=Contoso";
    extraqs += "&parentcustomeridtype=account";
    //Set the Address Type to "Primary".
    extraqs += "&address1_addressstypecode=3";
    //Set text in the Description column.
    extraqs += "&description=Default values for this record were set
programmatically.";
    //Set Do not allow E-mails to "Do Not Allow".
    extraqs += "&donotemail=1";
    //Set features for how the window will appear.
    var features = "location=no,menubar=no,status=no,toolbar=no";
    // Open the window.
    window.open("/main.aspx?etn=contact&pagetype=entityrecord&extraqs=" +
        encodeURIComponent(extraqs), "_blank", features, false);
}

```

See also

Open forms and views with a URL

[openForm](#)

Configure a form to accept custom querystring parameters

Configure a form to accept custom querystring parameters

Article • 11/30/2022

The ability to pass values to a web page by using query strings represents a concern for security. Model-driven apps applies the best practice of always comparing any parameter passed as a query string against a list of expected parameter names and data types.

By default, model-driven apps allows a specified set of query string parameters to be passed to a form. You use these parameters to set default values when you create a new record in the application. Each parameter must use a standard naming convention that includes a reference to the column logical name. More information: [Set column values using parameters passed to a form](#).

In your applications, you may want to pass custom query string parameters to a form. This article provides information about how you can define a set of specific parameter names and data types that can be accepted for a specific form.

Define allowed query string parameters

There are two ways to specify which query string parameters will be accepted by the form:

- Edit form properties
- Edit form XML

Edit form properties

When you edit a form, on the **Home** tab in the **Form** group, select **Form Properties**. In the **Form Properties** dialog box, select the **Parameters** tab.

Use this tab to modify the names and data types that the form allows.

Edit FormXml

Within the exported solution `customizations.xml` file, immediately following the `footer` element, you can add a `<formparameters>` element. In the `<formparameters>` element, add `<querystringparameter>` elements to specify which parameters will be allowed.

The following describes the `querystringparameter` element parameters, `name` and `type`:

- **name.** Each name column must contain at least one underscore ('_') character, but the name of the query string parameter cannot begin with an underscore. The name also can't start with "crm_". We strongly recommend that you use the customization prefix of the solution publisher as the naming convention. It's also recommended not to have the name (querystringparameter name) be the same as the name of a column on the table. A valid `querystringparameter` name value is "myISV_contact_specialvalue".

 **Important**

If a `querystringparameter` element name is not unique, it may be overwritten by another parameter definition using a different data type.

- **Type.** Match the data type values with the parameter values so that invalid data is not passed with the parameter. The following are valid data types:

- Boolean
- DateTime
- Double
- EntityType
- Integer
- Long
- PositiveInteger

 **Note**

PositiveInteger includes "0" in the range of valid values.

- SafeString
- UniqueId
- UnsignedInt

See also

[Set column values using parameters passed to a form](#)

[Open forms and views with a URL](#)

Open forms, views, and dashboards in mobile client with a URL

Article • 11/30/2022

Use the new application handler for mobile clients to directly link to forms, views, and dashboards from external applications so that when you click on the link in an external application, the target element opens in Dynamics 365 for phones or Dynamics 365 for tablets. You can also open an empty form for creating a table record.

If you are already signed in and have a model-driven app opened in Dynamics 365 for phones or Dynamics 365 for tablets, the target record is displayed in the mobile client when you click the link in external application. Otherwise, a login screen or the application list is displayed and link opening is cancelled. You must have Dynamics 365 for phones or Dynamics 365 for tablets installed on your mobile device to use this feature.

Query string parameters for the URL

Use the following application handler and query string parameters to compose the URL.

```
ms-dynamicsxrm://?pagetype=<VALUE>&etn=<VALUE>&id=<VALUE>
```

The query string parameters shown in the following table are used.

Query string parameter	Description
pagetype	Specify the page type to open. Valid values are <code>entity</code> , <code>view</code> , <code>dashboard</code> , and <code>create</code> . This parameter is required.
etn	Specify the logical name of the table for which you want to open or create a record. Logical name of tables are in lowercase, and defined in the EntityMetadata.LogicalName property. This parameter is required if the <code>pagetype</code> parameter is set to <code>entity</code> , <code>view</code> , or <code>create</code> .

Query string parameter	Description
id	<p>Specify the ID of the table, view, or dashboard record that you want to open.</p> <p>This parameter is required if the <code>pagetype</code> parameter is set to <code>entity</code> or <code>dashboard</code>.</p> <p>It is optional if the <code>pagetype</code> parameter is set to <code>view</code>. If you do not specify the view ID, the default view for the table specified in the <code>etn</code> parameter is displayed.</p>

Example URLs

Here are some example URLs for opening forms, views, and dashboards.

Action	Example URL
Open an account with account record ID as 91330924-802A-4B0D-A900-34FD9D790829	<code>ms-dynamicsxrm://?</code> <code>pagetype=entity&etn=account&id=91330924-</code> <code>802A-4B0D-A900-34FD9D790829</code>
Open a view with the view record ID as 899D4FCF-F4D3-E011-9D26-00155DBA3819 for the contact table	<code>ms-dynamicsxrm://?</code> <code>pagetype=view&etn=contact&id=899D4FCF-</code> <code>F4D3-E011-9D26-00155DBA3819</code>
Open a default view for the account table	<code>ms-dynamicsxrm://?</code> <code>pagetype=view&etn=account</code>
Open a dashboard with the dashboard record ID as 2E3D0841-FA6D-DF11-986C-00155D2E3002	<code>ms-dynamicsxrm://?</code> <code>pagetype=dashboard&id=2E3D0841-FA6D-DF11-</code> <code>986C-00155D2E3002</code>
Open a form for creating an account record	<code>ms-dynamicsxrm://?</code> <code>pagetype=create&etn=account</code>

See also

[Open forms, views, dialogs, and reports with a URL](#)

Use editable grids

Article • 12/16/2022

Editable grid is a custom control that provides rich inline editing capabilities on web and mobile clients (Dynamics 365 for phones and Dynamics 365 for tablets) including the ability to group, sort, and filter data within the same grid so that you do not have to switch records or views. The editable grid control is supported in the main grid and subgrids on a form in the web client and in dashboards and on form grids on the mobile clients. Although the editable grid control provides editing capability, it honors the read-only grid metadata and field-level security settings. Editable grids also support business rules and form scripting so you can apply custom business logic according to your organization's requirements.

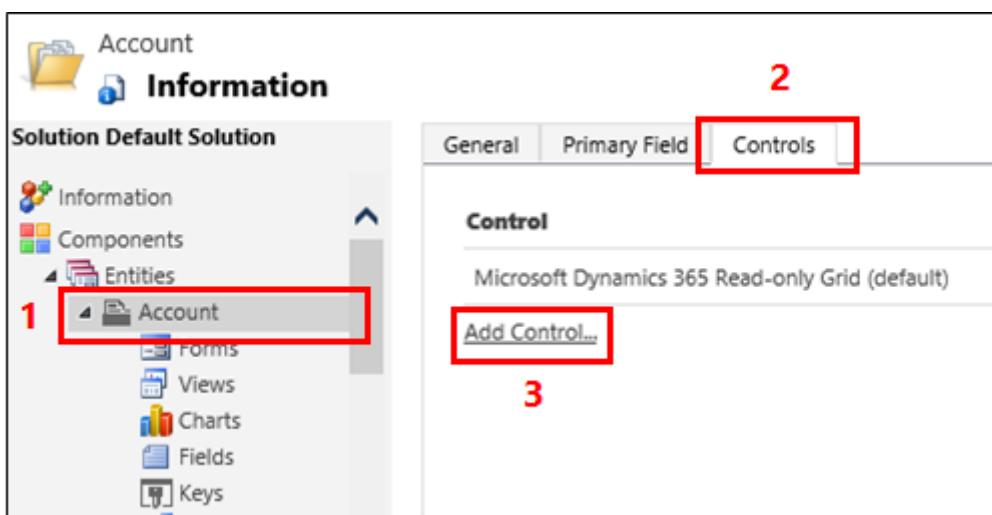
① Note

If you are using legacy forms (versions prior to Dynamics CRM 2016) and enable an editable grid on a subgrid, the editable subgrid will not be rendered. System administrators can turn off legacy forms in system settings, if needed.

Enable editable grids

You can enable editable grids at the table level to use in the main grid, or at the form level to replace read-only subgrids (associated grids) with an editable grid.

1. Open solution explorer.
2. In the Tables list, open the appropriate table, select the **Controls** tab, and then select **Add Control**.



3. In the Add Control dialog box, select **Editable Grid**, and then select **Add**.
4. In the **Editable Grid** row that's added, select the form factor(s) you want to apply the grid to. This makes the editable grid control the default control for the selected form factor(s).

Control	Web	Phone	Tablet
Microsoft Dynamics 365 Read-only Grid (default)	<input checked="" type="radio"/>	<input checked="" type="radio"/>	<input checked="" type="radio"/>
Editable Grid	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Add Control...			X

(!) Note

At runtime, users can toggle between editable grids and read-only grids.

To enable editable grid for a grid in a form, open the form editor, double-click the read-only grid that you want to replace with the editable grid, and then click on the **Controls** tab and select the editable grid from the list of controls.

You can revert to the non-editable grid at any time for main grid and associated grids, if required. Also, at runtime, users can toggle between editable grids and read-only grids.

More information: [Make model-driven apps grids \(lists\) editable using the Editable Grid custom control](#)

Form scripting support

The editable grids support client-side events and methods that can be used to write custom client extensions according to your business need. More information: [Grids and subgrids in model-driven apps \(Client API reference\)](#)

Tables and views supported by editable grid

Not all tables and views support the use of editable grid.

On the web client, a table will support editable grid if all of the following conditions are true:

- The table is customizable (`IsCustomizable = true`)
- The table is either refreshed (`IsAIRUpdated = true`) or a custom table (`IsCustomEntity = true`)

- The table is not a child table (`IsChildEntity = false`)

On the mobile client, a table will support editable grid if the table can be displayed in the mobile client's site map.

For information about the tables that support editable grids, see **Supported out-of-the-box tables** section in [Make model-driven apps grids \(lists\) editable using the Editable Grid custom control](#)

Editable grids do not support roll up associated views (`Rollup type = Related`).

Use the following sample code to generate an XML file that you can open in Excel as an XML table to view the table-support information for editable controls. Excel will figure out the schema automatically, and display the information under the following columns:

- `LogicalName`: Logical name of table.
- `DisplayName`: Display name of table.
- `CanEnableEditableGridWeb`: Displays status (True or False) whether editable grid is supported for the table on the web client.
- `CanEnableEditableGridMobile`: Displays status (True or False) whether editable grid is supported for the table on mobile clients. (Dynamics 365 for phones and Dynamics 365 for tablets).

C#

```
using System;
using System.Linq;
using System.Xml.Linq;
using System.ServiceModel;
using System.ServiceModel.Description;
using System.Collections.Generic;
using System.Xml.Serialization;
using System.Xml;
using System.IO;

// These namespaces are found in the Microsoft.Xrm.Sdk.dll assembly
// found in the SDK\bin folder.
using Microsoft.Xrm.Sdk;
using Microsoft.Xrm.Sdk.Query;
using Microsoft.Xrm.Sdk.Metadata;
using Microsoft.Xrm.Sdk.Client;
using Microsoft.Xrm.Sdk.Messages;

// This namespace is found in Microsoft.Crm.Sdk.Proxy.dll assembly
// found in the SDK\bin folder.
using Microsoft.Crm.Sdk.Messages;
```

```
namespace Microsoft.Crm.Sdk.Samples
{
    /// <summary>
    /// This sample generates an XML table to display the table-support
    information for
    /// editable controls.
    /// </summary>
    public class DumpEditableGridEntityInfo
    {
        #region Class Level Members

        /// <summary>
        /// Stores the organization service proxy.
        /// </summary>
        OrganizationServiceProxy _serviceProxy;

        // Create storage for new columns being created
        public List<AttributeMetadata> addedAttributes;

        // Specify which language code to use in the sample. If you are
        using a language
        // other than US English, you will need to modify this value
        accordingly.
        // See https://msdn.microsoft.com/library/0h88fahh.aspx
        public const int _languageCode = 1033;

        // Define the IDs/variables needed for this sample.
        public int _insertedStatusValue;

        #endregion Class Level Members

        #region How To Sample Code
        /// <summary>
        /// </summary>
        /// <param name="serverConfig">Contains server connection
        information.</param>
        /// <param name="promptForDelete">When True, the user will be
        prompted to delete all
        /// created entities.</param>
        public void Run(ServerConnection.Configuration serverConfig, bool
        promptForDelete)
        {
            try
            {

                // Connect to the Organization service.
                // The using statement assures that the service proxy will
                be properly disposed.
                using (_serviceProxy = new
OrganizationServiceProxy(serverConfig.OrganizationUri,
serverConfig.HomeRealmUri, serverConfig.Credentials,
serverConfig.DeviceCredentials))
                {
                    // This statement is required to enable early-bound type

```

```

support.

        _serviceProxy.EnableProxyTypes();

        //<snippetDumpEditableGridEntityInfo1>
        RetrieveAllEntitiesRequest request = new
RetrieveAllEntitiesRequest()
{
    EntityFilters = EntityFilters.Entity,
    RetrieveAsIfPublished = true
};

        // Retrieve the MetaData.
        RetrieveAllEntitiesResponse response =
(RetrieveAllEntitiesResponse)_serviceProxy.Execute(request);

        // Create an instance of StreamWriter to write text to a
file.
        // The using statement also closes the StreamWriter.
        // To view this file, right click the file and choose
open with Excel.
        // Excel will figure out the schema and display the
information in columns.

        String filename =
String.Concat("EditableGridEntityInfo.xml");
        using (StreamWriter sw = new StreamWriter(filename))
{
    // Create Xml Writer.
    XmlTextWriter metadataWriter = new
XmlTextWriter(sw);

    // Start Xml File.
    metadataWriter.WriteStartDocument();

    // Metadata Xml Node.
    metadataWriter.WriteStartElement("Metadata");

    foreach (EntityMetadata currentEntity in
response.EntityMetadata)
    {
        // Start Entity Node
        metadataWriter.WriteStartElement("Entity");

        bool canBeDisplayedInSitemap =
currentEntity.IsCustomizable.Value;

        if (canBeDisplayedInSitemap)
        {

metadataWriter.WriteLineString("LogicalName", currentEntity.LogicalName);

metadataWriter.WriteLineString("DisplayName",
currentEntity.DisplayName.UserLocalizedLabel?.Label);

metadataWriter.WriteLineString("CanEnableEditableGridWeb", (!

```

```

        (bool)currentEntity.IsChildEntity && ((bool)currentEntity.IsAIRUpdated ||  

        (bool)currentEntity.IsCustomEntity)).ToString());  
  

        metadataWriter.WriteElementString("CanEnableEditableGridMobile",  

        (currentEntity.IsVisibleInMobileClient.Value ||  

        currentEntity.IsVisibleInMobileClient.CanBeChanged).ToString());  

    }  
  

        // Write the Entity's Information.  
  

        //End Entity Node  

        metadataWriter.WriteEndElement();  

    }  
  

        // End Metadata Xml Node  

        metadataWriter.WriteEndElement();  

        metadataWriter.WriteEndDocument();  
  

        // Close xml writer.  

        metadataWriter.Close();  

        Console.WriteLine("Dumped information in the  

EditableGridEntityInfo.xml file");  

    }  

}  

}  
  

// Catch any service fault exceptions that Microsoft Dynamics  

CRM throws.  

catch  

(FaultException<Microsoft.Xrm.Sdk.OrganizationServiceFault>)  

{  

    // You can handle an exception here or pass it back to the  

calling method.  

    throw;  

}  

}  

#endregion How To Sample Code  
  

#region Main  

/// <summary>  

/// Standard Main() method used by most SDK samples.  

/// </summary>  

/// <param name="args"></param>  

static public void Main(string[] args)  

{  

    try  

{  

    // Obtain the target organization's Web address and client  

logon  

        // credentials from the user.  

ServerConnection serverConnect = new ServerConnection();  

        ServerConnection.Configuration config =  

serverConnect.GetServerConfiguration();  

        DumpEditableGridEntityInfo app = new  

DumpEditableGridEntityInfo();
```

```

        app.Run(config, false);
    }
    catch
(FaultException<Microsoft.Xrm.Sdk.OrganizationServiceFault> ex)
{
    Console.WriteLine("The application terminated with an
error.");
    Console.WriteLine("Timestamp: {0}", ex.Detail.Timestamp);
    Console.WriteLine("Code: {0}", ex.Detail.ErrorCode);
    Console.WriteLine("Message: {0}", ex.Detail.Message);
    Console.WriteLine("Plugin Trace: {0}", ex.Detail.TraceText);
    Console.WriteLine("Inner Fault: {0}",
        null == ex.Detail.InnerFault ? "No Inner Fault" : "Has
Inner Fault");
}
catch (System.TimeoutException ex)
{
    Console.WriteLine("The application terminated with an
error.");
    Console.WriteLine("Message: {0}", ex.Message);
    Console.WriteLine("Stack Trace: {0}", ex.StackTrace);
    Console.WriteLine("Inner Fault: {0}",
        null == ex.InnerException.Message ? "No Inner Fault" :
ex.InnerException.Message);
}
catch (System.Exception ex)
{
    Console.WriteLine("The application terminated with an
error.");
    Console.WriteLine(ex.Message);

    // Display the details of the inner exception.
    if (ex.InnerException != null)
    {
        Console.WriteLine(ex.InnerException.Message);

FaultException<Microsoft.Xrm.Sdk.OrganizationServiceFault> fe
        = ex.InnerException
        as
FaultException<Microsoft.Xrm.Sdk.OrganizationServiceFault>;
        if (fe != null)
        {
            Console.WriteLine("Timestamp: {0}",
fe.Detail.Timestamp);
            Console.WriteLine("Code: {0}", fe.Detail.ErrorCode);
            Console.WriteLine("Message: {0}",
fe.Detail.Message);
            Console.WriteLine("Plugin Trace: {0}",
fe.Detail.TraceText);
            Console.WriteLine("Inner Fault: {0}",
                null == fe.Detail.InnerFault ? "No Inner Fault"
: "Has Inner Fault");
        }
    }
}

```

```
        }

        // Additional exceptions to catch:
SecurityTokenValidationException, ExpiredSecurityTokenException,
    // SecurityAccessDeniedException, MessageSecurityException, and
SecurityNegotiationException.

    finally
    {

        Console.WriteLine("Press <Enter> to exit.");
        Console.ReadLine();
    }

}

#endregion Main

}

}
```

See also

[Grids and subgrids in model-driven apps \(Client API reference\)](#)

[Make model-driven apps grids \(lists\) editable using the editable grid custom control](#)

Query and edit an organization theme

Article • 11/30/2022

You can define and apply visual themes for an organization. This provides a supported way to apply an organization's logo and color choices to the application. You can create a custom theme for your application by making changes to the default colors and visual elements provided in the un-customized model-driven apps system. For example, you can create your personal product branding, add a company logo and provide table-specific coloring. The theme colors are applied globally throughout the application, with the exception of some legacy areas.

Theme customization is supported in this release only for the web application. The changes made for an organization's theme are not included in solutions exported from the organization. You can define multiple themes, but only one can be set and published as the default theme.

Query the current theme

You may need to query the current theme using client-side code if you have a solution with HTML web resources which you want to adapt to theme choices made for an organization. You can use the following query with the Web API to retrieve that information.

Request:

HTTP

```
GET [Organization URI]/api/data/v9.0/themes?$filter=isdefaulttheme eq true&$select=defaultentitycolor,defaultcustomentitycolor,controlborder,controlshade,selectedlinkeffect,globallinkcolor,processcontrolcolor,headercolor,logotooltip,hoverlinkeffect,navbars helfcolor,navbarbackgroundcolor
```

Response:

JSON

```
HTTP/1.1 200 OK
Content-Type: application/json; odata.metadata=minimal
OData-Version: 4.0

{
    "@odata.context": "[Organization
URI]/api/data/v9.0/$metadata#themes(defaultentitycolor,defaultcustomentityco
lor,controlborder,controlshade,selectedlinkeffect,globallinkcolor,processcon
```

```

        "trolcolor,headercolor,logotooltip,hoverlinkeffect,navbarshelfcolor,navbarbac
        kgroungcolor)",
        "value": [
            {
                "defaultentitycolor": "#001CA5",
                "defaultcustomentitycolor": "#006551",
                "controlborder": "#CCCCCC",
                "controlshade": "#F3F1F1",
                "selectedlinkeffect": "#B1D6F0",
                "globallinkcolor": "#1160B7",
                "processcontrolcolor": "#D24726",
                "headercolor": "#1160B7",
                "logotooltip": "Model-driven apps",
                "hoverlinkeffect": "#D7EBF9",
                "navbarshelfcolor": "#DFE2E8",
                "navbarbackgroundcolor": "#002050",
                "themeid": "f499443d-2082-4938-8842-e7ee62de9a23"
            }
        ]
    }
}

```

More information: [Query Data using the Web API](#).

Edit and publish theme data

A theme is created by using the customization tools in the UI, without requiring a developer to write code. Details about how to apply these customizations can be found in [Use a theme to create a custom look for your app](#).

Most theme data is stored within the theme table. Customized colors for specific tables is included in the `EntityMetadata.EntityColor` property. This data is exported with the table if the table is included in a solution.

The following table describes the `Theme` table columns that are valid for update and contain data that is applied by the theme:

Schema Name	Type	Value of default theme	Description
AccentColor	String	#E83D0F	The Unified Interface secondary theme color to be used on the process control.
BackgroundColor	String	#FFFFFF	For internal use only.
ControlBorder	String	#BDC3C7	The color that controls will use for borders.

Schema Name	Type	Value of default theme	Description
ControlShade	String	#FFFFFF	The color for controls to use to indicate when you hover over items.
DefaultCustomEntityColor	String	#00CCA3	The default custom table color if no color is assigned.
DefaultEntityColor	String	#666666	The default color for system tables if no color is assigned.
GlobalLinkColor	String	#1160B7	The color for links, such as email addresses or lookups.
HeaderColor	String	#1160B7	The color for header text, such as form tab labels.
HoverLinkEffect	String	#E7EFF7	The color that commands or lists will use when you hover over the items.
ImportSequenceNumber	Integer	null	Sequence number of the import that created this record.
IsDefaultTheme	Boolean	True	The default value for a custom theme is false.
Logold	String	null	The name of a web resource to use as a logo. Recommended dimensions are a height of 50 pixels and a maximum width of 400 pixels.
LogoToolTip	String	Model-driven apps	The text that will be used as the tooltip and alt text for the logo.
MainColor	String	#3B79B7	The Unified Interface primary theme color to be used on main command bar, buttons and tabs.
Name	String	Model-driven apps Default Theme	The name of the theme table.
NavBarBackgroundColor	String	#002050	The primary navigation bar color.
NavBarShelfColor	String	#DFE2E8	The secondary navigation bar color.

Schema Name	Type	Value of default theme	Description
OverriddenCreatedOn	DateTime	null	Date and time that the record was migrated.
PageHeaderBackgroundColor	String	#E0E0E0	The page header background color.
PanelHeaderBackgroundColor	String	#F3F3F3	The panel header background color.
ProcessControlColor	String	#41A053	The primary color for process controls.
SelectedLinkEffect	String	#F8FAFC	The color that commands or lists will use to indicate selected items.
TransactionCurrencyId	Lookup	null	Exchange rate for the currency associated with the Theme with respect to the base currency.

After you have applied changes, use the [PublishTheme Action](#) or [PublishThemeRequest class](#) to make one of the theme records the current theme.

Exporting and importing themes

Because themes aren't included as part of a solution, if you want to transfer themes from one organization to another you can use the Configuration Migration tool to generate a schema, export the theme data, and import it into a different organization. For details about how to use this tool, see [Move configuration data using the Configuration Migration Tool](#).

See also

[Theme table](#)

[Create a theme](#)

[Developers guide to customization](#)

Power Apps component framework overview

Article • 10/13/2022

Power Apps component framework empowers professional developers and app makers to create code components for model-driven and canvas apps. These code components can be used to enhance the user experience for users working with data on forms, views, dashboards, and canvas app screens. For example, you can:

- Replace a column on a form that displays a numeric text value with a `dial` or `slider` code component.
- Transform a list into an entirely different visual experience bound to the dataset, like a `Calendar` or `Map`.

<https://www.microsoft.com/en-us/videoplayer/embed/RE4slRe?postJs||Msg=true> ↗

ⓘ Important

- Power Apps component framework works only on Unified Interface and not on the legacy web client.
- Power Apps component framework is currently not supported for on-premises environments.

How is it different from web resources?

Unlike HTML web resources, code components are rendered as part of the same context and loaded at the same time as any other components, providing a seamless experience for the user.

You can create code components that can be used across the full breadth of Power Apps capabilities, and reuse these components many times across different tables and forms.

Developers can bundle all the HTML, CSS, and TypeScript files into a single `solution` package file and move across environments, and also make it available via [AppSource](#) ↗ .

Advantages

- Access to a rich set of framework APIs that expose capabilities like component lifecycle management, contextual data, and metadata
- Seamless server access via Web API; utility and data formatting methods; device features like camera, location, and microphone; and easy-to-invoke user experience elements like dialogs, lookups, and full-page rendering
- Support for modern web practices
- Optimized for performance
- Reusability
- Ability to bundle all files into a single solution file.
- Ability to handle being destroyed and reloaded for performance reasons while preserving state.

Licensing

Power Apps component framework licensing requirements are inline with existing connectors and components and are based on the type of data and connections used in your app. More information: [Power Apps pricing ↗](#). To align with the licensing requirements, we will be classifying code components into two types:

- Code components that connect to external services or data directly via the user's browser client and not through connectors are considered as premium. When these components are used in an app, the app becomes premium, and end-users are required to have **Power Apps** licenses.
- Code components that don't connect to external services or data. When these components are used in an app that uses standard features, the app remains standard, and end- users are required to be licensed at minimum for **Office 365**.
More information: [Power Apps pricing ↗](#)
- Code components can be declared as premium components by adding a `<external-service-usage>` node to the component's manifest file with all the external service domains this component is connecting to.

XML

```
<external-service-usage enabled="true">
  <domain>www.microsoft.com</domain>
</external-service-usage>
```

ⓘ Note

If you're currently using code components in model-driven apps connected to Microsoft Dataverse, end users will require **Power Apps** licenses.

Related topics

[What are code components?](#)

[Code components for canvas apps](#)

[Create and build a code component](#)

[Learn Power Apps component framework](#)

[Use code components in Power Pages](#)

Customize visualizations and dashboards

Article • 12/16/2022

Data visualization and analytics in model-driven apps enable you to graphically view and analyze the data for your business, and help you to derive quick insights to make important business decisions. You can configure dashboards in such a way that enables you to view data from multiple areas such as sales, marketing, and service. You can even adjust the data displayed in visualizations and dashboards per your business requirements by applying filters.

The following elements constitute the visualization and analytics abilities:

- **Visualization.** Visualizations enable you to present your data in the form of charts. Charts enable you to view the aggregate or non-aggregate summary of grid data in model-driven apps. The charts are integrated with the grid, and display data in context with the grids. The charts update automatically to reflect the filtering done on the data in the grids. Similarly, when you perform a drill-down operation on the chart, the data in the corresponding grid updates accordingly. You can also use a web resource instead of charts in visualization.
- **Dashboard.** Dashboards act as a business intelligence tool in model-driven apps by providing a snapshot of your data in various forms. You can use a dashboard to simultaneously present data from up to six charts, grids, IFrames, or web resources.

Related topics

[Analyze Data with Dashboards](#)

[View Data with Visualizations \(Charts\)](#)

[Visualization Data Description Schema](#)

Analyze data with dashboards

Article • 12/16/2022

The dashboard tables in Microsoft Dataverse enable you to present data from various charts, grids, IFRAMES, or web resources simultaneously. Dashboards allow you to compare and analyze various pieces of customer information, and give you data snapshots.

Types of dashboards

There are two types of dashboards: organization-owned dashboards and user-owned dashboards.

Organization-owned dashboard

An organization-owned dashboard is represented by the `SystemForm` table, and can't be assigned or shared. These dashboards are solution-aware tables. When you perform an update operation on this table, you must publish the changes for the updates to be available across the organization. When you publish customizations using the [PublishAllXmlRequest](#) message, the newly-created organization-owned dashboards are also published and become available across the organization.

Alternatively, you can publish the changes done to a specific dashboard by using the [PublishXmlRequest](#) message, and specifying the dashboard ID as the parameter in the request message.

User-owned dashboard

A user-owned dashboard is represented by the `UserForm` table, can be assigned and shared with other users, and can be viewed by other users depending on the dashboard's access privileges.

Note

You can't programmatically create or manage the new interactive dashboards. For information about working with interactive dashboards using the web client, see [Configure interactive experience dashboards](#)

See also

[Using FormXML for dashboards](#)

[Actions on dashboards](#)

[Create a dashboard](#)

[Sample dashboards](#)

[Sample: Create, retrieve, update, and delete \(CRUD\) a dashboard ↗](#)

[Sample: Assign a user-owned dashboard to another user ↗](#)

[Visualization data description schema](#)

[Customize visualizations and dashboards](#)

Understand dashboards: Dashboard components and FormXML

Article • 12/16/2022

Dashboards are one of the different types of forms in model-driven apps. You can use the `SystemForm.Type` or `UserForm.Type` to determine whether the form is a dashboard. A form of dashboard type has the property value of "0".

The definition of the form content and presentation is stored in the FormXML. More information: [Form XML Schema](#)

For sample FormXML strings for different types of dashboards, see [Sample Dashboards](#).

Dashboard components

A dashboard can contain charts, grids, IFRAMES, or web resources. By default, a single dashboard can contain up to six of these components.

Charts

An organization-owned dashboard can contain only organization-owned charts. However, a user-owned dashboard can contain user-owned and organization-owned charts. More information [Charts \(Visualizations\) for model-driven apps](#)

Grids

Grids fetch data from queries (views) in model-driven apps. An organization-owned dashboard can contain only the grids that fetch data from saved queries. However, a user-owned dashboard can contain grids that fetch data from user and saved queries. More information: [SavedQuery table](#)

IFRAMEs

When you add an IFRAME to an organization-owned dashboard, you can specify whether to restrict or allow cross-frame scripting. To do so, you have to use the `<Security>` parameter in the IFRAME control in the FormXML. However, for user-owned dashboards, cross-frame scripting for IFRAMES is restricted, and you can't change it. If you attempt to create a user-owned dashboard that contains an IFRAME with cross-frame scripting enabled, an error message will be displayed.

Web resources

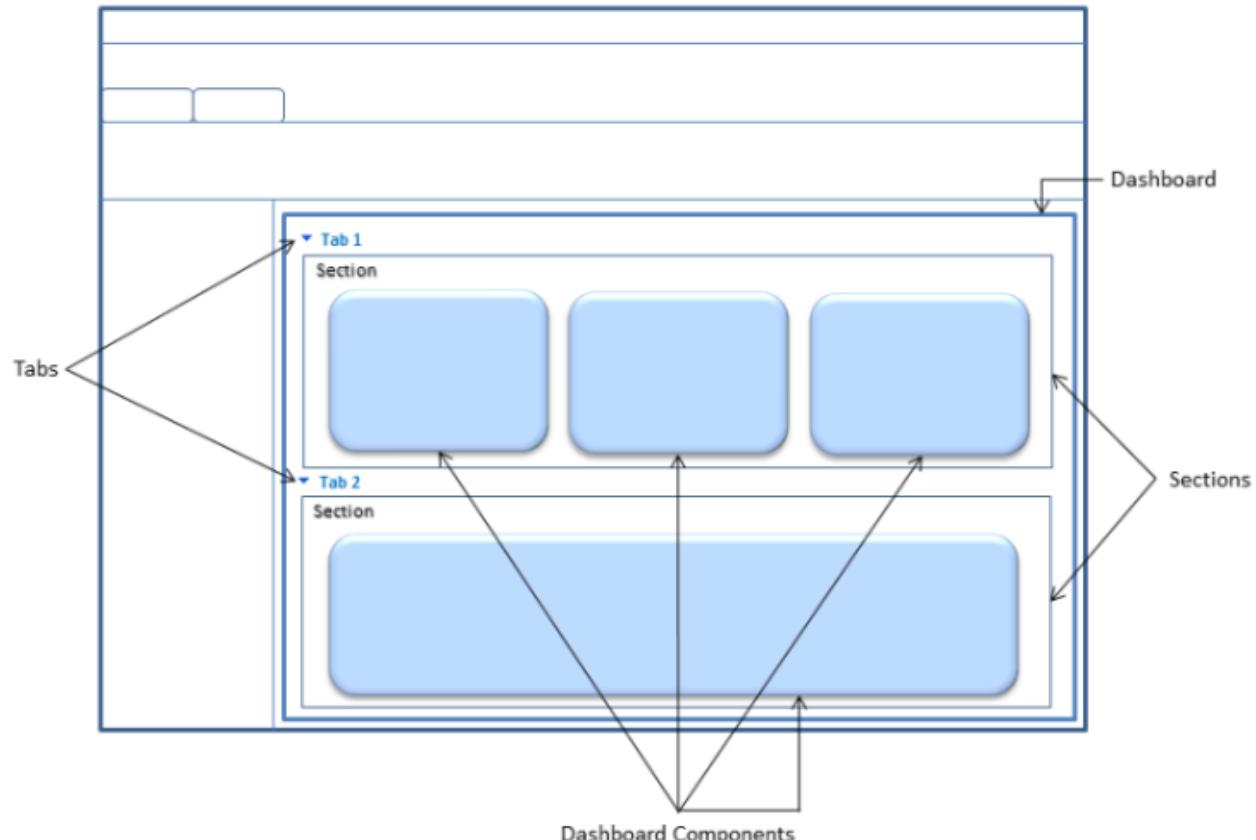
Only form-enabled web resources can be included in a dashboard. Although this restriction is applicable when you are adding a web resource using the Dashboard designer in the web application, there is no such restriction applied when adding a web resource to a dashboard using the SDK. More information: [Web resources for model-driven apps](#)

Dashboard components and FormXML elements

The dashboard components are displayed in model-driven apps based on the values specified in the FormXML. The following image shows an example of a dashboard. Each dashboard can include multiple tabs. Tabs are a vertical stack separating the body of the dashboard, and can be expanded or collapsed. A tab can contain multiple sections. Sections enable for grouping and layout of dashboard components.

 **Note**

Tab and section names are not displayed on a dashboard page.



FormXML elements supported for dashboards

Although dashboards are a type of forms, not all FormXML elements and parameters are supported by dashboards. The following table provides information about the FormXML elements, child elements, and parameters supported by dashboards.

For sample FormXML for different types of dashboards, see [Sample Dashboards](#).

Element	Child Elements	Element parameters
<form>	<tabs>	-
<tabs>	<tab>	-
<tab>	- <labels> - <columns>	- id - name - expanded - verticallylayout - showlabel - locklevel
<labels>	<label>	-
<label>	-	- description - languagecode
<columns>	<column>	-
<column>	<sections>	width
<sections>	<section>	addedby
<section>	- <labels> - <rows>	- id - name - showlabel - showbar - columns
<rows>	<row>	addedby
<row>	<cell>	addedby
<cell>	- <labels> - <control>	- auto - addedby - id - showlabel - rowspan - colspan

Element	Child Elements	Element parameters
<control>	<parameters>	- id - classid
<parameters>	- <Url> - <PassParameters> - <Security> - <Scrolling> - <Border> - <ViewIds> - <ViewId> - <IsUserView> - <IsUserChart> - <TargetEntityType> - <AutoExpand> - <RecordsPerPage> - <EnableQuickFind> - <EnableJumpBar> - <EnableChartPicker> - <EnableViewPicker> - <ChartGridMode> - <VisualizationId>	-

Set the number of dashboard controls

You can use Windows PowerShell to adjust the number of dashboard controls as described here. The maximum value is 20.

To retrieve and set the dashboard limit

1. Open a Windows PowerShell command window.
2. Add the model-driven apps WindowsPowerShell snap-in:

```
PowerShell
Add-PSSnapin Microsoft.Crm.PowerShell
```

3. Retrieve the current setting:

```
PowerShell
$setting = Get-CrmSetting -SettingType DashboardSettings
```

4. Modify the current setting:

```
PowerShell
```

```
$setting.MaximumControlsLimit = 5
```

```
PowerShell
```

```
Set-CrmSetting -Setting $setting
```

See also

[Dashboards](#)

[Actions on dashboards](#)

[Create a dashboard](#)

Actions on dashboards

Article • 12/16/2022

You can perform actions such as create, retrieve, update, or delete on organization-owned and user-owned dashboards.

Actions on an organization-owned dashboard

To perform the following actions on an organization-owned dashboard (`SystemForm`), you must have the System Administrator or the System Customizer role assigned to your account in Microsoft Dataverse:

- Create, retrieve, update, and delete. You can create or update an organization-owned dashboard using the Dataverse web services or customizing the form. For detailed information about creating a dashboard, see [Create a dashboard](#).
- Set an organization-owned dashboard as the default dashboard for an organization by setting the `SystemForm.IsDefault` value to `true` while creating or updating the dashboard.

Important

Using the methods available in the Dataverse Web Services, it is possible to set two dashboards as the default. Make sure that no other dashboard is the default dashboard for the organization before updating this setting programmatically.

After you update an organization-owned dashboard, you must publish the metadata changes to make it visible across the organization. You can use the [PublishAllXmlRequest](#) message or [PublishXmlRequest](#) message to publish the changes made for an organization-owned dashboard. For a sample code that demonstrates this, see [Sample: Create, retrieve, update, and delete \(CRUD\) a dashboard](#).

For a list of supported messages on the organization-owned dashboard table, see [SystemForm table](#).

Actions on a user-owned dashboard

You can perform the following actions on a user-owned dashboard (`UserForm`):

- Create, retrieve, update, and delete. For detailed information about creating a user-owned dashboard, see [Create a dashboard](#).
- Change the ownership of a user-owned dashboard by assigning it to another user or team using the [AssignRequest](#) message.
- Retrieve the access that the specified security principal (user or team) has to a user-owned dashboard using the [RetrievePrincipalAccessRequest](#) message. You can also retrieve all the security principals (users or teams) that have access to a user-owned dashboard, along with their access rights to the user dashboard using the [RetrieveSharedPrincipalsAndAccessRequest](#) message.
- Collaborate with other users and teams on specific areas by sharing a user-owned dashboard with them using the [GrantAccessRequest](#), [ModifyAccessRequest](#), and [RevokeAccessRequest](#) messages.

For a list of supported messages on the user-owned dashboard table, see [UserForm table](#).

See also

[Dashboards for Microsoft Dataverse](#)

[Using FormXML for dashboards](#)

[Create a dashboard](#)

[Sample dashboards](#)

[Sample: Create, retrieve, update, and delete \(CRUD\) a dashboard](#) ↗

[Sample: Assign a user-owned dashboard to another user](#) ↗

Create a dashboard

Article • 12/16/2022

Organization-owned dashboards can be created by using the Microsoft Dataverse or by customizing the form in Dataverse by editing the `customizations.xml` file.

ⓘ Note

Some dashboards that are created by using the SDK or by customizing the form are not supported by the dashboard designer in the Web application. More information: [Limitations: Creating dashboards by using the SDK or through form customization](#) later in this topic.

Before creating a dashboard, consider the following:

- **Type of dashboard:** If you want your dashboards to be available across the organization and do not want to manage the access levels at a more detailed level, you might want to create an organization-owned dashboard. However, if you are concerned about the access privileges and security of your dashboard, consider creating a user-owned dashboard where you have more control on who can access it.

To create organization-owned dashboards, you must have the System Administrator or System Customizer role.

- **Dashboard layout:** While creating dashboards, you have to use the FormXML to define the dashboard components and layout. More information: [Dashboard components and FormXML elements](#). For some sample FormXMLs of different types of dashboards, see [Sample dashboards](#).

Create a dashboard by using the SDK

To create a dashboard, create an instance of `SystemForm` for an organization-owned dashboard, or `UserForm` for a user-owned dashboard. The following sample shows how to create an organization-owned dashboard.

C#

```
//This is the language code for U.S. English. If you are running this code  
//in a different locale, you will need to modify this value.  
int languageCode = 1033;
```

```

//We set up our dashboard and specify the FormXml. Refer to the
//FormXml schema in the Microsoft Dynamics CRM SDK for more information.
SystemForm dashboard = new SystemForm
{
    Name = "Sample Dashboard",
    Description = "Sample organization-owned dashboard.",
    FormXml = String.Format(@"<form>
        <tabs>
            <tab name='Test Dashboard' verticallayout='true'>
                <labels>
                    <label description='Sample Dashboard'
languagecode='{0}' />
                </labels>
                <columns>
                    <column width='100%'>
                        <sections>
                            <section name='Information Section'
showlabel='false' showbar='false'
columns='111'>
                                <labels>
                                    <label description='Information
Section'
languagecode='{0}' />
                                </labels>
                                <rows>
                                    <row>
                                        <cell colspan='1' rowspan='10'
showlabel='false'>
                                            <labels>
                                                <label description='Top
Opportunities - 1'
languagecode='{0}' />
                                            </labels>
                                            <control
id='TopOpportunities'
classid='{{E7A81278-8635-
4d9e-8D4D-59480B391C5B}}'>
                                                <parameters>
                                                    <ViewId>{1}</ViewId>
                                                <RelationshipName />
                                            </control>
                                        </cell>
                                    </row>
                                </rows>
                            </section>
                        </sections>
                    </column>
                </columns>
            </tab>
        </tabs>
    </form>"), CultureInfo.CurrentCulture);
}

```



```
        <row />
    </rows>
</section>
</sections>
</column>
</columns>
</tab>
</tabs>
</form>" ,
languageCode,
defaultOpportunityQuery.SavedQueryId.Value.ToString("B"),
visualization.SavedQueryVisualizationId.Value.ToString("B")),
IsDefault = false
};
_dashboardId = service.Create(dashboard);
```

For a complete sample, see [Sample: Create, retrieve, update, and delete \(CRUD\) a dashboard](#). For a sample to create a user-owned dashboard, and assign it to another user, see [Sample: Assign a user-owned dashboard to another user](#).

Create an organization-owned dashboard by customizing the form

The customizations.xml file that is exported with an unmanaged solution contains definitions for forms and dashboards. You can add or modify the customizations.xml file to add or update a dashboard.

Create a dashboard by customizing a form

1. Log in to Dataverse.
2. Export a solution. For information about doing so, see [Exporting, Preparing to Edit, and Importing the Ribbon](#).
3. Browse to the customizations.xml file in the exported solution folder, and open it for editing.
4. Browse to the end of the dashboards area in the customizations.xml file by searching for the following tag: `</Dashboards>`
5. Before the `</Dashboards>` tag, add the following to define a new dashboard:

XML

```

<Dashboard>
  <LocalizedNames>
    <LocalizedString description="Dashboard_Name" languagecode="1033">
  />
  </LocalizedNames>
  <IsCustomizable>1</IsCustomizable>
  <IsDefault>0</IsDefault>
  <FormXml>
    <forms type="dashboard">
      *** Dashboard definition goes here. *** // See "Sample Dashboards"
      topic for the FormXML content to be used here.
    </forms>
  </FormXml>
</Dashboard>

```

6. Save the customizations.xml file.

7. Import the .zip file as a solution in Dataverse. More information: [Exporting, Preparing to Edit, and Importing the Ribbon](#).

Limitations: Creating dashboards by using the SDK or through form customization

Certain dashboards that are created or modified using the Dataverse or through form customization are not supported by the dashboard designer in the Web application. Avoid the following while creating or modifying a dashboard using the SDK or through form customization.

General

- **Problem:** You can create a dashboard that contains a tab without any section defined in the FormXML.

Resolution: Make sure that you create a dashboard with at least one section defined for each tab in the FormXML.

- **Problem:** You can create a dashboard that does not have the same number of `<row>` elements for a section as specified in the `rowspan` property of a `<cell>` element of the section in the FormXML. Ideally, the `rowspan` property value of a `<cell>` element and the number of `<row>` elements in a section must be same.

Resolution: Make sure that you create a dashboard that has the same number of `<row>` elements for a section as specified in the `rowspan` property of a `<cell>`

element in the section.

Grids

Problem: You can create a dashboard that contains grids with the `<AutoExpand>` parameter value set to `Auto` for the grid.

Resolution: Make sure that you specify the `<AutoExpand>` parameter value as `Fixed` for the grids in the FormXML while creating a dashboard.

IFRAMEs

Problem: You can create a dashboard that contains an IFRAME. This happens when you do not specify any value for the `<Url>` parameter for the IFRAME control in the FormXML.

Resolution: Make sure that you specify a value for the `<Url>` parameter while creating an IFRAME in the FormXML.

See also

[Dashboards](#)

[Using FormXML for dashboards](#)

[Actions on dashboards](#)

[Sample dashboards](#)

[Sample: Create, retrieve, update, and delete \(CRUD\) a dashboard ↗](#)

[Customize forms](#)

Sample dashboards

Article • 12/16/2022

This topic contains sample dashboards along with the respective FormXML strings. You can specify the FormXML string for a dashboard using the `SystemForm.FormXml` for an organization-owned dashboard or `UserForm.FormXml` for a user-owned dashboard.

Dashboard with charts and grids

The following is a sample dashboard that has four components: three charts and a grid. This is one of the default organization-owned dashboards, available in model-driven apps.



FormXML

The following sample shows the FormXML for this dashboard.

XML
<pre><form> <tabs> <tab showlabel="true" verticallayout="true" id="{4e5d00ec-6b2a-447a-afdf-235e6c5d4599}" name="{4e5d00ec-6b2a-447a-afdf-235e6c5d4599}" locklevel="0" expanded="true"> <columns> <column width="100%"> <sections> <section showlabel="false"</pre>

```
        showbar="false"
        columns="111"
        id="{7f7bbdb7-15d6-4664-bda7-6060d0cd3105}"
        name="{7f7bbdb7-15d6-4664-bda7-6060d0cd3105}">
    <rows>
        <row>
            <cell colspan="1"
                  rowspan="12"
                  showlabel="false"
                  id="{cfa8b70b-b0f0-4b91-9cf3-c4e29925186f}"
                  auto="false">
                <control id="Chart1"
                        classid="{E7A81278-8635-4d9e-8D4D-59480B391C5B}">
                    <parameters>
                        <TargetEntityType>opportunity</TargetEntityType>
                        <ChartGridMode>Chart</ChartGridMode>
                        <EnableQuickFind>true</EnableQuickFind>
                        <EnableViewPicker>false</EnableViewPicker>
                        <EnableJumpBar>true</EnableJumpBar>
                        <RecordsPerPage>12</RecordsPerPage>
                        <ViewId>{00000000-0000-0000-00AA-000010003001}</ViewId>
                        <IsUserView>false</IsUserView>
                        <ViewIds></ViewIds>
                        <AutoExpand>Fixed</AutoExpand>
                        <VisualizationId>{87293554-2482-DE11-9FF3-00155DA3B012}</VisualizationId>
                    </parameters>
                </control>
            </cell>
            <cell colspan="1"
                  rowspan="12"
                  showlabel="false"
                  id="{5fc07b79-9f15-4396-9a66-c8ca4a13cac3}"
                  auto="false">
                <control id="Chart2"
                        classid="{E7A81278-8635-4d9e-8D4D-59480B391C5B}">
                    <parameters>
                        <TargetEntityType>lead</TargetEntityType>
                        <ChartGridMode>Chart</ChartGridMode>
                        <EnableQuickFind>true</EnableQuickFind>
                        <EnableViewPicker>false</EnableViewPicker>
                        <EnableJumpBar>true</EnableJumpBar>
                        <RecordsPerPage>12</RecordsPerPage>
                        <ViewId>{5A926B99-3A5F-DF11-AE90-00155D2E3002}</ViewId>
                        <IsUserView>false</IsUserView>
                        <ViewIds></ViewIds>
                        <AutoExpand>Fixed</AutoExpand>
                        <VisualizationId>{3ED18B7C-5693-DE11-97D4-00155DA3B01E}</VisualizationId>
                    </parameters>
                </control>
            </cell>
        </row>
    </rows>

```



```
<sections>
    <section showlabel="false"
              showbar="false"
              columns="111"
              id="{02aab82b-167e-4f61-8d10-9b5a3fab3d76}"
              name="{02aab82b-167e-4f61-8d10-9b5a3fab3d76}">
        <rows>
            <row>
                <cell colspan="3"
                      rowspan="12"
                      showlabel="false"
                      id="{f0114d8b-f5c9-41b2-9b59-07ff839ef176}"
                      auto="false">
                    <control id="Grid1"
                             classid="{E7A81278-8635-4d9e-8D4D-59480B391C5B}">
                        <parameters>
                            <TargetEntityType>activitypointer</TargetEntityType>
                            <ChartGridMode>All</ChartGridMode>
                            <EnableQuickFind>false</EnableQuickFind>
                            <EnableViewPicker>true</EnableViewPicker>
                            <EnableJumpBar>true</EnableJumpBar>
                            <RecordsPerPage>8</RecordsPerPage>
                            <ViewId>{00000000-0000-0000-00AA-000010001899}</ViewId>
                            <IsUserView>false</IsUserView>
                            <ViewIds></ViewIds>
                            <AutoExpand>Fixed</AutoExpand>
                            <VisualizationId>{EDF35649-5293-DE11-97D4-00155DA3B01E}</VisualizationId>
                            <IsUserChart>false</IsUserChart>
                            <EnableChartPicker>false</EnableChartPicker>
                            <RelationshipName></RelationshipName>
                        </parameters>
                    </control>
                </cell>
            </row>
            <row></row>
            <row></row>
            <row></row>
            <row></row>
            <row></row>
            <row></row>
            <row></row>
            <row></row>
            <row></row>
            <row></row>
        </rows>
    </section>
</sections>
</column>
</columns>
</tab>
</tabs>
</form>
```

See also

[Dashboards](#)

View data with visualizations (charts)

Article • 12/16/2022

Visualizations let you see your business data graphically. A visualization is attached to a table in Microsoft Dataverse. You can attach multiple visualizations to a table, however, only one visualization can be displayed at a time along-side a grid. You can view multiple visualizations at the same time by using a dashboard. More information:

[Analyze data with dashboards](#)

You can use a chart or a web resource as a visualization in Dataverse. For charts, you can use the chart designer in model-driven apps. However, to use a web resource in a visualization, you must either use the SDK or import a custom visualization XML into model-driven apps.

Visualization ownership

In model-driven apps, there are two types of visualization ownership: organization-owned and user-owned.

- An organization-owned visualization is owned by an organization, and cannot be assigned or shared. The `SavedQueryVisualization` table represents the organization-owned visualization. These visualizations are solution-aware tables in model-driven apps. Whenever you update a saved query visualization, you must publish the changes for the updates to be available across the organization by using the [PublishAllXmlRequest](#) message. This table is referred to as a *System Chart* in the model-driven apps web application.
- A user-owned visualization is owned by an individual user, and can be assigned and shared with other users and teams. The `UserQueryVisualization` table represents the user-owned visualization. This table is referred to as a *User Chart* in the model-driven apps web application, and is displayed under **My Charts** in the chart drop-down list.

A user query visualization is not associated with a user query (view), despite the table name. The view aspect of this table is used only for setting the filter criteria.

Chart visualizations

Charts let you see summaries of grid data. The charts are built by using the Microsoft Chart Controls for Microsoft .NET Framework 3.5. For more information about Microsoft

Chart Controls, see [Download: Chart Controls for .NET Framework](#).

These charts are integrated with the grids in the web application. When you apply a filter (query) to the data in a grid, the filter is applied to the chart also, and the chart is updated accordingly. Similarly, when you perform drill-down operations on a chart, the grid data is updated automatically.

A chart attached to a table is available for all the views for the table. A chart displays data according to the currently selected (or displayed) view of a table. A chart can display data from both a saved query (organization-owned view) and a user query (user-owned view).

Charts display data for only those saved queries (organization-owned views) that use FetchXML (`SavedQuery.FetchXml`) to filter the records. If a saved query uses the query API (`SavedQuery.QueryAPI`) to filter the records, the chart will not appear for that saved query. This limitation is not applicable for user queries (user-owned views) because the user query table does not use the `QueryAPI` to filter the records.

For more information about how to work with charts, see [Understanding Charts: Underlying data and chart representation](#).

Chart types in Microsoft chart controls

Microsoft Chart Controls is used to build charts in model-driven apps. Microsoft Chart Controls enable you to create various chart types such as column, bar, area, stacked, line, bubble, and pie.

The following chart types are supported out-of-box in Dataverse: *Column, Area, Bar, Line, Pie, and Funnel*. However, you can extend the functionality by creating other supported Microsoft Chart Controls chart types such as multi-series, stacked, and 100% stacked (comparison) charts by specifying appropriate content in the data description and presentation description XML strings for a chart. More information: [Specifying Chart Data](#)

Web resource visualizations

Web resources are virtual files that are stored in the model-driven apps database and may be retrieved using a unique URL address. You can display an existing web resource as a visualization, and display it in the **Charts** area in model-driven apps together with other charts for a table. For more information about web resources, see [Web resources for model-driven apps](#).

You can use the following types of web resources in a visualization: [Webpage \(HTML\)](#) [web resources](#) and [Image \(JPG, PNG, GIF, ICO\) web resources](#). For more information about how to create a visualization with a web resource, see [Create a web resource visualization](#).

Tables supported for visualizations

You can create and attach visualizations to only those tables in Dataverse that support the new ribbon interface. This is because all of the chart controls are only present in the ribbon interface of Dataverse. Custom tables are also supported for visualizations. You can turn off the visualization support for custom tables if you want to. However, you cannot disable visualization support for the default tables.

The following lists the default tables that are supported for visualizations.

- Account
- ActivityPointer
- Appointment
- BulkOperation
- Campaign
- CampaignActivity
- CampaignResponse
- Competitor
- Connection
- Contact
- Contract
- Email
- Entitlement
- EntitlementChannel
- EntitlementTemplateChannel
- Fax
- Goal
- GoalRollupQuery
- Incident
- Invoice
- InvoiceDetail
- KbArticle
- Lead
- Letter
- List
- Mailbox

Metric
Opportunity
OpportunityProduct
PhoneCall
Position
PriceLevel
Product
ProductAssociation
ProductSubstitute
QueueItem
Quote
QuoteDetail
RecurringAppointmentMaster
Report
SalesLiterature
SalesOrder
SalesOrderDetail
Service
ServiceAppointment
SLAKPIInstance
SocialActivity
SocialProfile
SystemUser
Task
Team
Territory
UoMSchedule

See also

[Chart and analyze data](#)
[Specifying chart data](#)
[Actions on chart](#)
[Create a chart](#)
[Sample charts](#)
[SavedQueryVisualization table](#)
[UserQueryVisualization table](#) [Download: Chart Controls for .NET Framework language pack](#) ↗
[Download: Chart Controls Add-on for Visual Studio](#) ↗
[Download: Chart Controls for .NET Framework documentation](#)

[Samples Environment for Microsoft Chart Controls ↗](#)

[Chart Controls forum ↗](#)

Understand charts: Underlying data and chart representation

Article • 12/16/2022

Charts display data visually by mapping textual values on two axes: horizontal (x) and vertical (y). The x axis is called the *category* axis and the y axis is called the *series* axis. The category axis can display numeric as well as non-numeric values whereas the series axis only displays numeric values.

Charts in model-driven apps can be further classified into the following:

- **Single-series charts:** Charts that display data with a series (y) value mapped to a category (x) value.
- **Multi-series charts:** Charts that display data with multiple series values mapped to a single category value. Multi-series charts include stacked column charts, which vertically display the contribution of each series to a total across categories, and 100% stacked column charts, which compare the percentage that each series contributes to a total across categories. You can combine different compatible chart types in multi-series charts, for example, column and line, bar and line, and so on.

ⓘ Note

Multi-category charts can be created through the web application or by modifying the XML strings described here.

While authoring a chart in model-driven apps using the SDK, you need to consider the following two important aspects:

- **Underlying data for the chart:** Specified using the *data description* XML string.
- **Data representation (appearance):** Specified using the *presentation description* XML string.

ⓘ Note

Microsoft Chart Controls lets you create various types of charts such as column, bar, area, line, pie, funnel, bubble, and radar. The chart designer in model-driven apps lets you create only certain types of charts. However, using the SDK, you can create most of the chart types that are supported by Microsoft Chart Controls.

Use the data description XML string to specify chart data

The data description XML string defines the data that is displayed on the chart. The contents of the XML string are validated against the visualization data description schema. For more information about the schema, see [Visualization Data Description Schema](#).

You can specify the data description XML string while you are creating a chart using the `SavedQueryVisualization.DataDescription` or `UserQueryVisualization.DataDescription` for the organization-owned or user-owned chart respectively.

The data description XML string contains the following two elements:

`<FetchCollection>` and `<CategoryCollection>`.

The `<FetchCollection>` element

The `<FetchCollection>` element uses FetchXML to retrieve data for the chart. The FetchXML query specifies information about the table columns, aggregate functions, and the group by clauses for the data to be displayed in a chart. All the FetchXML aggregate functions are supported for charts. For more information about the FetchXML aggregate functions, see [Use FetchXML aggregation](#).

The FetchXML query enables you to filter your data. Also, filters are applied on charts through views. Therefore, if a filter condition is already specified in the FetchXML query in the `<FetchCollection>` element, and additionally a filter is applied through a view, the chart will display data that is returned after it applies all the filters. For more information about how to use the FetchXML query to filter data, see [Use FetchXML to construct a query](#).

Note

Although the data description XML string is validated again the visualization data description schema, the FetchXML query inside the `<FetchCollection>` element is not. The FetchXML query is validated against the FetchXML schema. For more information, see [FetchXML schema](#).

If the chart is a comparison chart, the `<FetchCollection>` element will contain two groups by* clauses.

The `<CategoryCollection>` element

The `<CategoryCollection>` element contains information about the category (horizontal) and the series (vertical) axes in a chart.

- Each `<Category>` sub-element has a child element called `<MeasureCollection>` that maps to the `<Series>` element in the presentation description XML. A single series chart has a single `<MeasureCollection>` child element whereas a multi-series chart will have multiple `<MeasureCollection>` child elements, each mapped to the respective `<Series>` element in the presentation description XML.
- Each `<MeasureCollection>` child element has an element called `<Measure>` that corresponds to the series (vertical) axis value, corresponding to each value on the category (horizontal) axis.

Example

The following is a sample data description XML string:

XML

```
<datadefinition>
  <fetchcollection>
    <fetch mapping="logical" count="10">
      <entity name="opportunity">
        <attribute name="estimatedvalue" />
        <order attribute="estimatedvalue" descending="true" />
      </entity>
    </fetch>
  </fetchcollection>
  <categorycollection>
    <category>
      <measurecollection>
        <measure alias="estimatedvalue" />
      </measurecollection>
    </category>
  </categorycollection></datadefinition>
```

For more sample data description XML strings, see [Sample Charts](#).

Use the presentation description XML string to specify data representation

The presentation description XML string contains information about the appearance of the chart such as chart title, chart color, and chart type (bar, column, line, and so on). There is no schema definition for this XML string. However, the XML is a serialization of the [Chart](#) class in Microsoft Chart Controls. More information: [Chart Controls](#)

You can specify the presentation description XML string while you are creating a chart using the `SavedQueryVisualization.PresentationDescription` or `UserQueryVisualization.PresentationDescription` for the organization-owned or user-owned chart, respectively.

ⓘ Important

In Unified Interface, only a subset of properties are supported. More information: [Supported methods and properties in Unified Interface](#)

Example for web client

The following is a sample presentation description XML string for web client:

XML

```
<Chart Palette="BrightPastel">
  <Series>
    <Series _Template_="All" Color="153, 204, 255" BorderColor="164, 164,
    164" BorderDashStyle="Solid" BorderWidth="1" ShadowColor="128, 128, 128,
    128" ShadowOffset="1" IsValueShownAsLabel="true" Font="{0}, 6.75pt"
    BackGradientStyle="TopBottom" BackSecondaryColor="0, 102, 153"
    LabelForeColor="100, 100, 100" ChartType="Column">
      <SmartLabelStyle Enabled="True" />
      <Points />
    </Series>
  </Series>
  <ChartAreas>
    <ChartArea _Template_="All" BackColor="White" BorderColor="26, 59, 105"
    BorderWidth="0" BorderDashStyle="Solid">      <AxisY LineColor="204, 204,
    204" TitleFont="{0}, 8pt, GdiCharSet=0" TitleForeColor="100, 100, 100"
    LabelAutoFitMaxFontSize="7" LabelAutoFitMinFontSize="7">
      <MajorTickMark LineColor="Gray" />
      <MajorGrid Enabled="false" />
      <LabelStyle Font="{0}, 6.75pt, GdiCharSet=0" ForeColor="100, 100,
    100" />
    </AxisY>
    <AxisX LineColor="204, 204, 204" TitleFont="{0}, 8pt, GdiCharSet=0" />
```

```

    TitleForeColor="100, 100, 100" LabelAutoFitMaxFontSize="7"
    LabelAutoFitMinFontSize="7" >           <MajorTickMark LineColor="Gray" />
    <MajorGrid Enabled="false" />
        <LabelStyle Font="{0}, 6.75pt, GdiCharSet=0" ForeColor="100, 100,
    100" />
    </AxisX>
</ChartArea>
</ChartAreas>
<Titles>
    <Title _Template_="All" Font="{0}, 9pt, style=Bold, GdiCharSet=0"
    ForeColor="100, 100, 100"></Title>
</Titles>
<BorderSkin PageColor="Control" BackColor="CornflowerBlue"
    BackSecondaryColor="CornflowerBlue" />
</Chart>

```

For more sample presentation description XML strings, see [Sample Charts](#).

Methods and properties supported in Unified Interface

The following section shows the methods and properties that are supported in Unified Interface:

AxisX

Gets or sets the X-axis type of the series.

Properties

Property Name	Description
Enabled	Gets or sets a value that indicates whether an axis is enabled.
LabelStyle Enabled	Gets or sets a flag that indicates whether the label is enabled.
LabelStyle ForeColor	Gets or sets the color of the label.
LabelStyle Format	Gets or sets the formatting string for the label text. More information: Supported numeric format for charts
LineColor	Gets or sets the line color of an axis. More information: Supported color format

Property Name	Description
IsReversed	Gets or sets a flag which indicates whether the axis is reversed. If set to true, it has two effects for x-axis: - x-axis labels are flipped in the reversed order (from right-to-left) - It also bring the y-axis to the opposite side, to accommodate above right-to-left x-axis label.
MajorGrid Enabled	Gets or sets a flag that determines whether major or minor grid lines are enabled.
MajorGrid LineColor	Gets or sets the line color of a grid. More information: Supported color format
MajorTickMark Enabled	Gets or sets a flag that determines whether major grid lines are enabled.
MajorTickMark LineColor	Gets or sets the line color of a grid.
Title	Gets or sets the title of the axis.
TitleForeColor	Gets or sets the text color of an axis title. More information: Supported color format

💡 Tip

- When there are too many `LABELS`, HighCharts omits every second label and tries to render again. A quick work around is to either remove the records, or zoom out the browser.

Example

XML

```
<AxisX Enabled="True" LineColor="165, 172, 181" Title="Test XAxis Title"
TitleForeColor="91,151,213" IsReversed="true">
    <MajorTickMark LineColor="165, 172, 181" Enabled="true" />
    <MajorGrid LineColor="green" Enabled="true"/>
    <LabelStyle ForeColor="red" Format="#,0,.##K" Enabled="true" />
</AxisX>
```

AxisY

Gets or sets the Y-axis type of the series.

Properties

Property	Description
Name	
AxisY2	<p>Gets or sets an Axis object that represents the secondary Y-axis.</p> <ul style="list-style-type: none">- Second Y-axis only applies to multiple series chart.- If you create multiple series chart with the chart editor, by default, the <code>YAxisType=Secondary</code> property will be added to the 2nd series of your chart, and a <code>AxisY2</code> node is added to the XML.- If you want another series to be measured by second Y axis, you can move the <code>YAxisType=Secondary</code> to that series node.- If you don't want a second Y axis, you can delete the <code>YAxisType=Secondary</code>.- If a Y Axis (either primary or secondary) measures more than 1 series, title will not be added to that Y Axis, because Y Axis title doesn't know which series to display.
Enabled	Gets or sets a value that indicates whether an axis is enabled.
Interval	Gets or sets the interval of an axis.
LabelStyle Enabled	Gets or sets a flag that indicates whether the label is enabled.
LabelStyle ForeColor	Gets or sets the color of the label.
LabelStyle Format	Gets or sets the formatting string for the label text. More information: Supported numeric format for charts
LineColor	Gets or sets the line color of an axis. More information: Supported color format
MajorGrid Enabled	Gets or sets a flag that determines whether major grid lines are enabled.
MajorGrid LineColor	Gets or sets the line color of a grid. More information: Supported color format
MajorTickMark Enabled	Gets or sets a flag that determines whether major grid lines are enabled.
MajorTickMark LineColor	Gets or sets the line color of a grid.
Maximum	Gets or sets the maximum value of an axis.
Minimum	Gets or sets the minimum value of an axis.
Title	Gets or sets the title of the axis.

Property	Description
Name	
TitleForeColor	Gets or sets the text color of an axis title. More information: Supported color format

Example

XML

```
<AxisY Enabled="True" LineColor="165, 172, 181" Title="Test YAxis Title"
TitleForeColor="91,151,213" Interval="1" Minimum="0" Maximum="5">
  <MajorTickMark LineColor="165, 172, 181" Enabled="true" />
  <MajorGrid LineColor="green" Enabled="true"/>
  <LabelStyle ForeColor="red" Enabled="true" />
</AxisY>
<AxisY2 Enabled="True" LineColor="165, 172, 181" Title="Test YAxis2 Title"
TitleForeColor="91,151,213" Interval="10" Minimum="0" Maximum="100">
  <MajorTickMark LineColor="165, 172, 181" Enabled="true" />
  <MajorGrid LineColor="green" Enabled="true"/>
  <LabelStyle ForeColor="red" Enabled="true" />
</AxisY2>
```

Chart

The root class for the charts.

Properties

Property Name	Description
PaletteCustomColor	Gets or sets an array of custom palette colors. It follows the priority as shown below: <ul style="list-style-type: none"> - Renders the color defined in the <code>Series</code> node. - If the color pallet is specified, chart picks the color from the color pallet. - If none is specified, it picks up the default color pallet. More information: Supported color format

Example

XML

```
<Chart Palette="None" PaletteCustomColors="91, 151, 213; #4169E1, red,
127,97,142,206">
```

ChartArea

Represents a chart area on the chart image.

Properties

Property Name	Description
Area3DStyle	Gets or sets a value that indicates whether the flag toggles the 3D on and off for a chart area. It supports the following 3D chart types: <ul style="list-style-type: none">- 3D Column- 3D Bar- 3D StackedColumn- 3D StackedBar- 3D StackedColumn100- 3D StackedBar100- 3D Pie
Enable3D	
BackColor	Allow users to set the plot background to either a solid or a gradient color. More information: Supported color format
BackSecondaryColor	Allow users to set the plot background to either a solid or a gradient color. More information: Supported color format
BackGradientStyle	Allow users to set the plot background to either a solid or a gradient color.

Example

XML

```
<ChartArea BackColor="orange" BackSecondaryColor="purple"
           BackGradientStyle="LeftRight" >
    <Area3DStyle Enable3D="true" />
</ChartArea>
```

Legend

Represents the legend for the chart image.

Properties

Property Name	Description
Enabled	Defines whether the legend is enabled. By default it is set to <code>True</code> .

Example

XML

```

<Legends>
  <Legend Enabled="True"/>
</Legends>

```

Series

Stores data points and series.

Properties

Property Name	Description
BorderColor	Gets or sets the border color of the data point. More information: Supported color format
BorderWidth	Gets or sets the border width of the data point.
ChartType	An enumeration value that indicates the chart type that is used to represent the series. The default value is Column. It supports the following chart types: - Column - StackedColumn - StackedColumn100 - Bar - StackedBar - StackedBar100 - Area - StackedArea - StackedArea100 - Line - Pie - Funnel - Tag - Doughnut - Point
Color	Gets or sets the color of the data point. For funnel and pie charts, the color property defined in the series node is ignored, but picks the chart color from color palette. More information: Supported color format
IsValueShownAsLabel	Gets or sets a flag that indicates whether to show the value of the data point on the label.
CustomProperties	Allows users to set <code>FunnelNeckHeight</code> and <code>FunnelNeckWidth</code> to customize funnel chart's shape. <code>FunnelNeckHeight</code> & <code>FunnelNeckWidth</code> represents the percentage. This parameter is only supported for funnel chart types.
IsVisibleInLegend	Gets or sets a flag that indicates whether the item is shown in the legend.

Property Name	Description
LabelForeColor	Gets or sets the text color of the label. More information: Supported color format
LabelFormat	Gets or sets the format of the data point label. More information: Supported numeric format for charts
LegendText	Gets or sets the text of the item in the legend. For funnel and pie charts, the legend displays each data point's value in a series. Instead of displaying the series name as a whole.
YAxisType	Gets or sets the Y-axis type of a series. Only the second Y-axis is supported, not second X-axis.

ⓘ Note

- Currently, we partially support #PERCENT. #VAL and #TOTAL are not supported in Unified Interface.
- For non comparison charts, we support a maximum of 5 series (1 category). For comparison charts, we only support 1 series and 2 categories.

Example

XML

```

<Series>
  <Series ChartType="Column" Color="91, 151, 213" LegendText="Est Revenue"
  IsVisibleInLegend="True" BorderColor="red" BorderWidth="1"
  IsValueShownAsLabel="True" LabelFormat="#,0,.##K" LabelForeColor="59, 59,
  59">
    </Series>
  <Series ChartType="Column" Color="237, 125, 49" LegendText="Actual
  Revenue" IsVisibleInLegend="True" BorderColor="red" BorderWidth="1"
  IsValueShownAsLabel="True" LabelFormat="#,0,.##K" LabelForeColor="59, 59,
  59" YAxisType="Secondary">
    </Series>
  </Series>

```

Supported color format in Unified Interface

Unified Interface supports the following color formats in chart presentation xml, which is compatible with web client:

- RGB Decimal format: 97,142,206

- RGB HEX format: #4169E1
- ARGB Decimal format: 127,90,138,57
- Browser recognized named colors: red, transparent

Supported numeric format for charts in Unified Interface

Formatting values	Description
#,0	No scaling, No decimals, leading zero
#,0,.##K	Thousands, up to 2 decimals, leading zero
#,0,,.##M	Millions, up to 2 decimals, leading zero
#,0,,,.##B	Billions, up to 2 decimals, leading zero
C	Currency with default decimals
C0	Currency with no decimals
C2	Currency with 2 decimals
F0	Fixed point
#,0;(#,0);' '	No scaling, no decimals, leading zero, negative value shown in braces, suppress zeros
#,0,.##K; (#,0,.##K);' '	Thousands, up to 2 decimals, leading zero negative value shown in braces, suppress zeros
#,0,,.##M; (#,0,,.##M);' '	Millions, up to 2 decimals, leading zero negative value shown in braces, suppress zeros
#,0,,,.##B; (#,0,,,.##B);' '	Billions, up to 2 decimals, leading zero negative value shown in braces, suppress zeros
%	Percent sign (%) in a format string causes a number to be multiplied by 100 before it is formatted

See also

[Visualizations \(Charts\)](#)

[Actions on visualizations \(Charts\)](#)

[Create a chart](#)

[Use FetchXML to construct a query](#)

[FetchXML schema Visualization data description schema](#)

Sample charts

Chart class (Microsoft Chart Controls)

Actions on visualizations (charts)

Article • 12/16/2022

Using the Microsoft Dataverse Web Services, you can perform the following actions on the visualization tables.

Actions on organization-owned visualizations

To perform actions on an organization-owned visualization (`SavedQueryVisualization`), you must have the System Administrator or the System Customizer role. You can perform the following actions on an organization-owned visualization:

- Create, retrieve, update, and delete an organization-owned visualization. More information: [Create a visualization](#)

ⓘ Note

After updating an organization-owned visualization, you must publish the table definitions changes to make it visible across the organization by using the `PublishAllXmlRequest` message. Alternatively, whenever you publish a table, all the unpublished organization-owned visualizations that are attached to the table are published automatically.

- Query and retrieve all the organization-owned visualizations that are attached to a table using the `SavedQueryVisualization.PrimaryEntityTypeCode`. Multiple organization-owned visualizations can be attached to a single table. For a list of tables with which you can attach a visualization, see [Tables Supported for visualizations](#). For a code sample that demonstrates how to retrieve all the organization-owned visualizations attached to a table, see [Sample: Retrieve all charts attached to a table ↗](#).

ⓘ Note

You cannot change or update a visualization to attach it with a different table after you have created the visualization. It implies that the `SavedQueryVisualization.PrimaryEntityTypeCode` is not valid for the update action on the organization-owned visualization.

- Specify an organization-owned visualization as the default visualization for the associated table by setting the `SavedQueryVisualization.IsDefault` to `true`. When you set an organization-owned visualization as the default visualization for a table, the visualization is displayed by default when you select to view the visualizations for this table in Dataverse.

 **Note**

Using the Dataverse Web Services, if you set an organization-owned visualization as default for a table that already has another visualization set as default, both the visualizations are marked as default visualizations for the table. To set a visualization as a default visualization for a table, make sure that no other visualization is set as the default visualization for the table.

For a list of supported messages on the organization-owned visualization table, see [SavedQueryVisualization table](#).

Actions on user-owned visualizations

You can perform the following actions on a user-owned visualization (`UserQueryVisualization`):

- Create, retrieve, update, and delete a user-owned visualization. More information: [Create a visualization](#)
- Query and retrieve all the user-owned visualizations that are attached to a table using the `UserQueryVisualization.PrimaryEntityTypeCode`. Multiple user-owned visualizations can be attached to a table. For a list of tables with which you can attach a visualization, see [Tables supported for visualizations](#).

 **Note**

You cannot change or update a visualization to attach it with a different table after you have created the visualization. It implies that the `UserQueryVisualization.PrimaryEntityTypeCode` is not valid for the update action on the user-owned visualization.

- Change the ownership of a user-owned visualization by assigning it to another user or team using [AssignRequest](#).

- Retrieve the access that the specified security principal (user or team) has to a user-owned visualization using [RetrievePrincipalAccessRequest](#). You can also retrieve all the security principals (users or teams) that have access to a user-owned visualization, along with their access rights to the user-owned visualization using the [RetrieveSharedPrincipalsAndAccessRequest](#).
- Collaborate with other users and teams on specific areas by sharing a user-owned visualization with them using [GrantAccessRequest](#), [ModifyAccessRequest](#), and [RevokeAccessRequest](#).

For a list of supported messages on the user-owned visualization table, see [UserQueryVisualization table](#).

See also

Charts

[Understanding Charts: Underlying data and chart representation](#)

[Create a chart](#)

[Sample charts](#)

[Sample: Create, retrieve, update, and delete \(CRUD\) a chart](#) ↗

[Sample: Retrieve all charts attached to a table](#) ↗

[Sample: Assign a chart to another user](#) ↗

[SavedQueryVisualization table](#)

[UserQueryVisualization table](#)

Create a visualization (chart)

Article • 12/16/2022

To create a visualization programmatically, you must create a record for the [SavedQueryVisualization table](#) or [UserQueryVisualization table](#) to create an organization-owned or user-owned chart respectively. This topic shows how to create a chart visualization and a web resource visualization.

Before you create a visualization

Before creating a visualization, make sure that you are aware of the following:

- **Type of visualization:** If you want your visualizations to be available across the organization and don't want to manage the access levels at a more detailed level, you might want to create an organization-owned visualization. However, if you're concerned about the access privileges and security of your visualization, consider creating a user-owned visualization where you have more control over who can access it.

Note

Organization-owned visualizations can only be created by those users who have the System Administrator or System Customizer role.

- **Associated Table:** Visualizations are attached to tables. More information: [Tables supported for visualizations](#). You can attach a chart to a supported table by using the `SavedQueryVisualization.PrimaryEntityTypeCode` or `UserQueryVisualization.PrimaryEntityTypeCode` parameter.

Create a chart visualization

Charts require you to specify the underlying data for the charts and how the charts will look in the form of *data description* and *presentation description* XML strings. More information: [Specifying chart data](#) and [Sample Charts](#).

For a complete sample on how to create an organization-owned chart, see [Sample: Create, retrieve, update, and delete \(CRUD\) a chart](#).

Create a multi-series chart

Multi-series charts map multiple series (vertical) axis values to a single category (horizontal) axis value. The only difference from a single series chart is that these charts have multiple `<measurecollection>` and corresponding `<series>` elements specified in the XML strings. Each `<measurecollection>` element contains a child element called `<measure>` that defines a series (vertical) axis value for the same category (horizontal) value. More information: [Understanding Charts: Underlying data and chart representation](#).

For a sample multi-series chart and the corresponding data description and presentation descriptions XML strings, see [Multi-Series chart](#).

Create a web resource visualization

Visualizations containing web resources don't require you to specify the data description and presentation description XML strings. The following sample demonstrates how to create an organization-owned visualization containing a web resource by using the SDK.

C#

```
var newWebResourceVisualization = new SavedQueryVisualization()
{
    Name = "Sample Dashboard Visualization",
    Description = "Sample organization-owned visualization",
    PrimaryEntityTypeCode = Account.EntityLogicalName,
    WebResourceId = new EntityReference(WebResource.EntityLogicalName,
    _webResourceId)

};

_orgOwnedVisualizationId = service.Create(newWebResourceVisualization);
```

If you want to create a web resource visualization by using the Microsoft Dataverse, you must create an XML file in the following format, and then use **Import Chart** in the ribbon to import the visualization.

XML

```
<visualization>
    <name>Visualization_Name</name>
    <description>Description</description>
    <webresourcename>Name_Of_An_Existing_Web_Resource</webresourcename>
    <primaryentitytypecode>Entity_Logical_Name</primaryentitytypecode>
    <isdefault>Value: true or false</isdefault>
</visualization>
```

For example, to create a *Sample Visualization* that displays an existing Web resource called *new_TestWebResource*, and the visualization should be attached to the *account* table, the XML should look like.

XML

```
<visualization>
  <name>Sample Visualization</name>
  <description>Sample Web Resource Visualization.</description>
  <webresourcename>new_TestWebResource</webresourcename>
  <primaryentitytypecode>account</primaryentitytypecode>
  <isdefault>false</isdefault>
</visualization>
```

See also

[Charts](#)

[Specifying chart data](#)

[Actions on chart](#)

[Sample charts](#)

[Data visualization and analytics](#)

[Sample: Create, retrieve, update, and delete \(CRUD\) a chart ↗](#)

Sample charts

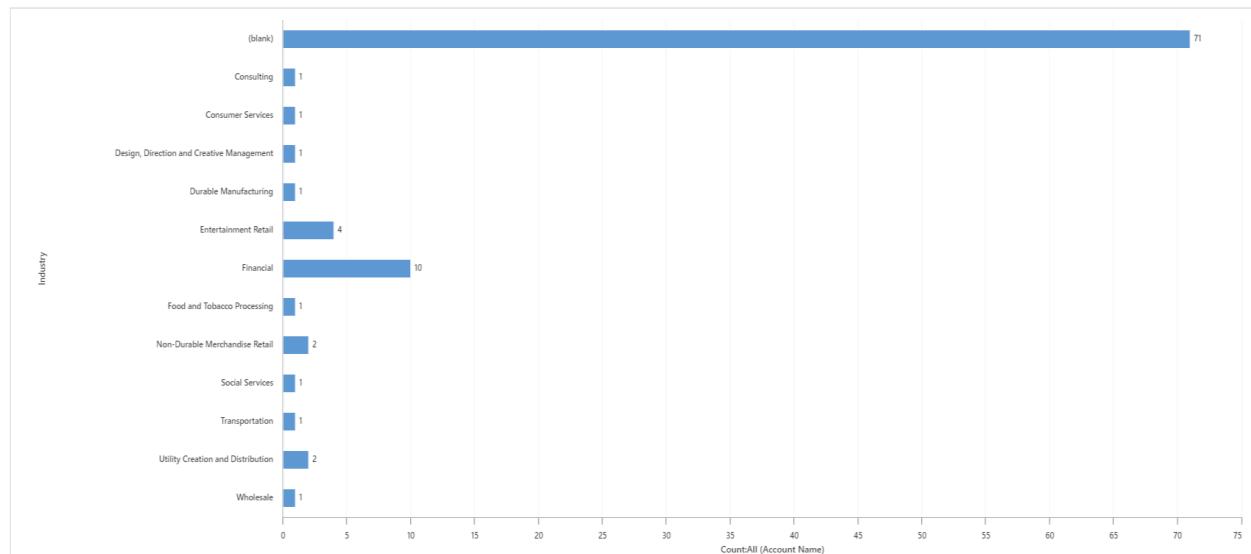
Article • 12/16/2022

This article contains sample charts along with the respective data description and presentation XML strings. You can specify the:

- *Data description XML string* for a chart using the `SavedQueryVisualization.DataDescription` or `UserQueryVisualization.DataDescription` for the organization-owned or user-owned chart respectively.
- *Presentation description XML string* for a chart using the `SavedQueryVisualization.PresentationDescription` or `UserQueryVisualization.PresentationDescription` for the organization-owned or user-owned chart respectively.

Column chart

The following is a column chart that shows the account by industry. We modified the presentation description of the existing Account By Industry default chart available in model-driven apps for the `Account` table to change it to a column chart.



Data description

The following is the contents of the data description XML string for this chart.

XML

```

<datadefinition>
    <fetchcollection>
        <fetch mapping="logical" aggregate="true">
            <entity name="account">
                <attribute groupby="true" alias="groupby_column"
name="industrycode" />
                <attribute alias="aggregate_column" name="name" aggregate="count"
/>
            </entity>
        </fetch>
    </fetchcollection>
    <categorycollection>
        <category>
            <measurecollection>
                <measure alias="aggregate_column" />
            </measurecollection>
        </category>
    </categorycollection>
</datadefinition>

```

Presentation description

The following is the contents of the presentation description XML string for this chart.

XML

```

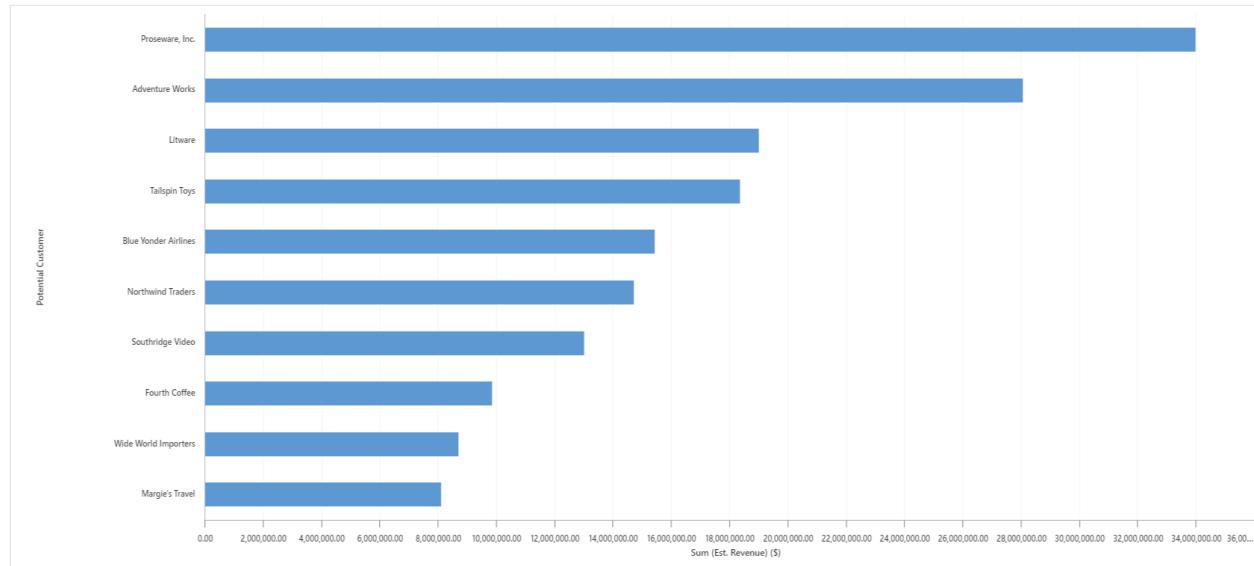
<Chart>
    <Series>
        <Series ChartType="Column" IsValueShownAsLabel="True" Color="91, 151,
213" BackSecondaryColor="112, 142, 50" Font="{0}, 9.5px" LabelForeColor="59,
59, 59" CustomProperties="PointWidth=0.75, MaxPixelPointWidth=40">
            <SmartLabelStyle Enabled="True" />
        </Series>
    </Series>
    <ChartAreas>
        <ChartArea BorderColor="White" BorderDashStyle="Solid">
            <AxisY LabelAutoFitMinFontSize="8" TitleForeColor="59, 59, 59"
TitleFont="{0}, 10.5px" LineColor="165, 172, 181" IsReversed="False">
                <MajorGrid LineColor="239, 242, 246" />
                <LabelStyle Font="{0}, 10.5px" ForeColor="59, 59, 59" />
            </AxisY>
            <AxisX LabelAutoFitMinFontSize="8" TitleForeColor="59, 59, 59"
TitleFont="{0}, 10.5px" LineColor="165, 172, 181" IsReversed="False">
                <MajorGrid Enabled="False" />
                <MajorTickMark Enabled="False" />
                <LabelStyle Font="{0}, 10.5px" ForeColor="59, 59, 59" />
            </AxisX>
        </ChartArea>
    </ChartAreas>
    <Titles>
        <Title Name="Chart Title" DockingOffset="-3" Font="{0}, 13px"

```

```
ForeColor="59, 59, 59" Alignment="TopLeft" />
    </Titles>
</Chart>
```

Bar chart

The following is a bar chart that shows the top 10 customers. This is one of the default charts available in model-driven apps for the `Opportunity` table.



Data description

The following is the contents of the data description XML string for this chart.

XML

```
<datadefinition>
    <fetchcollection>
        <fetch mapping="logical" count="10" aggregate="true">
            <entity name="opportunity">
                <attribute name="estimatedvalue" aggregate="sum"
alias="sum_estimatedvalue" />
                <attribute name="customerid" groupby="true" alias="customerid" />
                <order alias="sum_estimatedvalue" descending="true" />
            </entity>
        </fetch>
    </fetchcollection>
    <categorycollection>
        <category>
            <measurecollection>
                <measure alias="sum_estimatedvalue" />
            </measurecollection>
        </category>
    </categorycollection>

```

```
</categorycollection>  
</datadefinition>
```

Presentation description

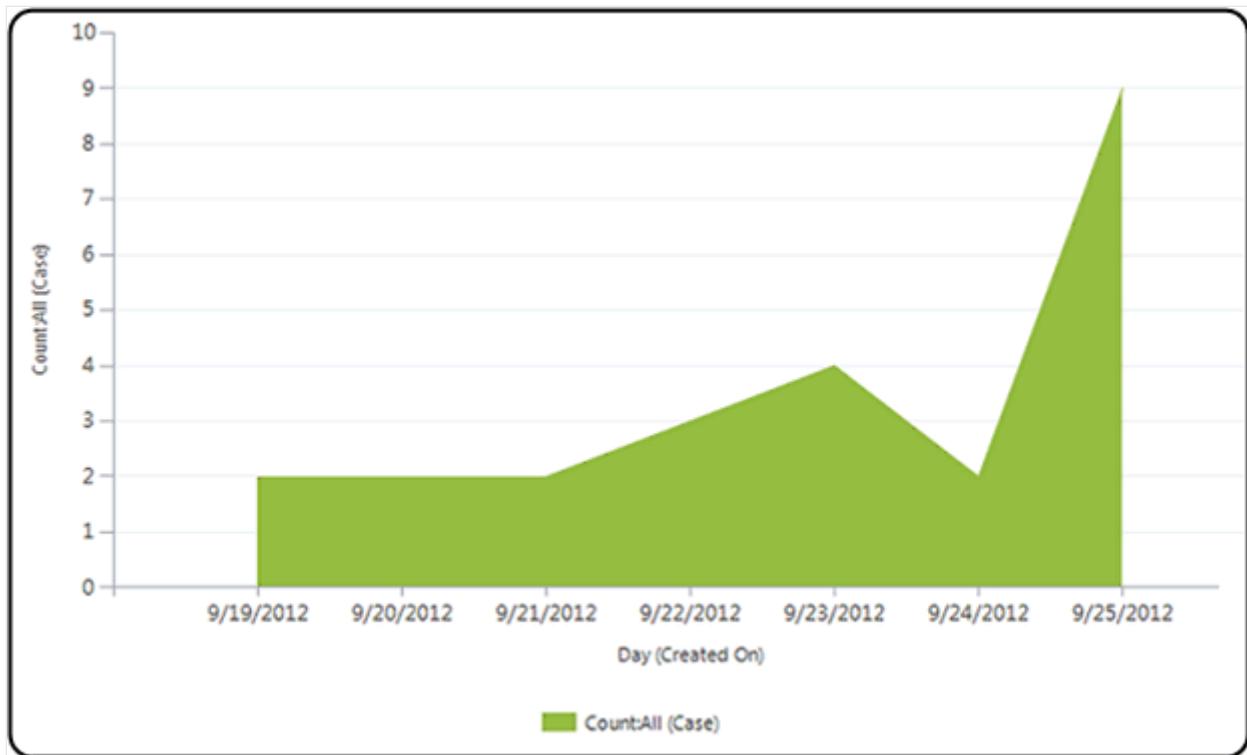
The following is the contents of the presentation description XML string for this chart.

XML

```
<Chart>  
  <Series>  
    <Series ChartType="Bar" IsValueShownAsLabel="False" Color="91, 151, 213"  
      BackSecondaryColor="112, 142, 50" Font="{0}, 9.5px" LabelForeColor="59, 59,  
      59" CustomProperties="PointWidth=0.75, MaxPixelPointWidth=40">  
      <SmartLabelStyle Enabled="True" />  
    </Series>  
  </Series>  
  <ChartAreas>  
    <ChartArea BorderColor="White" BorderDashStyle="Solid">  
      <AxisY LabelAutoFitMinFontSize="8" TitleForeColor="59, 59, 59"  
        TitleFont="{0}, 10.5px" LineColor="165, 172, 181" IsReversed="False">  
        <MajorGrid LineColor="239, 242, 246" />  
        <LabelStyle Font="{0}, 10.5px" ForeColor="59, 59, 59" />  
      </AxisY>  
      <AxisX LabelAutoFitMinFontSize="8" TitleForeColor="59, 59, 59" TitleFont="  
      {0}, 10.5px" LineColor="165, 172, 181" IsReversed="False">  
        <MajorGrid Enabled="False" />  
        <MajorTickMark Enabled="False" />  
        <LabelStyle Font="{0}, 10.5px" ForeColor="59, 59, 59" />  
      </AxisX>  
    </ChartArea>  
  </ChartAreas>  
  <Titles>  
    <Title DockingOffset="-3" Font="{0}, 13px" ForeColor="59, 59, 59"  
      Alignment="TopLeft" />  
  </Titles>  
</Chart>
```

Area chart

The following is an area chart that shows the number of records generated between a given date range.



Data Description

The following is the contents of the data description XML string for this chart.

XML

```
<datadefinition>
    <fetchcollection>
        <fetch mapping="logical" aggregate="true">
            <entity name="incident"><order alias="groupby_column" descending="false" />
                <attribute alias="aggregate_column" name="incidentid" aggregate="count" />
                <attribute groupby="true" alias="groupby_column" dategrouping="day" name="createdon" />
                <attribute groupby="true" alias="groupby_priority" name="prioritycode" />
            </entity>
        </fetch>
    </fetchcollection>
    <categorycollection>
        <category>
            <measurecollection>
                <measure alias="aggregate_column" />
            </measurecollection>
        </category>
    </categorycollection>
</datadefinition>
```

Presentation description

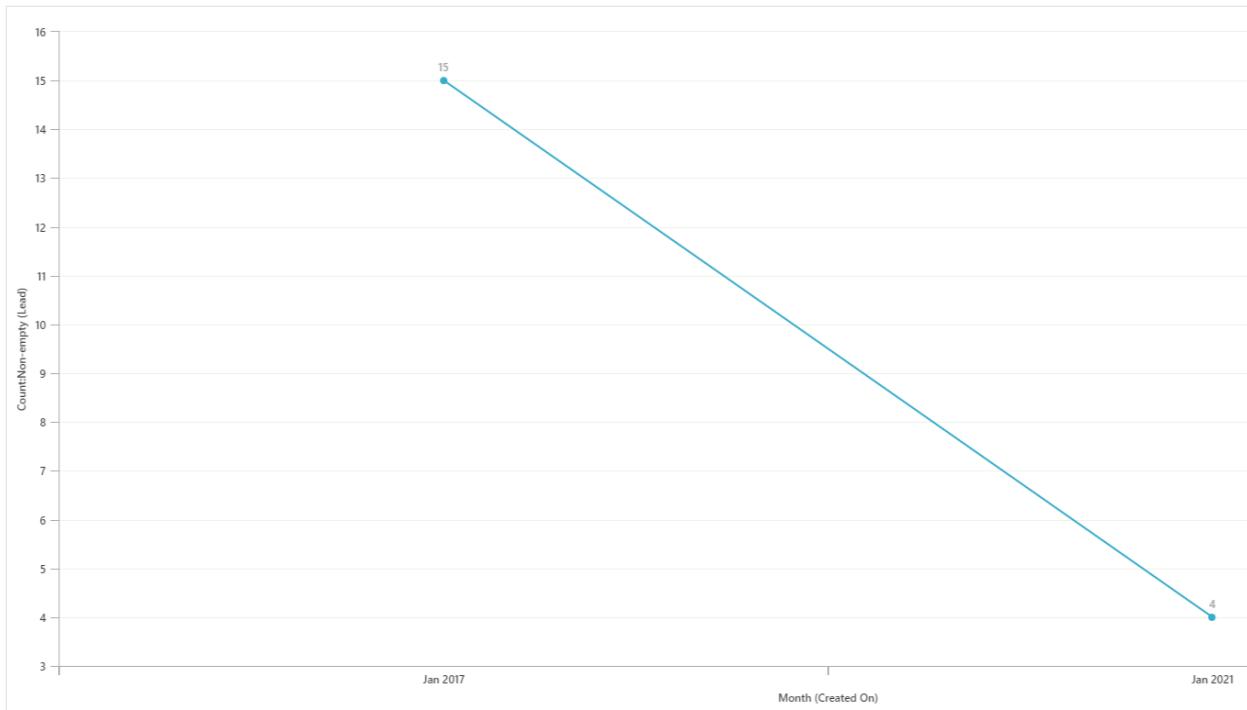
The following is the contents of the presentation description XML for this chart.

XML

```
<Chart Palette="None" PaletteCustomColors="91,151,213; 237,125,49;  
160,116,166; 255,192,0; 68,114,196; 112,173,71; 37,94,145; 158,72,14;  
117,55,125; 153,115,0; 38,68,120; 67,104,43; 124,175,221; 241,151,90;  
186,144,192; 255,205,51; 105,142,208; 140,193,104; 50,125,194; 210,96,18;  
150,83,159; 204,154,0; 51,90,161; 90,138,57;">  
  <Series>  
    <Series ChartType="StackedColumn" Font="{0}, 9.5px" LabelForeColor="59,  
59, 59" CustomProperties="PointWidth=0.75, MaxPixelPointWidth=40" />  
  </Series>  
  <ChartAreas>  
    <ChartArea BorderColor="White" BorderDashStyle="Solid">  
      <AxisY LabelAutoFitMinFontSize="8" TitleForeColor="59, 59, 59"  
TitleFont="{0}, 10.5px" LineColor="165, 172, 181"  
IntervalAutoMode="VariableCount">  
        <MajorGrid LineColor="239, 242, 246" /><MajorTickMark LineColor="165,  
172, 181" />  
        <LabelStyle Font="{0}, 10.5px" ForeColor="59, 59, 59" />  
      </AxisY>  
      <AxisX LabelAutoFitMinFontSize="8" TitleForeColor="59, 59, 59" TitleFont="  
{0}, 10.5px" LineColor="165, 172, 181" IntervalAutoMode="VariableCount">  
        <MajorGrid Enabled="False" /><MajorTickMark Enabled="False" />  
        <LabelStyle Font="{0}, 10.5px" ForeColor="59, 59, 59" />  
      </AxisX>  
    </ChartArea>  
  </ChartAreas>  
  <Titles>  
    <Title Alignment="TopLeft" DockingOffset="-3" Font="{0}, 13px"  
ForeColor="0, 0, 0" />  
  </Titles>  
</Chart>
```

Line chart

The following is a line chart that shows the number of leads generated in the last five months. This is one of the default charts available in model-driven apps for the [Lead](#) table.



Data Description

The following is the contents of the data description XML string for this chart.

XML

```
<datadefinition>
  <fetchcollection>
    <fetch mapping="logical" count="5" aggregate="true">
      <entity name="lead">
        <attribute name="leadid" aggregate="countcolumn"
alias="count_leadid" />
        <attribute name="createdon" groupby="true" dategrouping="month"
usertimezone="false" alias="createdon" />
        <order alias="createdon" descending="false" />
      </entity>
    </fetch>
  </fetchcollection>
  <categorycollection>
    <category>
      <measurecollection>
        <measure alias="count_leadid" />
      </measurecollection>
    </category>
  </categorycollection>
</datadefinition>
```

Presentation description

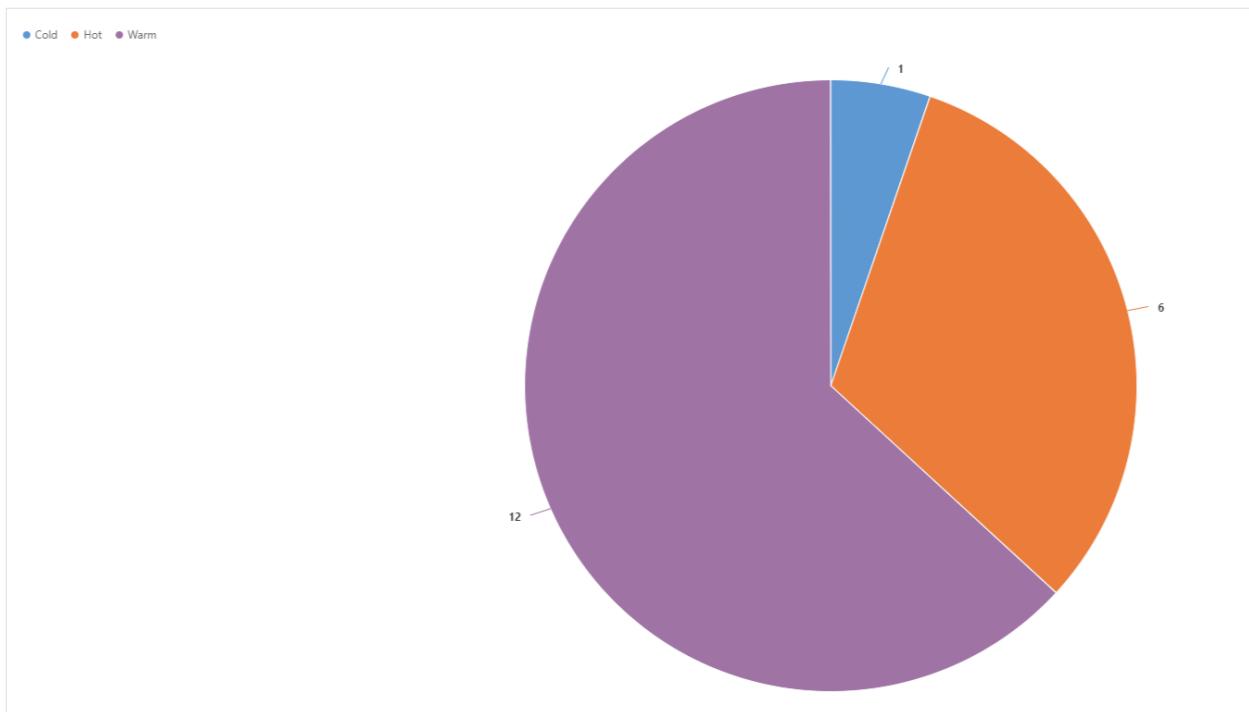
The following is the contents of the presentation description XML string for this chart.

XML

```
<Chart>
  <Series>
    <Series IsValueShownAsLabel="True" BorderWidth="3" ChartType="Line"
Color="49, 171, 204" MarkerStyle="Square" MarkerSize="9" MarkerColor="37,
128, 153" />
  </Series>
  <ChartAreas>
    <ChartArea BorderColor="White">
      <AxisY LabelAutoFitMinFontSize="8" TitleForeColor="59, 59, 59"
TitleFont="{0}, 10.5px" LineColor="165, 172, 181">
        <MajorGrid LineColor="239, 242, 230" /><MajorTickMark LineColor="165,
172, 181" />
        <LabelStyle Font="{0}, 10.5px" ForeColor="59, 59, 59" />
      </AxisY>
      <AxisX LabelAutoFitMinFontSize="8" TitleForeColor="59, 59, 59" TitleFont="
{0}, 10.5px" LineColor="165, 172, 181">
        <MajorGrid Enabled="False" />
        <LabelStyle Font="{0}, 10.5px" ForeColor="59, 59, 59" />
      </AxisX>
    </ChartArea>
  </ChartAreas>
  <Titles>
    <Title DockingOffset="-3" Font="{0}, 13px" ForeColor="59, 59, 59"
Alignment="TopLeft" />
  </Titles>
</Chart>
```

Pie chart

The following is a pie chart that shows the total number of leads and their importance. This is one of the default charts available in model-driven apps for the `Lead` table.



Data description

The following is the contents of the data description XML string for this chart.

```
XML

<datadefinition>
  <fetchcollection>
    <fetch mapping="logical" aggregate="true">
      <entity name="lead"><attribute groupby="true" alias="groupby_column" name="leadqualitycode" />
        <attribute alias="aggregate_column" name="fullname" aggregate="count" />
      </entity>
    </fetch>
  </fetchcollection>
  <categorycollection>
    <category>
      <measurecollection>
        <measure alias="aggregate_column" />
      </measurecollection>
    </category>
  </categorycollection>
</datadefinition>
```

Presentation description

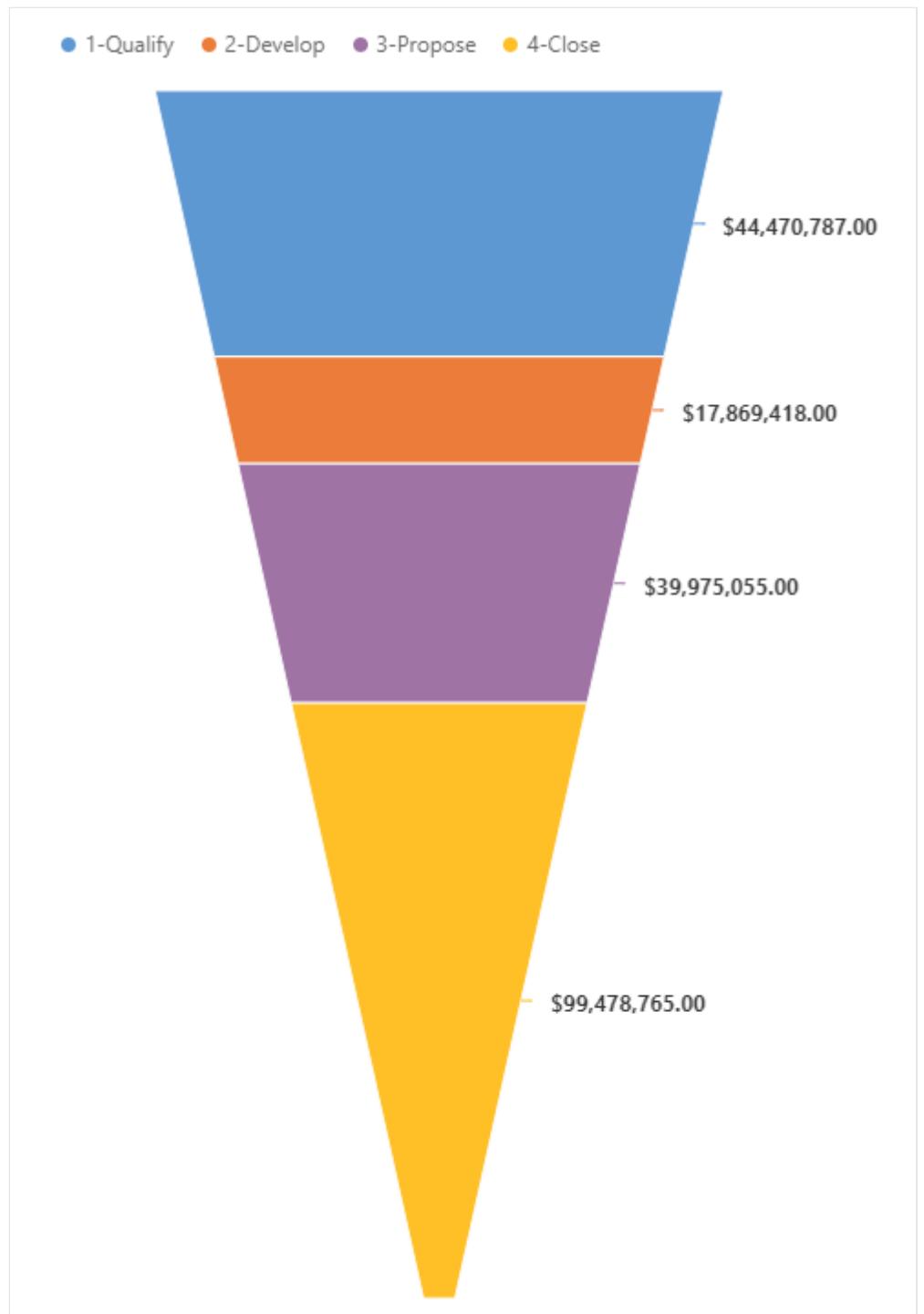
The following is the contents of the presentation description XML string for this chart.

```
XML
```

```
<Chart Palette="None" PaletteCustomColors="91,151,213; 237,125,49;  
160,116,166; 255,192,0; 68,114,196; 112,173,71; 37,94,145; 158,72,14;  
117,55,125; 153,115,0; 38,68,120; 67,104,43; 124,175,221; 241,151,90;  
186,144,192; 255,205,51; 105,142,208; 140,193,104; 50,125,194; 210,96,18;  
150,83,159; 204,154,0; 51,90,161; 90,138,57;">  
    <Series>  
        <Series ShadowOffset="0" IsValueShownAsLabel="true" Font="{0}, 9.5px"  
LabelForeColor="59, 59, 59" CustomProperties="PieLabelStyle=Inside,  
PieDrawingStyle=Default" ChartType="pie">  
            <SmartLabelStyle Enabled="True" />  
        </Series>  
    </Series>  
<ChartAreas>  
    <ChartArea>  
        <Area3DStyle Enable3D="false" />  
    </ChartArea>  
</ChartAreas>  
<Legends>  
    <Legend Alignment="Center" LegendStyle="Table" Docking="right" Font="{0},  
11px" ShadowColor="0, 0, 0, 0" ForeColor="59, 59, 59" />  
</Legends>  
<Titles>  
    <Title Alignment="TopLeft" DockingOffset="-3" Font="{0}, 13px"  
ForeColor="0, 0, 0" />  
</Titles>  
</Chart>
```

Funnel chart

The following is a funnel chart that shows the sum of estimated revenue in each stage of the sales pipeline. This is one of the default charts available in model-driven apps for the `Opportunity` table.



Data description

The following is the contents of the data description XML string for this chart.

```
XML

<datadefinition>
  <fetchcollection>
    <fetch mapping="logical" count="10" aggregate="true">
      <entity name="opportunity">
        <attribute name="estimatedvalue" aggregate="sum"
alias="sum_estimatedvalue" />
        <attribute name="stepname" groupby="true" alias="stepname" />
    
```

```

        <order alias="stepname" descending="false" />
    </entity>
</fetch>
</fetchcollection>
<categorycollection>
    <category>
        <measurecollection>
            <measure alias="sum_estimatedvalue" />
        </measurecollection>
    </category>
</categorycollection>
</datadefinition>

```

Presentation description

The following is the contents of the presentation description XML string for this chart.

XML

```

<Chart Palette="None" PaletteCustomColors="91,151,213; 237,125,49;
160,116,166; 255,192,0; 68,114,196; 112,173,71; 37,94,145; 158,72,14;
117,55,125; 153,115,0; 38,68,120; 67,104,43; 124,175,221; 241,151,90;
186,144,192; 255,205,51; 105,142,208; 140,193,104; 50,125,194; 210,96,18;
150,83,159; 204,154,0; 51,90,161; 90,138,57;">
    <Series>
        <Series ShadowOffset="0" IsValueShownAsLabel="true" Font="{0}, 9.5px"
LabelForeColor="59, 59, 59" ChartType="Funnel"
CustomProperties="FunnelLabelStyle=Outside, FunnelNeckHeight=0,
FunnelPointGap=1, FunnelNeckWidth=5">
            <SmartLabelStyle Enabled="True" />
        </Series>
    </Series>
<ChartAreas>
    <ChartArea>
        <Area3DStyle Enable3D="false" />
    </ChartArea>
</ChartAreas>
<Legends>
    <Legend Alignment="Center" LegendStyle="Table" Docking="right" Font="{0},
11px" ShadowColor="0, 0, 0, 0" ForeColor="59, 59, 59" />
</Legends>
<Titles>
    <Title Alignment="TopLeft" DockingOffset="-3" Font="Segeo UI, 13px"
ForeColor="0, 0, 0" />
</Titles>
</Chart>

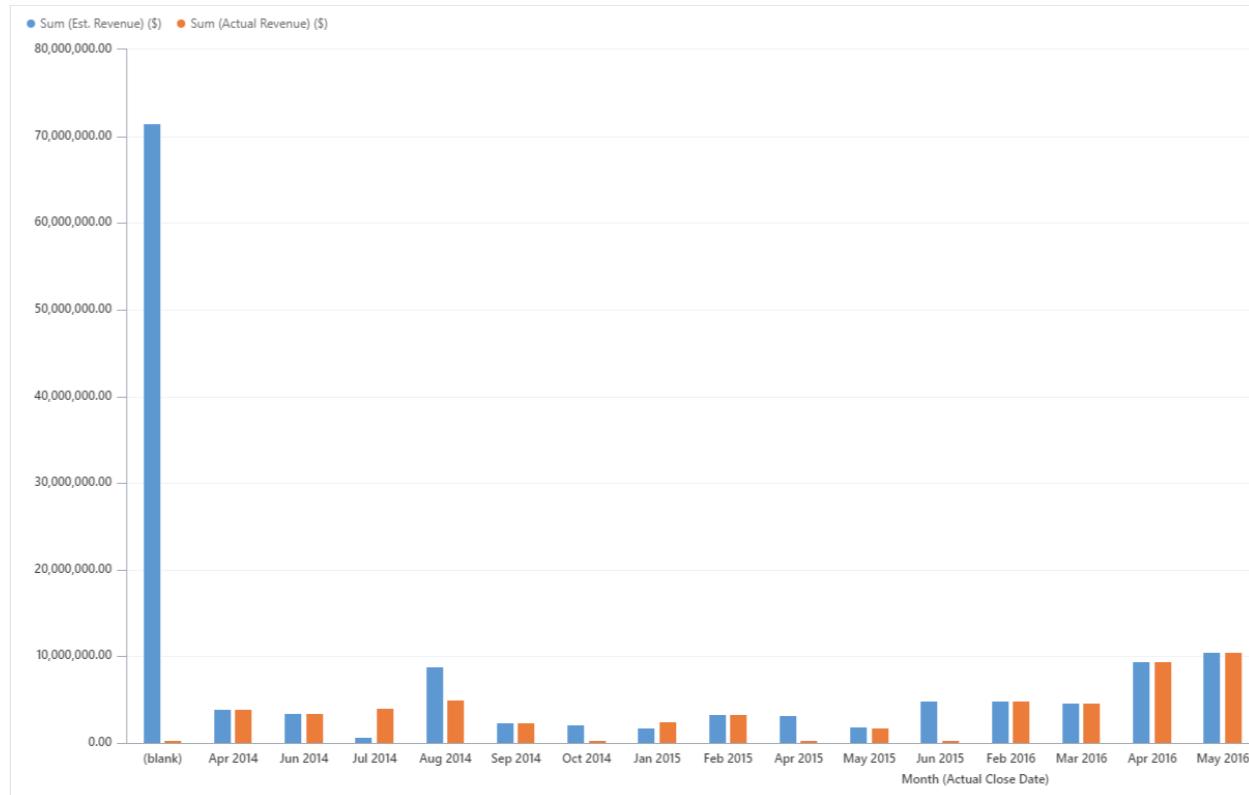
```

Multi-Series chart

The following is a multi-series chart that shows the estimated vs. actual revenue closed by month. You can use the chart designer in model-driven apps or methods described in the developer documentation to create these types of charts.

A multi-series chart has multiple `<measurecollection>` elements in the data description, each mapping to the corresponding `<Series>` element in the presentation description XML string.

A multi-series chart has multiple `<Series>` elements in the presentation description; the number of `<Series>` elements is the same as the number of `<measurecollection>` elements in the data description XML string.



Data description

The following is the contents of the data description XML string for this chart.

XML

```
<datadefinition>
  <fetchcollection>
    <fetch mapping="logical" aggregate="true">
      <entity name="opportunity">
        <attribute name="estimatedvalue" aggregate="sum" alias="estvalue" />
        <attribute name="actualvalue" aggregate="sum" alias="actvalue" />
        <attribute name="actualclosedate" groupby="true"
alias="actclosedate" dategrouping="month" />
      </entity>
    </fetchcollection>
  </datadefinition>
```

```

</fetch>
</fetchcollection>
<categorycollection>
    <category>
        <measurecollection>
            <measure alias="estvalue" />
        </measurecollection>
        <measurecollection>
            <measure alias="actvalue" />
        </measurecollection>
    </category>
</categorycollection>
</datadefinition>

```

Presentation description

The following is the contents of the presentation description XML string for this chart.

XML

```

<Chart>
    <Series>
        <Series Color="91, 151, 213" BackSecondaryColor="74,107,155" Font="{0}, 9.5px" LabelForeColor="59, 59, 59" CustomProperties="PointWidth=0.75, MaxPixelPointWidth=40">
            <SmartLabelStyle Enabled="True" />
            <Points />
        </Series>
        <Series Color="237, 125, 49" BackSecondaryColor="126,153,79" Font="{0}, 9.5px" LabelForeColor="59, 59, 59" CustomProperties="PointWidth=0.75, MaxPixelPointWidth=40">
            <SmartLabelStyle Enabled="True" />
            <Points />
        </Series>
    </Series>
    <ChartAreas>
        <ChartArea BorderColor="White" BorderDashStyle="Solid">
            <AxisY LabelAutoFitMinFontSize="8" TitleForeColor="59, 59, 59" TitleFont="{0}, 10.5px" LineColor="165, 172, 181">
                <MajorGrid LineColor="239, 242, 246" />
                <MajorTickMark LineColor="165, 172, 181" />
                <LabelStyle Font="{0}, 10.5px" ForeColor="59, 59, 59" />
            </AxisY>
            <AxisX LabelAutoFitMinFontSize="8" TitleForeColor="59, 59, 59" TitleFont="{0}, 10.5px" LineColor="165, 172, 181">
                <MajorGrid Enabled="False" />
                <MajorTickMark Enabled="False" />
                <LabelStyle Font="{0}, 10.5px" ForeColor="59, 59, 59" />
            </AxisX>
        </ChartArea>
    </ChartAreas>
    <Legends>

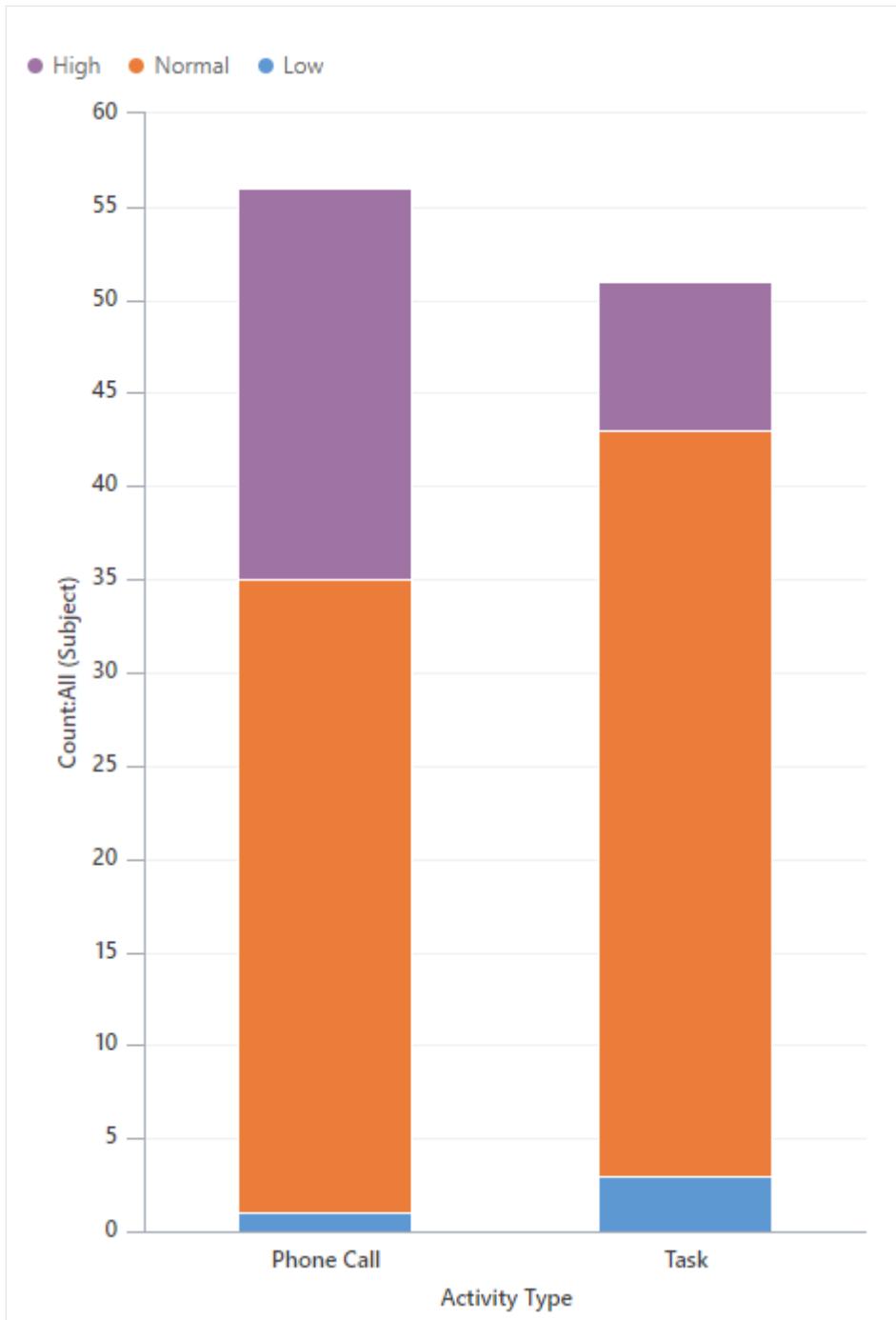
```

```
    <Legend Alignment="Center" LegendStyle="Table" Docking="Bottom" Font="
{0}, 11px" ShadowColor="0, 0, 0, 0" ForeColor="59,59,59" />
</Legends>
<Titles>
    <Title Alignment="TopLeft" DockingOffset="-3" Font="{0}, 13px"
ForeColor="59, 59, 59" />
</Titles>
    <BorderSkin PageColor="Control" BackColor="CornflowerBlue"
BackSecondaryColor="CornflowerBlue" />
</Chart>
```

Comparison chart (Stacked Chart)

The following is a comparison chart that shows the number of activities by type and priority. You can use the chart designer in model-driven apps or methods described in the developer documentation to create these types of charts.

A comparison chart has two `groupby` clauses in the data description XML.



Data description

The following is the contents of the data description XML string for this chart.

```
XML

<datadefinition>
  <fetchcollection>
    <fetch mapping="logical" aggregate="true">
      <entity name="activitypointer">
        <attribute alias="aggregate_column" name="subject" aggregate="count"
      />
        <attribute groupby="true" alias="groupby_column"
          name="activitytypecode" />
        <attribute groupby="true" alias="groupby_priority"
```

```

        name="prioritycode" />
    </entity>
</fetch>
</fetchcollection>
<categorycollection>
    category>
        <measurecollection>
            <measure alias="aggregate_column" />
        </measurecollection>
    </category>
</categorycollection>
</datadefinition>

```

Presentation description

The following is the contents of the presentation description XML string for this chart.

XML

```

<Chart Palette="None" PaletteCustomColors="91,151,213; 237,125,49;
160,116,166; 255,192,0; 68,114,196; 112,173,71; 37,94,145; 158,72,14;
117,55,125; 153,115,0; 38,68,120; 67,104,43; 124,175,221; 241,151,90;
186,144,192; 255,205,51; 105,142,208; 140,193,104; 50,125,194; 210,96,18;
150,83,159; 204,154,0; 51,90,161; 90,138,57;">
    <Series>
        <Series ChartType="StackedColumn" Font="{0}, 9.5px" LabelForeColor="59,
59, 59" CustomProperties="PointWidth=0.75, MaxPixelPointWidth=40">
            </Series>
    </Series>
    <ChartAreas>
        <ChartArea BorderColor="White" BorderDashStyle="Solid">
            <AxisY LabelAutoFitMinFontSize="8" TitleForeColor="59, 59, 59"
TitleFont="{0}, 10.5px" LineColor="165, 172, 181"
IntervalAutoMode="VariableCount">
                <MajorGrid LineColor="239, 242, 246" />
                <MajorTickMark LineColor="165, 172, 181" />
                <LabelStyle Font="{0}, 10.5px" ForeColor="59, 59, 59" />
            </AxisY>
            <AxisX LabelAutoFitMinFontSize="8" TitleForeColor="59, 59, 59"
TitleFont="{0}, 10.5px" LineColor="165, 172, 181"
IntervalAutoMode="VariableCount">
                <MajorGrid Enabled="False" />
                <MajorTickMark Enabled="False" />
                <LabelStyle Font="{0}, 10.5px" ForeColor="59, 59, 59" />
            </AxisX>
        </ChartArea>
    </ChartAreas>
    <Legends>
        <Legend Alignment="Center" LegendStyle="Table" Docking="Bottom" Font="
{0}, 11px" ShadowColor="0, 0, 0, 0" ForeColor="59,59,59">
            </Legend>

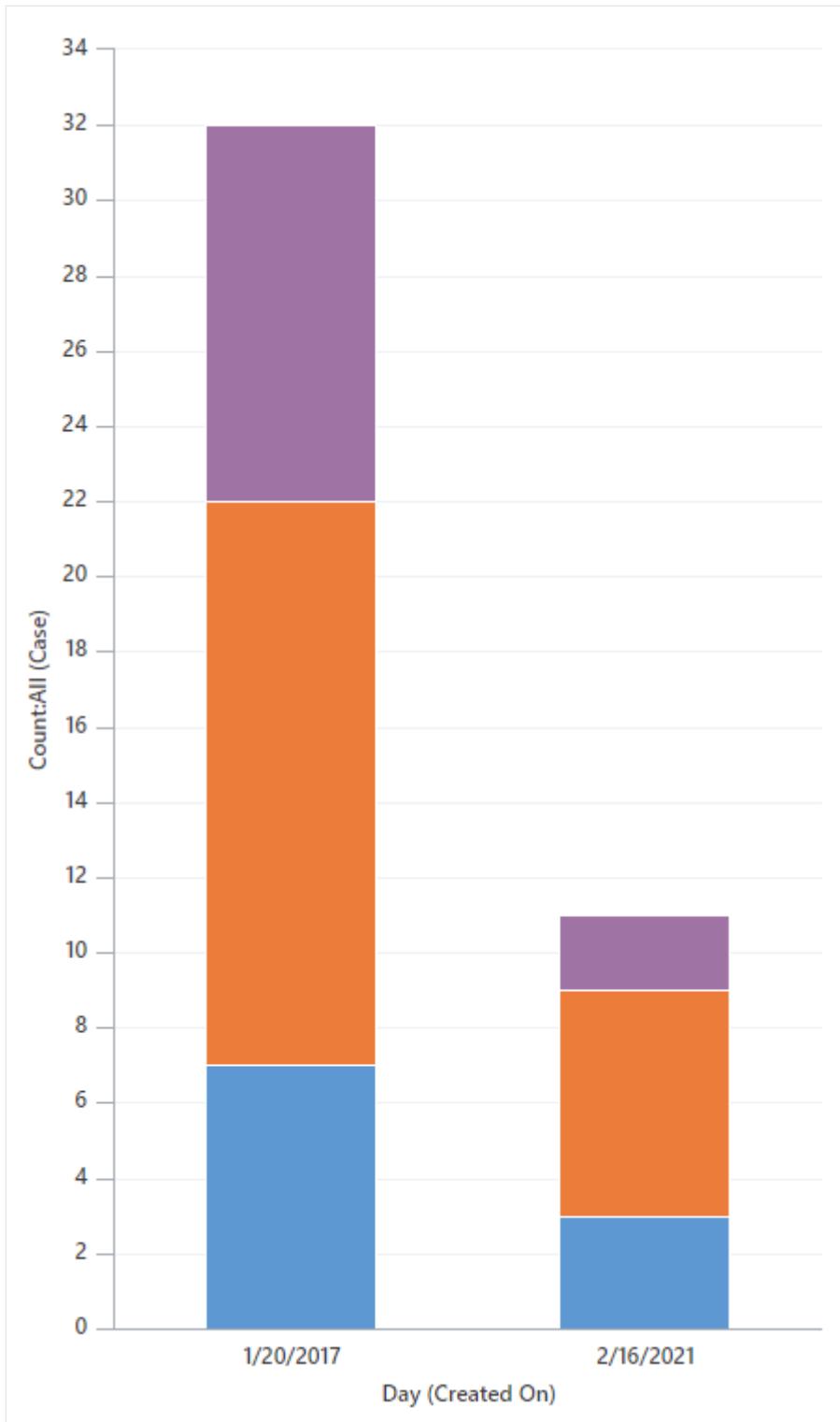
```

```
</Legends>
<Titles>
    <Title Alignment="TopLeft" DockingOffset="-3" Font="{0}, 13px"
ForeColor="59, 59, 59">
        </Title>
    </Titles>
</Chart>
```

Comparison chart (100% Stacked Chart)

The following is a comparison chart that shows the number of cases opened on any date, grouped by priority. You can use the chart designer in model-driven apps or methods available in the Web Services to create these types of charts.

A comparison chart has two `groupby` clauses in the data description XML.



Data description

The following is the contents of the data description XML string for this chart.

XML

```
<datadefinition>
    <fetchcollection>
        <fetch mapping="logical" aggregate="true">
            <entity name="incident">
                <order alias="groupby_column" descending="false" />
```

```

        <attribute alias="aggregate_column" name="incidentid"
aggregate="count" />
        <attribute groupby="true" alias="groupby_column"
dategrouping="day" name="createdon" />
        <attribute groupby="true" alias="groupby_priority"
name="prioritycode" />
    </entity>
</fetch>
</fetchcollection>
<categorycollection>
    <category>
        <measurecollection>
            <measure alias="aggregate_column" />
        </measurecollection>
    </category>
</categorycollection>
</datadefinition>

```

Presentation description

The following is the contents of the presentation description XML string for this chart.

XML

```

<Chart Palette="None" PaletteCustomColors="149,189,66; 197,56,52;
55,118,193; 117,82,160; 49,171,204; 255,136,35; 168,203,104; 209,98,96;
97,142,206; 142,116,178; 93,186,215; 255,155,83">
    <Series>
        <Series ChartType="StackedBar100" Font="{0}, 9.5px"
LabelForeColor="59, 59, 59" CustomProperties="PointWidth=0.75,
MaxPixelPointWidth=40">
            <SmartLabelStyle Enabled="True" />
        </Series>
    </Series>
    <ChartAreas>
        <ChartArea BorderColor="White" BorderDashStyle="Solid">
            <AxisY LabelAutoFitMinFontSize="8" TitleForeColor="59, 59, 59"
TitleFont="{0}, 10.5px" LineColor="165, 172, 181"
IntervalAutoMode="VariableCount">
                <MajorGrid LineColor="239, 242, 246" />
                <MajorTickMark LineColor="165, 172, 181" />
                <LabelStyle Font="{0}, 10.5px" ForeColor="59, 59, 59" />
            </AxisY>
            <AxisX LabelAutoFitMinFontSize="8" TitleForeColor="59, 59, 59"
TitleFont="{0}, 10.5px" LineColor="165, 172, 181"
IntervalAutoMode="VariableCount">
                <MajorGrid Enabled="False" />
                <MajorTickMark Enabled="False" />
                <LabelStyle Font="{0}, 10.5px" ForeColor="59, 59, 59" />
            </AxisX>
        </ChartArea>
    </ChartAreas>

```

```
</ChartAreas>
<Titles>
    <Title Alignment="TopLeft" DockingOffset="-3" Font="{0}, 13px"
ForeColor="0, 0, 0" />
</Titles>
</Chart>
```

See also

[Data visualization and analytics](#)

[Visualization data description schema](#)

[Create a chart](#)

[View data with visualizations \(charts\)](#)

Apply business logic using client scripting in model-driven apps using JavaScript

Article • 11/30/2022

Client-side scripting using JavaScript is one of the ways to apply custom business process logic for displaying data on a form in a model-driven app.

Important

All the client scripting concepts and APIs explained in this documentation also apply to [Dynamics 365 Customer Engagement \(on-premises\)](#) users.

Client scripting shouldn't be your first choice though for applying custom business process logic in model-driven app forms. *Business rules* provide a way for someone, who does not know JavaScript and is not a developer, to apply business process logic in a form. More information: [Create business rules to apply logic](#). You will find the business rule designer within the **Data > Tables > [table_name]** area on [make.powerapps.com](#). When you view table, look for the **Business rules** tab.

However, if your business requirement can't be achieved using a business rule, you will find that client-scripting using the client API object model provides a powerful way to extend the behavior of the application and enable automation in the client.

Use client scripting in model-driven apps

Forms in model-driven apps help display data to the user. A form in model-driven apps can contain items such as columns, a quick form, or a grid. An [event](#) occurs in model-driven apps forms whenever:

- A form loads.
- Data is changed in a column or an item within the form.
- Data is saved in a form.

You can attach your JavaScript code to "react" to these events so that your code gets executed when the event occurs on the form. You attach your JavaScript code (scripts) to these events by using a [Script web resource](#) in model-driven apps.

Model-driven apps provides you a rich set of **client APIs** to interact with form objects and events to control what and when to display on a form.

 **Note**

Some client APIs are deprecated in the current release of model-driven apps. Ensure that you are aware of these APIs as you write your client-side code for model-driven apps. More information: [Deprecated client APIs](#)

Get started here

[Events in forms and grids](#)

[Understand the Client API object model](#)

[Walkthrough: Write your first client script](#)

Reference

[Client API reference](#)

Related topics

[Web resources for model-driven apps](#)

[Customize commands and the ribbon](#)

Events in forms and grids in model-driven apps

Article • 11/30/2022

All client-side code is initiated by events. In model-driven apps, you associate a specific function in a JavaScript library ([Script web resource](#)) to be executed when an event occurs. This function is called an *event handler*. Each event handler specifies a single function within a JavaScript library and any parameters that can be passed to the function.

You can associate event handlers to only some events using the UI. For events that are not available to be associated through UI, Client API provides methods that can be used to attach event handlers to such events.

Add or remove event handler function to event using UI

Legacy

Use the **Event Handlers** section of the **Form Properties** dialog box to associate your script with an event for forms and columns.

Form Properties

Modify this form's properties.

The screenshot shows the 'Events' tab selected in the 'Form Properties' dialog. Below it is a table with columns 'Name', 'Display Name', and 'Description'. Under the 'Event Handlers' section, there is a heading 'Manage functions that are called for form or field events.' A red box highlights the 'Control' dropdown set to 'Form' and the 'Event' dropdown set to 'OnLoad'. Below this is another table with columns 'Library', 'Function', and 'Enabled'.

Bulk edit forms

By default, events handlers aren't called when a form is in bulk edit mode.

To enable an event handler in bulk edit mode, modify the Form XML by finding the relevant `event` element and creating/setting the `BehaviorInBulkEditForm` attribute to `Enabled`. Currently, this is only supported for [OnLoad events](#).

For more information on Form XML customization, see [When to edit the customizations file](#), [Customize forms](#), and the [Form XML schema](#).

To determine when an event handler is called on a form in bulk edit mode use `getFormType` method.

Add or remove event handler function to event using code

Using the following methods to add and remove event handler for events that cannot be associated through UI:

Events	Event handler
Attribute OnChange	addOnChange and removeOnChange methods
Form OnLoad	formContext.ui addOnLoad and removeOnLoad methods
Form data OnLoad	formContext.data addOnLoad and removeOnLoad methods
Form OnSave	addOnSave and removeOnSave methods
Lookup control PreSearch	addPreSearch and removePreSearch methods
kbsearch control OnResultOpened	addOnResultOpened and removeOnResultOpened methods
kbsearch control OnSelection	addOnSelection and removeOnSelection methods
kbsearch control PostSearch	addOnPostSearch and removeOnPostSearch methods

 **Important**

The execution context is automatically passed as the first parameter to functions that are set using the code. More information: [Client API execution context](#)

Form event pipeline

You can define up to 50 event handlers for each event. Each event handler is executed in the order that it is displayed in the **Event Handlers** section in the **Events** tab of the **Form Properties** dialog box.

Use the [setSharedVariable](#) and [getSharedVariable](#) methods to pass a common variable between event handlers (functions). Use the execution context [getDepth](#) method to know the sequence that an event handler is being executed in relative to other event handlers.

Related topics

[Understand the Client API object model](#)

[Client API execution context](#)

[Events \(Client API reference\)](#)

Understand the Client API object model

Article • 12/16/2022

The Client API object model for model-driven apps provides you objects and methods that you can use to apply custom business logic in model-driven apps using JavaScript, such as:

- Get or set column values.
- Show and hide user interface elements.
- Reference multiple controls per column.
- Access multiple forms per table.
- Manipulate form navigation items.
- Interact with the business process flow control.

It's important that you understand the model-driven apps Client API object model to effectively write and use your JavaScript code.

Root objects in the Client API object model

At the root of the Client API object model are the following contexts and the Xrm object:

Object	Description
executionContext	Represents the execution context for an event in model-driven apps forms and grids. More information: Client API execution context
formContext	Provides a reference to a form or an item on the form against which the current code executes. To get the formContext object, use the executionContext.getFormContext method. More information: Client API form context
gridContext	Provides a reference to a grid or a subgrid on a form against which the current code executes. More information: Client API grid context
Xrm	Provides a global object for performing operations that do not directly impact the data and UI in forms, grids, subgrids, controls, or columns. For example, navigate forms, create and manage records using Web API. More information: Client API Xrm object

Related topics

[Client API global context](#)

[Client API reference](#)

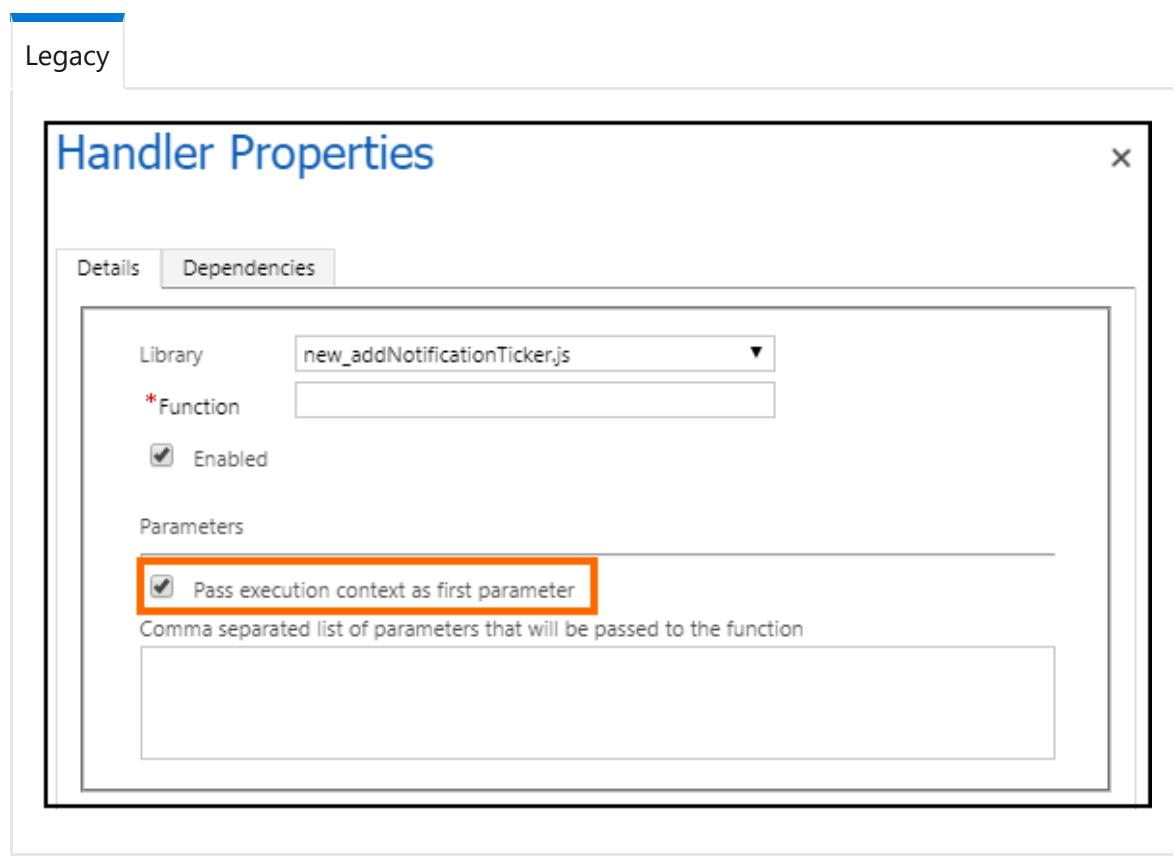
Client API execution context

Article • 12/16/2022

The execution context defines the event context in which your code executes. The execution context is passed when an event occurs on a form or grid, which you can use it in your event handler to perform various tasks such as determine [formContext](#) or [gridContext](#), or manage the save event.

The execution context is passed in one of the following ways:

- **Defining event handlers using UI:** The execution context is an *optional* parameter that can be passed to a JavaScript library function through an event handler. Use the **Pass execution context as first parameter** option in the **Handler Properties** dialog while specify the name of the function to pass the event execution context. The execution context is the first parameter passed to a function.



- **Defining event handlers using code:** The execution context is automatically passed as the first parameter to functions set using code. For a list of methods that can be used to define event handlers in code, see [Add or remove functions to events using code](#).

The execution context object provides a number of methods to further work with the context. More information: [Execution context \(Client API reference\)](#)

Related topics

[Client API form context](#)

[Client API grid context](#)

[Form and grid context in ribbon actions](#)

Client API form context

Article • 11/30/2022

The Client API form context (`formContext`) provides a reference to the form or to an item on the form, such as, a quick view control or a row in an editable grid, against which the current code is executed.

Earlier, the global `Xrm.Page` object was used to represent a form or an item on the form. With the latest version, the `Xrm.Page` object is [deprecated](#), and you should use the [getFormContext](#) method of the passed in execution context object to return reference to the appropriate form or an item on the form.

Important

Deprecated means that we intend to remove a feature or capability from a future major release of model-driven apps; the feature or capability will continue to work and is fully supported until it is officially removed. A public announcement here in the documentation, on the official blog, and in many other places will be made at least six months before removal.

Use of the `Xrm.Page` object as a static access to the primary form context is *still* supported to maintain backward compatibility with the existing scripts, and won't be removed as soon as some other client API methods listed in the [Client API deprecation](#) section. We recommend that you use the new `formContext` object instead of the `Xrm.Page` object in your code targeting version 9.0 or later where possible. Also, using the `formContext` object enables you to create common event handlers that can operate either on a form or in an editable grid depending on where its called. More information: [getFormContext \(Client API reference\)](#).

Getting the `formContext` object for JavaScript functions for ribbon actions is different from how you get it in form scripting. More information: [Form and grid context in ribbon actions](#).

Using the `formContext` object instead of the `Xrm.Page` object

It's easy to convert existing code with `Xrm.Page` to use the new `formContext` object. For example, consider the following script that uses the `Xrm.Page` object:

JavaScript

```
function displayName()
{
    var firstName = Xrm.Page.getAttribute("firstname").getValue();
    var lastName = Xrm.Page.getAttribute("lastname").getValue();
    console.log(firstName + " " + lastName);
}
```

Here is the updated script that uses the passed in execution context to retrieve the **formContext** object instead of using the static **Xrm.Page** object:

JavaScript

```
function displayName(executionContext)
{
    var formContext = executionContext.getFormContext(); // get formContext

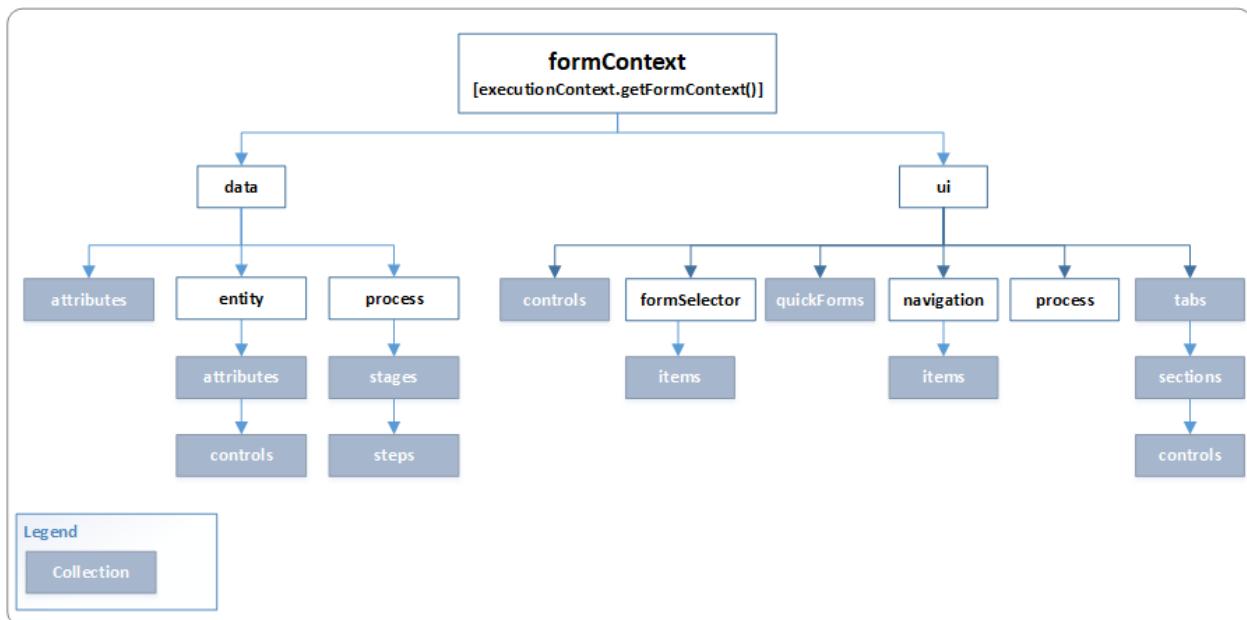
    // use formContext instead of Xrm.Page
    var firstName = formContext.getAttribute("firstname").getValue();
    var lastName = formContext.getAttribute("lastname").getValue();
    console.log(firstName + " " + lastName);
}
```

ⓘ Important

You must remember to select the **Pass execution context as first parameter** option in the **Handler Properties** dialog while defining your event handlers to use the **formContext** object. More information: [Client API execution context](#)

formContext object model

Use the **data** and **ui** objects under the **formContext** object to programmatically manipulate data and user interface elements in model-driven apps.



data object

Provides properties and methods to work with the data on a form, including table data and data in the business process flow control. Contains the following objects:

Object	Description
attributes	Collection of non-table data on the form. Items in this collection are of the same type as the column collection, but they are not columns of the form table. More information: Collections
entity	Provides methods to retrieve information specific to the record displayed on the page, the save method, and a collection of all the columns included on the form. Column data is limited to columns represented on the form. More information: formContext.data.entity
process	Provides objects and methods to interact with the business process flow data on a form. More information: formContext.data.process

It also provides an **attributes** collection for accessing non-table bound control. See the **Collections in the formContext object model** section later in this topic.

More information: [formContext.data](#)

ui object

Provides methods to retrieve information about the user interface, in addition to collections for several sub components of the form or grid. Contains the following objects:

Object	Description
formSelector	Provides an items collection that provides capabilities to query the forms available for the current user. Use the navigate method to close the current form and open a different one.
navigation	Does not contain any methods. Provides access to navigation items through the items collection. See the next section on collections for more information.
process	Provides methods to interact with the business process flow control on a form.

More information: [formContext.ui](#)

Collections in the `formContext` object model

The following table describes the collections in `Xrm` object model. For information about the methods available for collections in general, see [Collections \(Client API reference\)](#).

Collection	Description
<code>attributes</code>	<p>Two objects contain a column collection:</p> <ul style="list-style-type: none"> - <code>formContext.data.attributes</code> collection provides access to non-table bound columns. - <code>formContext.data.entity.attributes</code> collection provides access to each table column that is available on the form. Only those columns added to the form are available.
<code>controls</code>	<p>Three objects contain a controls collection:</p> <ul style="list-style-type: none"> - <code>formContext.ui.controls</code>: Provides access to each control present on the form. - <code>formContext.data.entity.attribute.controls</code>: Because a column may have more than one control on the form, this collection provides access to each of them. This collection will contain only one item unless multiple controls for the column are added to the form. - <code>formContext.ui.tabs.sections.controls</code>: This collection only contains the controls found in the section.
<code>formContext.data.process.stages</code> and <code>formContext.data.process.steps</code>	Provides access to stages and steps collection in a business process flow. These also allow for adding and removing of items from the collection.

Collection	Description
<code>formContext.ui.formSelector.items</code>	When multiple forms are provided for a table, you can associate each form with security roles. When the security roles associated with a user enable them to see more than one form, the <code>formContext.ui.formSelector.items</code> collection provides access to each form definition available to that user.
<code>formContext.ui.navigation.items</code>	The <code>formContext.ui.navigation.items</code> collection provides access to navigation items that are defined using the navigation area of the form editor. People navigate to these using the command bar.
<code>formContext.ui.quickForms</code>	Provides methods to access all the quick view controls and its constituent controls on the Customer Engagement forms.
<code>formContext.ui.tabs</code>	You can organize each form by using one or more tabs. This collection provides access to each of these tabs.
<code>formContext.ui Tab.sections</code>	You can organize each form tab by using one or more sections. The tab <code>sections</code> collection provides access to each of these sections. You need to define the tab which contains the desired section or iterate through each tab to find the relevant section.

Related topics

- [getFormContext method](#)
- [getGlobalContext method](#)
- [getAttribute method](#)
- [getControl method](#)
- [Execution context methods](#)

Client API grid context

Article • 12/16/2022

Grids present data in a tabular format. Grids can span the entire form or can be one of the items on a form; the latter are called **subgrids**.

The Client API grid context object provides reference to a subgrid on a form against which the current code is executed.

ⓘ Note

Getting the context of a grid (spanning the entire form) is only supported in ribbon commands. More information: [Form and grid context in ribbon actions](#)

Use the [formContext](#) object to get an instance of the form where the code is executed, and then retrieve the subgrid control on the form. For example, when you know the name of a subgrid control (say **Contacts** subgrid in the default account form), you can access it using the following code:

JavaScript

```
function doSomething(executionContext) {
    var formContext = executionContext.getFormContext(); // get the form
    Context
    var gridContext = formContext.getControl("Contacts"); // get the grid
    context

    // Perform operations on the subgrid
}
```

Related topics

[Client API form context](#)

[Client API execution context](#)

[Understand the Client API object model](#)

[Grids and subgrids](#)

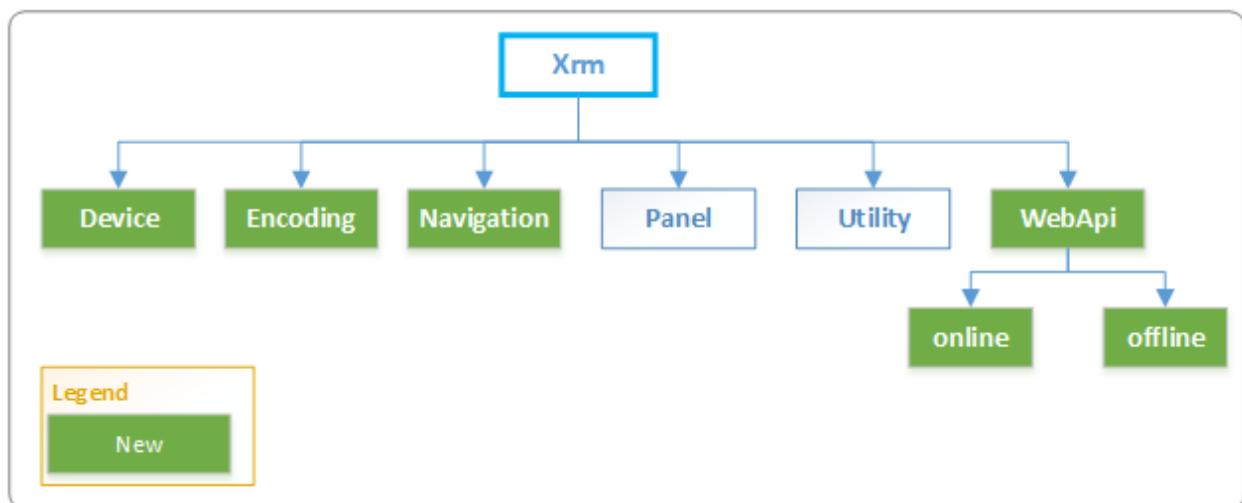
Client API Xrm object

Article • 12/16/2022

The **Xrm** object is globally available to use in your code without having to use the execution context in Client API.

Xrm object model

The following illustration displays the Xrm object model:



Here is the information about each of the namespaces in the Xrm object:

Namespace	Description
Xrm.Device	Provides methods to use native device capabilities.
Xrm.Encoding	Provides methods to encode strings.
Xrm.Navigation	Provides methods for navigating forms and items in model-driven apps.
Xrm.Panel	Provides a method to display a web page in the side pane of model-driven apps form.
Xrm.Utility	Provides a container for useful methods.

Namespace	Description
Xrm.WebApi	<p>Provides methods to use Web API to create and manage records and execute Web API actions and functions.</p> <p>Xrm.WebApi.offline: Provides methods to create and manage records in the model-driven apps in mobile clients while working in <i>offline</i> mode.</p> <p>Xrm.WebApi.online: Provides methods to use Web API to create and manage records and execute Web API actions and functions when connected to the server.</p>

Client API global context

Use the [Xrm.Utility.getGlobalContext](#) method in forms to retrieve information specific to an organization, a user, or the client where script is run without going through the form execution context. This is a change from previous versions where you had to use the form context to retrieve global context by using [Xrm.Page.context](#).

 Note

[Xrm.Page.context](#) is deprecated in the current release, and you should now use the new [Xrm.Utility.getGlobalContext](#) method to get global context in your code targeting version 9.0 or later.

To access the global context information in a standalone HTML Web resource, you should include a reference to [ClientGlobalContext.js.aspx](#) in the web resource, and then use the [GetGlobalContext](#) function. More information: [GetGlobalContext function and ClientGlobalContext.js.aspx](#)

Related topics

[Understand the Client API object model](#)

[Deprecated client APIs](#)

Walkthrough: Write your first client script

Article • 02/02/2023

Ready to write your first client script to see things in action? Lets get started. We'll keep it simple.

Objective

After completing this walkthrough, you will know how to use JavaScript code in model-driven apps, which involves the following steps at a high level:

- Locate or create a solution to add your work to
- Write JavaScript code to address a business issue
- Upload your code as a web resource
- Associate your web resource to a form
- Configure form and field events
- Test your code

Step 1: Locate or create a solution

Solutions are the way that customizations can be transported from one environment to another. You should write and test your Javascript code in a development environment as part of an unmanaged solution. When you have finished testing, export the solution as a managed solution and import/install it in your production environment.

There may already be an existing solution that you should use. The model-driven app you want to customize with your script should already be part of a solution. You may edit that solution or create a new solution that depends on another existing solution.

To create a new solution:

1. Navigate to [Power Apps](#).
2. In the left navigation pane, select **Solutions** and then select **New solution**.
3. In the fly-out dialog, specify **Display name**, **Name**, and **Publisher**.

Most companies will have an existing publisher that they will always use. You should use that publisher. If you are the very first person, you get to create your

own.

Click **New Publisher** to open the **New Publisher** dialog. In this walkthrough, we will use a publisher with this definition:

New publisher

Publishers indicate who developed associated solutions. [Learn more](#)

[Properties](#) [Contact](#)

Display name *

Name *

Description

Prefix *

Choice value prefix *

Preview of new object name

example_Object

Save **Cancel**

Notice the **Prefix** value. This should be something that identifies your company. In this case, we are using `example`.

This is the solution we will use in this walkthrough

New solution X

Display name *

Name *

Publisher *

▼ Edit

+ New publisher

Version *

More options ▾

Create Cancel

4. Locate or add a model-driven app to your solution.

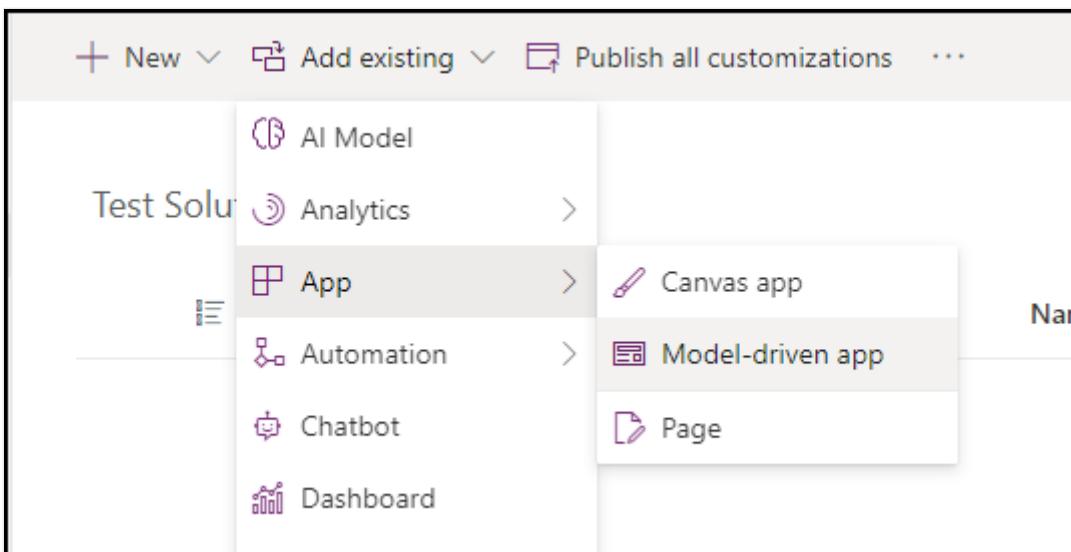
A new solution will not have any model-driven apps included in it. You will need to add an existing model-driven app or create a new one.

! Note

For the purpose of this walkthrough, make sure the app includes the Account table. The scripts and instructions below expects fields found in a form for the Account table.

Option A: Add an existing model-driven app to your solution

- a. In your solution, select **Add existing > App > Model-driven app**.



- a. Select an existing app and click **Add**.

Option B: Create a new model-driven app in your solution

In your solution, you can select **New > App > Model-driven app**.

See the [Build your first model-driven app](#) tutorial. Create an app that includes the Account table.

Step 2: Write your JavaScript code

The first step is to identify the business issue you are trying to address using client scripting. Once you have identified it, you need to write your JavaScript code containing the custom business logic that addresses your business issue.

Model-driven apps do not provide a JavaScript editor. Use an external authoring tool that provides features to specifically support editing JavaScript files, such as [Notepad++](#), [Visual Studio Code](#), or [Microsoft Visual Studio](#).

This is the JavaScript code that this walkthrough uses:

```
JavaScript

// A namespace defined for the sample code
// As a best practice, you should always define
// a unique namespace for your libraries
var Example = window.Example || {};
(function () {
    // Define some global variables
    var myUniqueId = "_myUniqueId"; // Define an ID for the notification
    var currentUserName =
        Xrm.Utility.getGlobalContext().userSettings.userName; // get current user
    name
    var message = currentUserName + ": Your JavaScript code in action!";
```

```

// Code to run in the form OnLoad event
this.formOnLoad = function (executionContext) {
    var formContext = executionContext.getFormContext();

    // Display the form level notification as an INFO
    formContext.ui.setFormNotification(message, "INFO", myUniqueId);

    // Wait for 5 seconds before clearing the notification
    window.setTimeout(function () {
        formContext.ui.clearFormNotification(myUniqueId); }, 5000);
    }

// Code to run in the column OnChange event
this.attributeOnChange = function (executionContext) {
    var formContext = executionContext.getFormContext();

    // Automatically set some column values if the account name contains
    "Contoso"
    var accountName = formContext.getAttribute("name").getValue();
    if (accountName.toLowerCase().search("contoso") != -1) {

        formContext.getAttribute("websiteurl").setValue("https://www.contoso.com");
        formContext.getAttribute("telephone1").setValue("425-555-0100");
        formContext.getAttribute("description").setValue("Website URL,
Phone and Description set using custom script.");
    }
}

// Code to run in the form OnSave event
this.formOnSave = function () {
    // Display an alert dialog
    Xrm.Navigation.openAlertDialog({ text: "Record saved." });
}
}.call(Example);

```

Copy this code into a text file and save it with the name: `Example-form-script.js`.

Detailed code explanation

Let's look at the code in detail:

- **Define namespace:** The code starts by defining a namespace for your custom script. As a best practice, you should always create namespaced JavaScript libraries to avoid having your functions overridden by functions in another library.

JavaScript

```
var Example = window.Example || {};
```

In this case, all the functions defined in this library can be used as `Example.`

`[functionName]`. You should choose a namespace that matches your solution publisher name.

- **Define global variables:** The following section defines some global variables to be used in the script. Context information is available globally using the `Xrm.Utility.getGlobalContext` method.

JavaScript

```
// Define some global variables
var myUniqueId = "_myUniqueId"; // Define an ID for the notification
var currentUserName =
Xrm.Utility.getGlobalContext().userSettings.userName; // get current
user name
var message = currentUserName + ": Your JavaScript code in action!";
```

- **Function to execute on the OnLoad event:** This section contains the function that will be executed when the account form loads. For example, when you create a new account record or when you open an existing account record.

The `Example.formOnLoad` function uses the `executionContext` parameter to get the `formContext` object. When you attach your code with the form event later, you must remember to select the option to pass the `execution context` to this function.

This function displays a form level notification using the `formContext.ui.setFormNotification` method.

Finally, this function uses the JavaScript [setTimeOut method](#) to delay the execution of the `formContext.ui.clearFormNotification` method to clear the notification after 5 seconds.

JavaScript

```
// Code to run in the form OnLoad event
this.formOnLoad = function (executionContext) {
    var formContext = executionContext.getFormContext();

    // Display the form level notification as an INFO
    formContext.ui.setFormNotification(message, "INFO", myUniqueId);

    // Wait for 5 seconds before clearing the notification
    window.setTimeout(function () {
        formContext.ui.clearFormNotification(myUniqueId); }, 5000);
}
```

- **Function to execute on the OnChange event:** The `Example.attributeOnChange` function will be associated with the **Account Name** column in the account form so that it gets executed **only** when you change the account name value.

This function performs a case-insensitive search for `Contoso` in the account `name`, and if present, sets values for the `websiteurl`, `telephone1`, and `description` columns in the account form.

JavaScript

```
// Code to run in the column OnChange event
this.attributeOnChange = function (executionContext) {
    var formContext = executionContext.getFormContext();

    // Automatically set some column values if the account name
    // contains "Contoso"
    var accountName = formContext.getAttribute("name").getValue();
    if (accountName.toLowerCase().search("contoso") != -1) {

        formContext.getAttribute("websiteurl").setValue("https://www.contoso.co
        m");
        formContext.getAttribute("telephone1").setValue("425-555-
        0100");
        formContext.getAttribute("description").setValue("Website URL,
        Phone and Description set using custom script.");
    }
}
```

- **Function to execute on the OnSave event:** The `Example.formOnSave` function displays an alert dialog box using the `Xrm.Navigation.openAlertDialog` method. This dialog box displays a message with an **OK** button. The user can close the alert by clicking **OK**.

(!) Note

This function doesn't use the execution context because the `Xrm.Navigation`. methods don't require them.

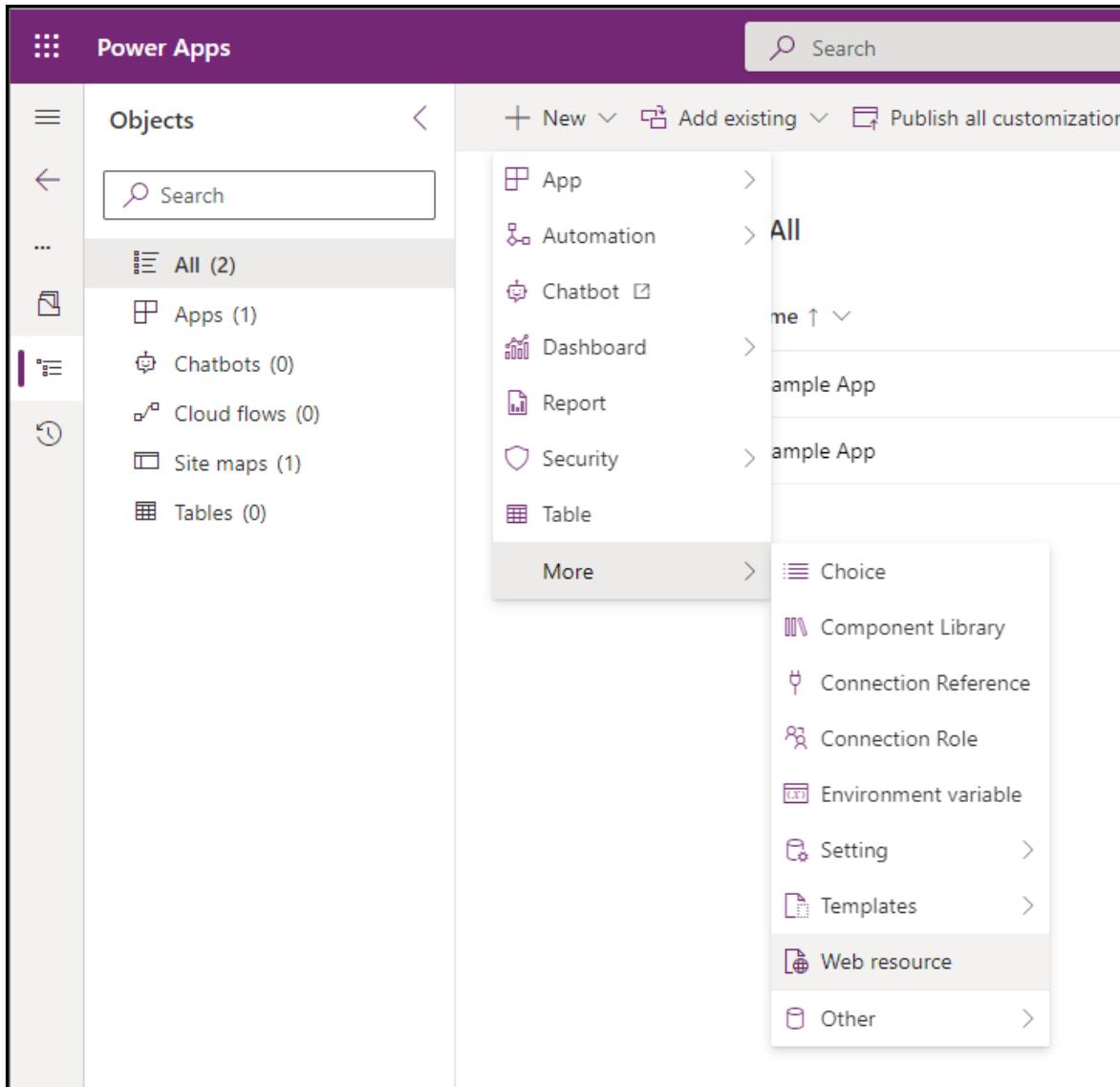
JavaScript

```
// Code to run in the form OnSave event
this.formOnSave = function () {
    // Display an alert dialog
    Xrm.Navigation.openAlertDialog({ text: "Record saved." });
}
```

Step 3: Upload your code as a web resource

Now that your code is ready, you need to upload it into your solution.

1. In your solution select **New > More > Web resource**



2. In the **New web resource** dialog, click **Choose file** and select the `Example-form-script.js` file you saved earlier.

3. Type in the **Display name**, **Name**, and optionally a **Description**. Make sure the **Type** is **JavaScript (JS)**.

New web resource

Upload a file *

[Choose file](#)

Example-form-script.js

Display name

Example Script

Name *

example_ example-form-script

Type *

JavaScript (JS)

Description

Example script used for the Write your first client script walkthrough.

[Advanced options](#) ▾

[Save](#)

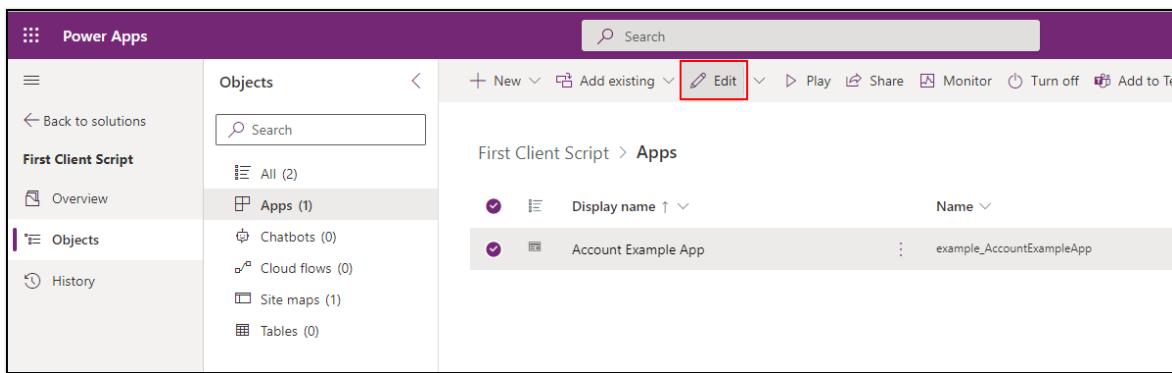
[Cancel](#)

ⓘ Note

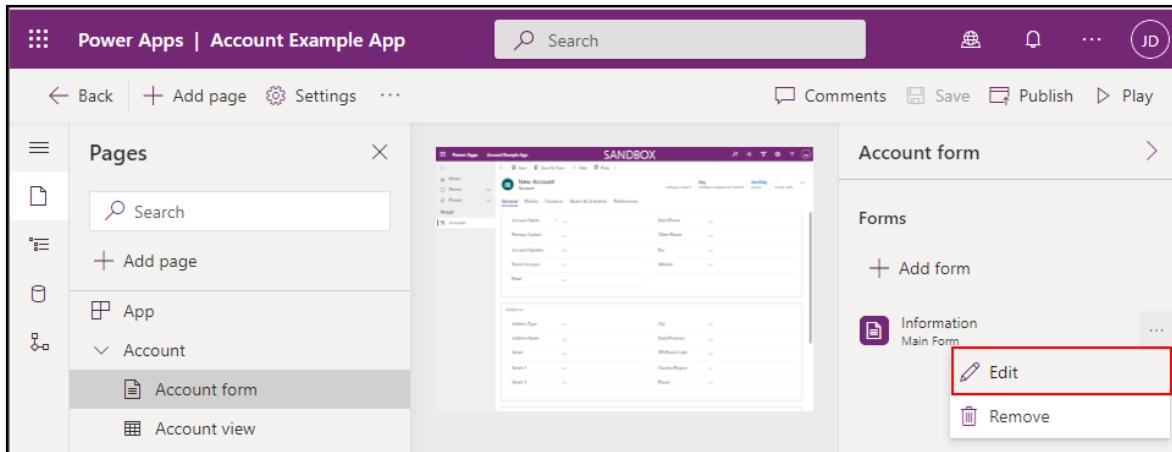
- Notice how the **Name** has a prefix that matches the solution publisher customization prefix. There are other ways to create web resources, but creating a web resource this way ensures that the Web Resource is part of your solution.
- The name of the web resource is `example_example-form-script`.

Step 4: Associate your web resource to a form

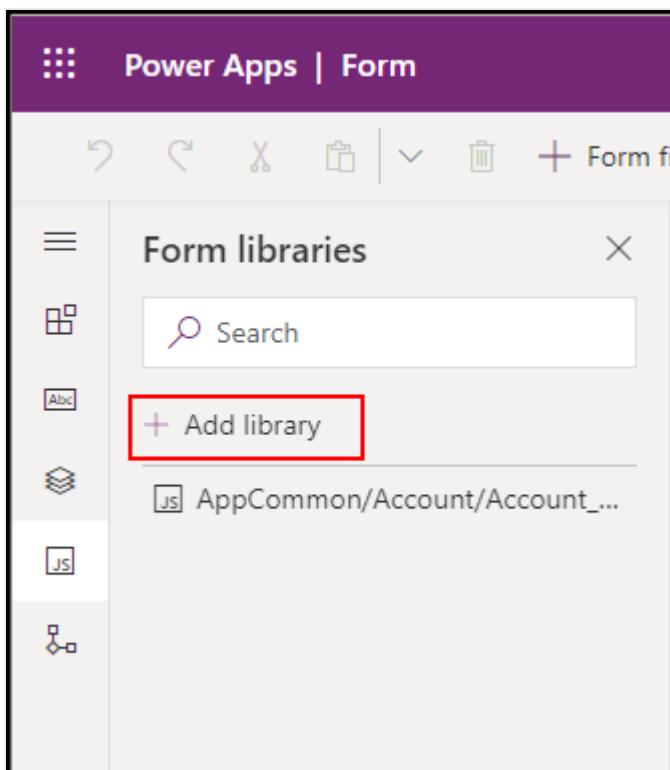
1. In your solution, select **Objects > Apps > Your App** and click **Edit**.



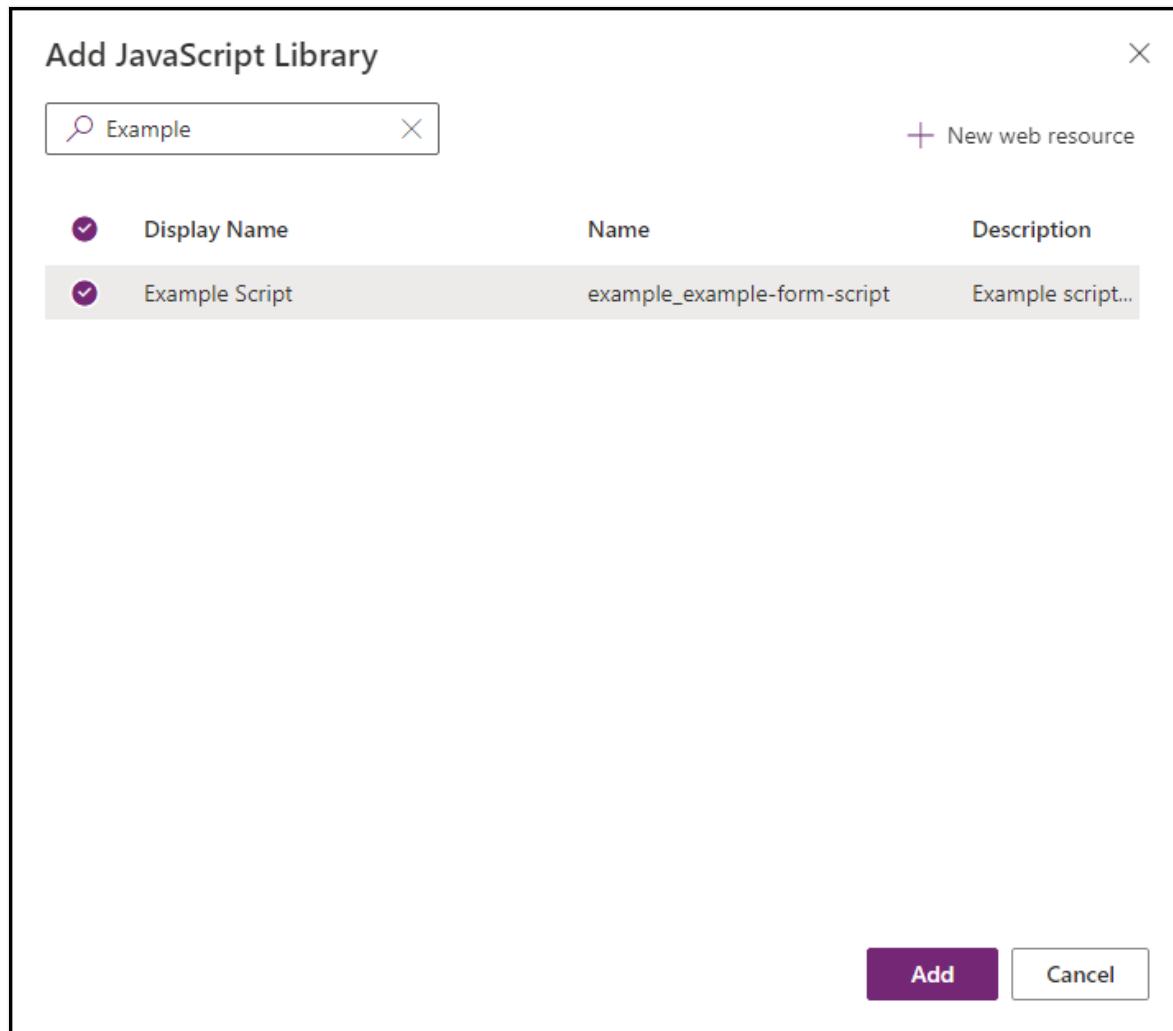
2. Expand Account and select the Account form, click the ellipses (...) to the right of the Information form and select Edit.



3. In the left navigation, select Form Libraries and click Add library.



4. In the Add JavaScript Library dialog, search for the JavaScript web resource you created by name: Example Script.



5. Select the **Example Script** web resource and click **Add**.

Step 5: Configure form and field events

1. Select the **Events** tab.

The screenshot shows the "New Account" form configuration page. The "General" tab is selected in the top navigation bar. On the right side, there is an "Information" section with tabs for "Main form", "Properties", and "Events". The "Events" tab is currently selected and highlighted with a red border. Under the "Events" tab, there are sections for "On Save" and "On Load", each containing a "+ Event Handler" button.

Configure form On Load event

1. Select **On Load** event handler and click **+ Event Handler**.

The screenshot shows the Oracle Forms configuration interface. On the left, the 'Configure Event' dialog is open, showing settings for an 'On Load' event. The 'Function' field contains 'Example.form onLoad'. Under 'Properties', 'Enabled' is checked. Under 'Events', the 'On Load' section is expanded, and a new 'Event Handler' button is highlighted with a red box. On the right, the event tree shows 'Information Main form' with 'Properties' and 'Events' tabs selected. The 'Events' tab shows 'On Save' and 'On Load' sections, each with its own 'Event Handler' button.

Configure Event

Event Type
On Load

Library
example_example-form-script

Edit library

Function *
Example.formOnLoad

Enabled

Pass execution context as first parameter

Comma separated list of parameters that will be passed to the function

Done Cancel

Information Main form

Properties Events

On Save

+ Event Handler

On Load

+ Event Handler

Make sure that:

- The Event Type is On Load.
 - The example_example-form-script library is selected.
- a. Type the name of the function in the Function field. In this case `Example.formOnLoad`.
 - b. Select Pass execution context as first parameter.
 - c. Click Done.

Configure Form On Save event

1. Select On Save event handler and click + Event Handler.

The screenshot shows two parts of a software interface. On the left is a modal dialog titled "Configure Event". It has a "Event Type" dropdown set to "On Save", a "Library" dropdown set to "example_example-form-script" with an "Edit library" link, a "Function *" input field containing "Example.formOnSave" (which is underlined in red), and a checked "Enabled" checkbox. Below these are two optional checkboxes: "Pass execution context as first parameter" (unchecked) and a text area for parameters (empty). At the bottom are "Done" and "Cancel" buttons. On the right is the main application window showing the "Information" section for a "Main form". The "Events" tab is selected, displaying two event sections: "On Save" and "On Load". The "On Save" section contains an "Event Handler" entry, which is highlighted with a red box. The "On Load" section also contains an "Event Handler" entry. Both sections have a "Handlers" list with "Example.formOnLoad".

Make sure that:

- The **Event Type** is **On Save**.
- The **example_example-form-script** library is selected.

a. Type the name of the function in the **Function** field. In this case

`Example.formOnSave`.

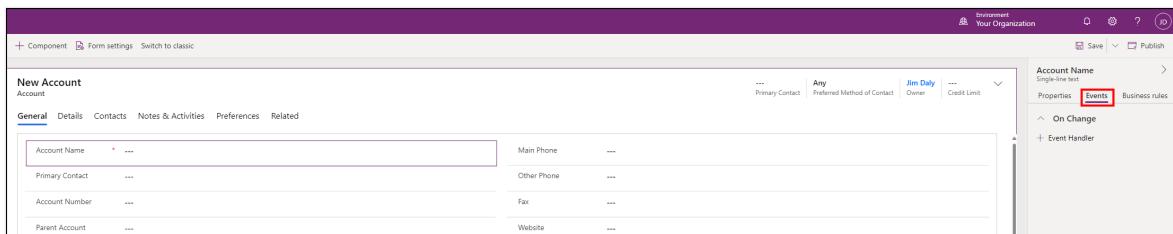
ⓘ Note

It is not necessary to elect **Pass execution context as first parameter** for this function because it doesn't use it.

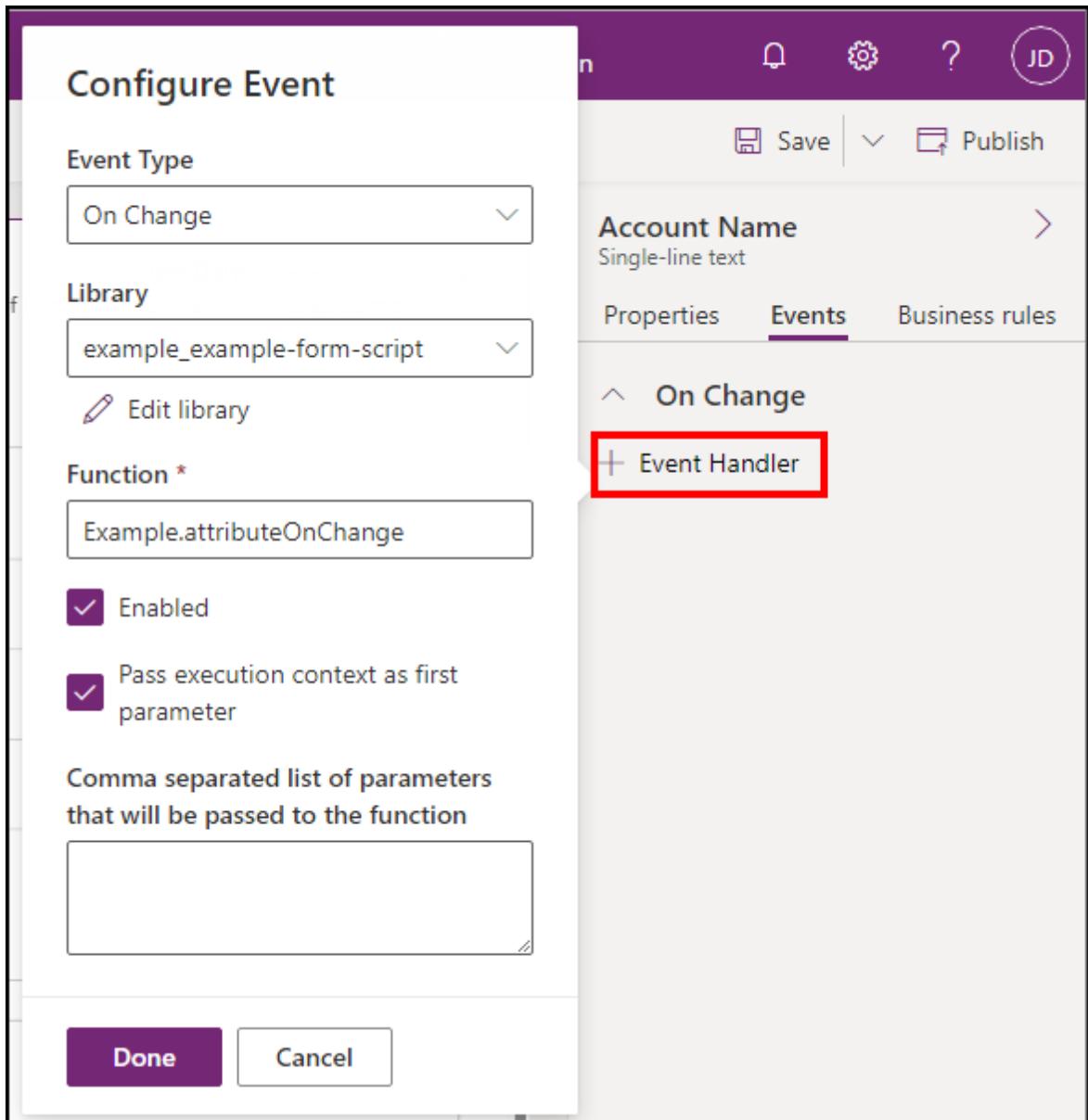
b. Click **Done**

Configure Field On Change event

1. Select the Account Name field and the Events tab.



2. Select the On Change event handler and click + Event Handler.



Make sure that:

- The Event Type is On Change.
- The example_example-form-script library is selected.

a. Type the name of the function in the Function field. In this case

Example.attributeOnChange.

b. Select Pass execution context as first parameter.

c. Click Done

Save and publish your changes

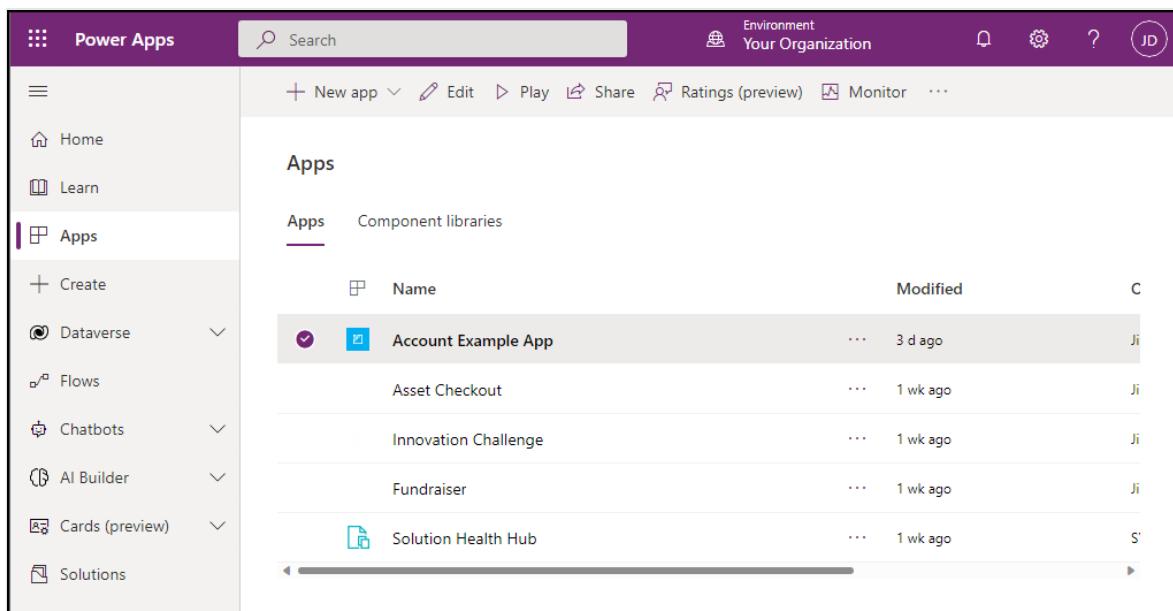
Save the form and click Publish.

Step 6: Test your code

It is recommended that you refresh your browser for the changes to take effect in your model-driven apps instance.

To test your code:

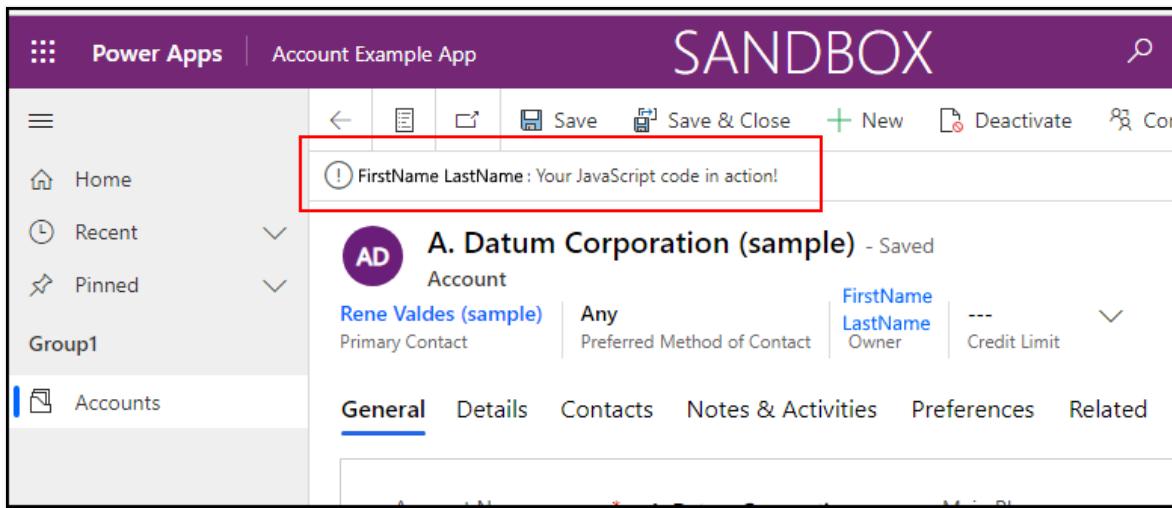
1. Navigate to [Power Apps](#).
2. In the left navigation area select **Apps**.
3. Double-click the model-driven app you just edited, or select it and click **Play**.



The screenshot shows the Microsoft Power Apps portal interface. The top navigation bar includes 'Power Apps', a search bar, 'Environment Your Organization', and user profile icons. The left sidebar has a 'Navigation' menu with 'Home', 'Learn', 'Apps' (which is selected), 'Create', 'Dataaverse', 'Flows', 'Chatbots', 'AI Builder', 'Cards (preview)', and 'Solutions'. The main content area is titled 'Apps' and lists several applications: 'Account Example App' (selected, indicated by a checkmark icon), 'Asset Checkout', 'Innovation Challenge', 'Fundraiser', and 'Solution Health Hub'. Each item in the list has a 'Name', 'Modified' timestamp ('3 d ago', '1 wk ago', '1 wk ago', '1 wk ago'), and a small circular icon with initials ('Ji', 'Ji', 'Ji', 'S').

Test form On Load function

1. Click on any account record in the list to open it.
2. Verify that the notification appears.



3. Verify the notification disappears in 5 seconds.

Test field On Change function

1. Edit the Account Name to include "Contoso" in the name and move to the next column by pressing TAB.
2. Verify the expected values set to the Main Phone, Website, and Description columns.

Contoso Pharmaceuticals (sample) - Unsaved

Account

Primary Contact: Robert Lyon (sample) | Preferred Method of Contact: Any | Owner: [Redacted] | Credit Limit: ---

General Details Contacts Notes & Activities Preferences Related

Account Name: * Contoso Pharmaceuticals	Main Phone: 425-555-0100
Primary Contact: Robert Lyon...	Other Phone: ---
Account Number: ---	Fax: ---
Parent Account: ---	Website: https://www.contoso.com
Email: someone7@example...	

Address

Address Type: ---	City: Redmond
Address Name: ---	State/Province: WA
Street: 9906 Oak Grove Road	ZIP/Postal Code: 80803
Street 2: ---	Country/Region: U.S.
Street 3: ---	Phone: ---

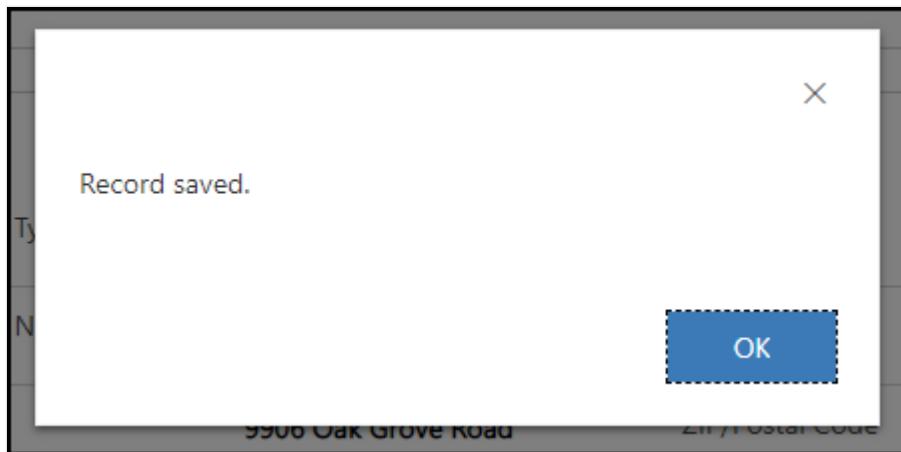
Shipping Method: --- Freight Terms: ---

Description

Website URL, Phone and Description set using custom script.

Test form On Save function

1. Click Save.
2. Verify that the alert dialog with a message that you configured in your code.



3. Click **OK** to close the alert.

See also

- [Debug JavaScript code for model-driven apps](#)
- [Events in forms and grids in model-driven apps](#)
- [Form OnLoad event](#)
- [Form OnSave event \(Client API reference\) in model-driven apps](#)
- [Column OnChange event \(Client API reference\)](#)

Debug JavaScript code for model-driven apps

Article • 03/17/2023

Custom logic using JavaScript in model-driven apps are contained within JavaScript web resources. JavaScript web resources provide the libraries that define functions developers register as event handlers.

In a model-driven app viewed within a web browser, you can use developer tools that all modern browsers provide. With these tools you can locate the JavaScript libraries loaded in the model-driven application, set break points and debug your code using common methods.

Model-driven apps viewed using mobile apps on Android, or the Windows desktop app require some additional steps. See:

- [Debug JavaScript in mobile apps on Android](#)
- [Debug JavaScript in the Windows desktop app](#)

Note

Because of the way that the libraries are added to the page, you may not easily find the library representing the JavaScript Web resource. These libraries may not be listed in the file list or in the hierarchy of source files.

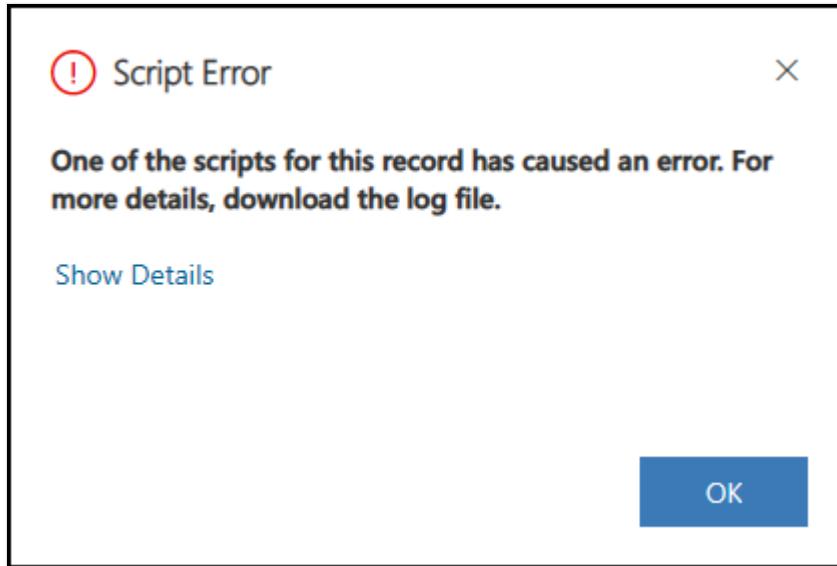
If you know the name of the JavaScript web resource you want to debug, for Microsoft Edge or Google Chrome sources you can use the `Ctrl+P` [Open file](#) command to locate the file by name and start debugging. If you have an event handler that is causing an error, but you don't know the name of the file, see [Identify JavaScript web resource causing error](#).

More information:

- [mdn web docs: What are browser developer tools?](#)
 - [Microsoft Edge Sources](#)
 - [Google Chrome Sources](#)
 - [Mozilla Firefox JavaScript Debugger](#)
 - [Apple Safari Debugger tab](#)

Identify JavaScript web resource causing error

When an event handler causes a script error in a model-driven app, the following dialog appears:



If you click the **Show Details** link, you can find the details that include: event name, function name, web resource name, solution name, and publisher name.



In this case, the name of the function was incorrect, `openalertDialog` should be `openAlertDialog`

ⓘ Note

You can get the same details on errors using Monitor. More information: [Custom script errors](#).

Debug JavaScript in mobile apps on Android

While using JavaScript web resources for mobile scenarios, you can use your Android device to debug your mobile-specific code and ensure it works as expected.

Note

It is not currently possible to debug devices using iOS.

To debug JavaScript in mobile apps, you must complete the three steps below:

1. Configure your device

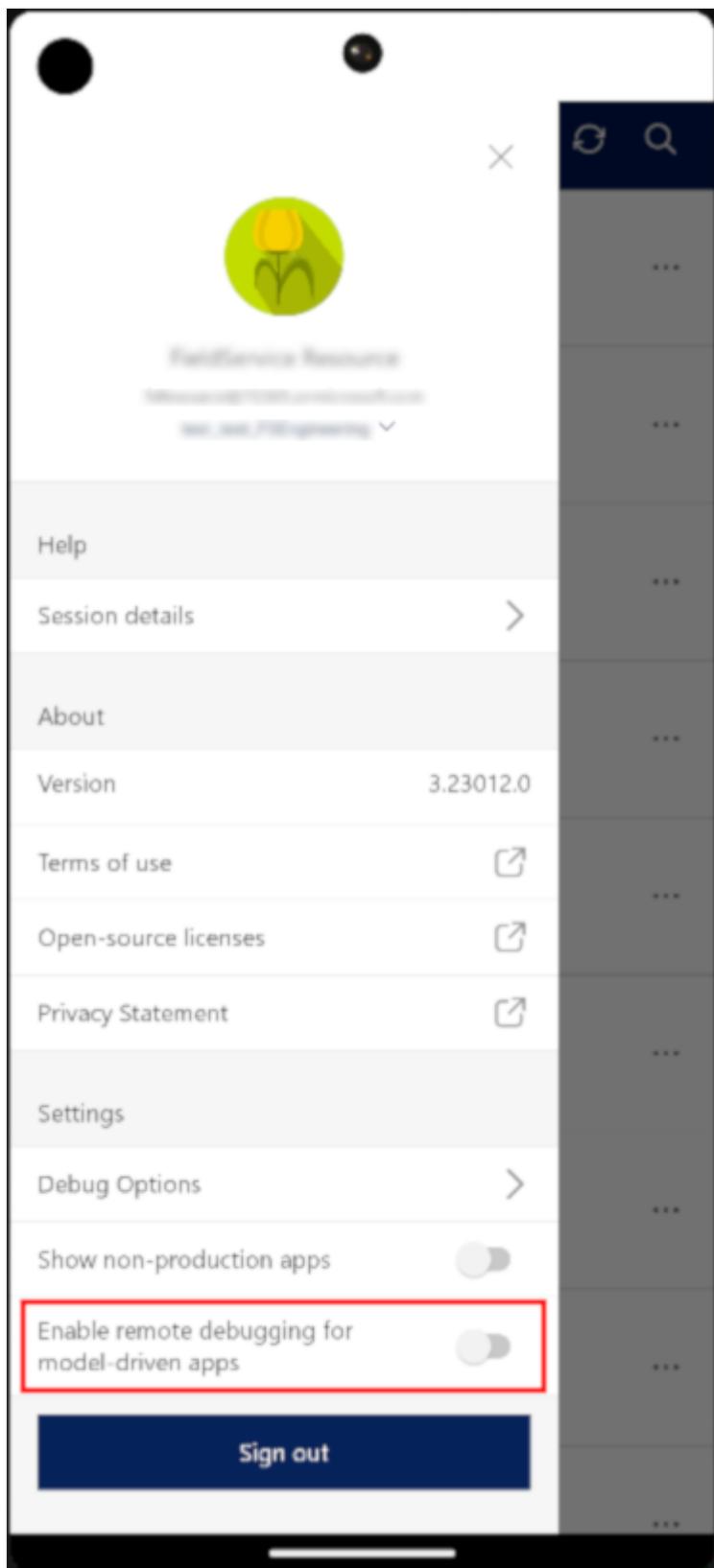
- Refer to the Android documentation to enable Developer options and USB debugging on your device. More information: [Android Developers: Configure on-device developer options ↗](#)
- In the Microsoft Edge or Chrome browser, discover your Android device. More information: [Chrome Developers: Remote debug Android devices ↗](#)
 - On Microsoft Edge: `edge://inspect/#devices`
 - On Chrome: `chrome://inspect/#devices`

Note

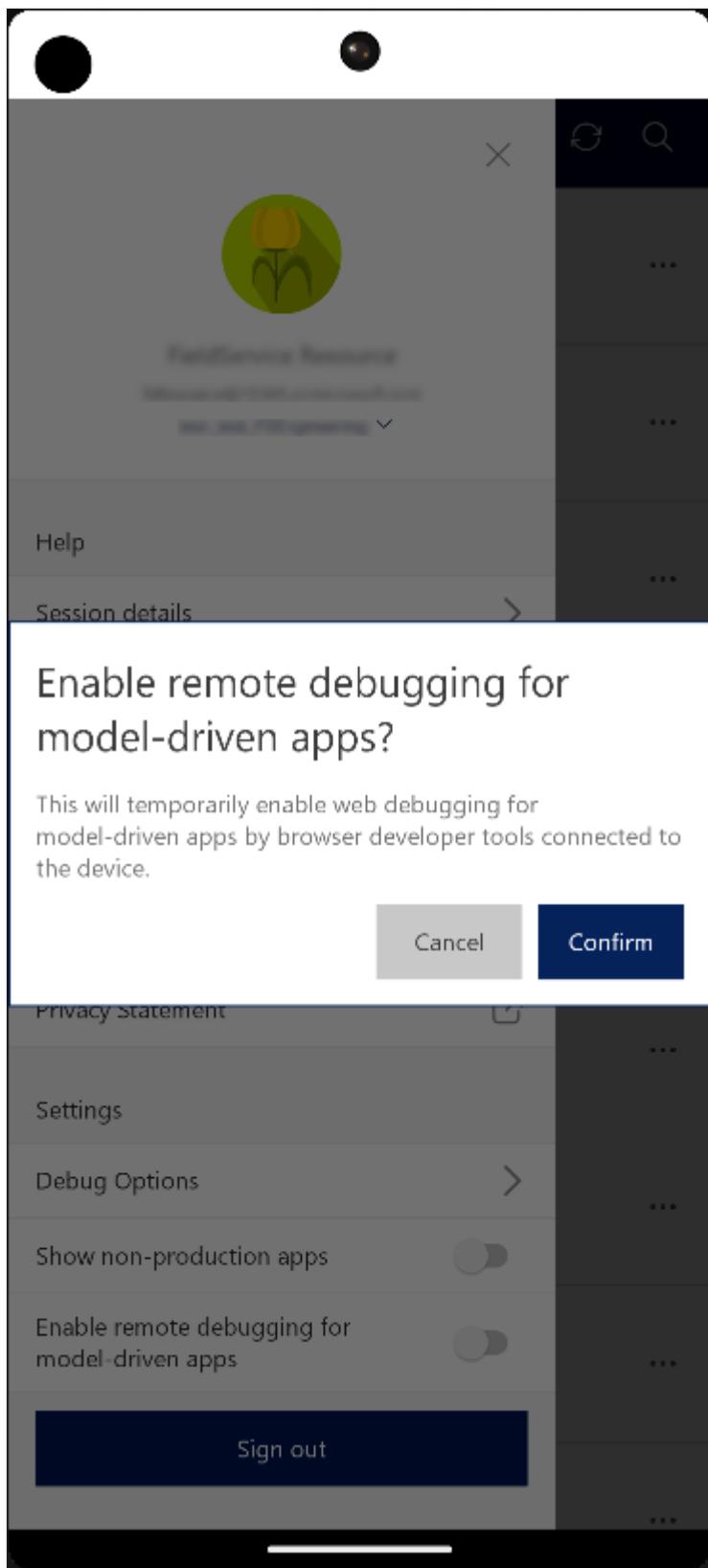
Make sure that **Discover USB devices** is enabled.

2. Configure the mobile application

1. In the mobile app, go to the list of Power Apps and select on the menu button.
2. Make sure that the toggle **Enable remote debugging for model-driven apps** is on.



3. When enabling this option, there's a confirmation dialog. Select **Confirm**.



3. Debug from your development machine

1. Plug your computer to your Android device.
2. Open any model-driven app from Power Apps or the Field Service Mobile application

3. In the `edge://inspect/#devices` page in your browser, find your organization URL in the Remote Target section.

The screenshot shows the Microsoft Edge DevTools interface under the 'Devices' tab. On the left sidebar, there are sections for 'Devices', 'Pages', 'Extensions', 'Apps', 'Shared workers', 'Service workers', and 'Other'. Under 'Devices', there are checkboxes for 'Discover USB devices' and 'Discover network targets', and buttons for 'Port forwarding...' and 'Configure...'. Below these, it says 'Connect to a remote Windows device' and shows a text input field with 'http://machine-name:' and a 'Connect to Device' button. A note below states: 'Windows Device Portal must be enabled in order to connect. [Learn more](#) about enabling remote diagnostics.' There is also a link 'Open dedicated DevTools for Node'. The main area displays a 'Remote Target' entry for '#LOCALHOST'. It lists a device entry: 'sdk_gphone64_x86_64 #EMULATOR-5554'. Underneath, it shows a 'WebView in com.microsoft.crm.crmphone.fieldServices' entry with a status of '(idle) trace'. Below this, a note says 'Bookable Resource Bookings Bookings - https://[REDACTED].crm.dynamics.com/nga/main.htm?liveld=1 at (0,136) size 1080 x 2201'. At the bottom of the list, there are two buttons: 'inspect' (which is highlighted with a red box) and 'pause'.

4. Click on **inspect**.

More information: [Microsoft Edge: Remotely debug Android devices](#)

Debug JavaScript in the Windows desktop app

To debug JavaScript on Windows, you must first install the prerequisite applications.

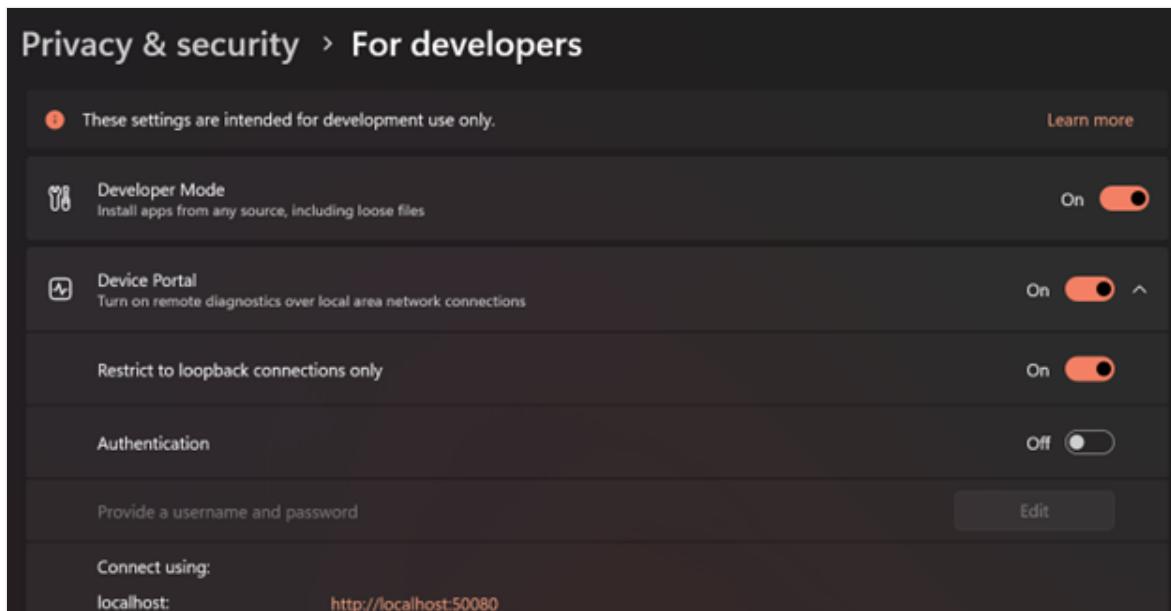
- You must have a WebView2 runtime installed on your machine with a minimum version of 111. [Download the WebView2 Runtime](#)
- You must install the Remote Tools for Microsoft Edge from the Microsoft Store: [Remote Tools for Microsoft Edge - Microsoft Store Apps](#)

1. Configure your Windows device

1. Enable Developer Mode.
 - a. Open **Windows Settings > Privacy & security > For developers**.
 - b. Enable **Developer Mode**.
2. Enable Device Portal.
 - a. Open **Windows Settings > Privacy & security > For developers**.
 - b. Enable **Device Portal**.
 - c. Click **Yes** to install Windows Developer Mode package when prompted.
 - d. Once Device Portal is enabled, **note the URL you will use to connect using localhost**. On most devices, it's `https://localhost:50080`

- e. For debugging Power Apps / Field Service Mobile locally, you can continue with **Restrict to loopback connections only** enabled and **Authentication** disabled (otherwise you need to set a user password)

Here's a screenshot summary of the recommended developer settings:



2. Configure the Windows application

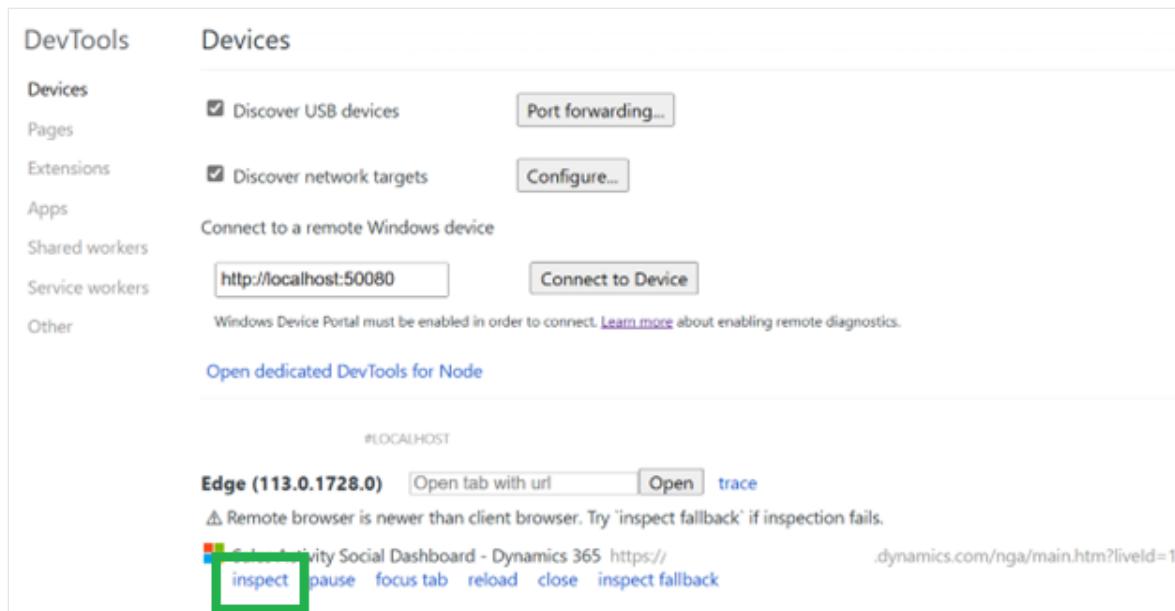
Configure the Power Apps or Field Service Windows desktop app for remote debugging.

Use the **Run command** (shortcut is `Windows + R`) and use the following deep link to launch the Windows app with special arguments.

- Power Apps: `ms-apps://?addWebView2AdditionalBrowserArgument=--enable-features=msEdgeDevToolsWdpRemoteDebugging`
- Field Service Mobile: `ms-apps-fs://?addWebView2AdditionalBrowserArgument=--enable-features=msEdgeDevToolsWdpRemoteDebugging`

3. Debug from Windows

1. Launch Power Apps or Field Service for Windows.
2. Open the Microsoft Edge browser and navigate to `edge://inspect`.
3. Use the **Connect to a remote Windows device** section and connect to `http://localhost:50080`. It takes a few seconds to connect, but you should see the organization URL.



4. Click **inspect** and the DevTools will open.

See also

[JavaScript web resources](#)

[Debug a model-driven app with Monitor](#)

[Troubleshoot issues in the Power Apps mobile app](#)

Troubleshoot form issues in model-driven apps

Article • 11/30/2022

Troubleshooting issues in Unified Interface is essential when you're working with forms that occur when loading a form, running a script, working with events, or saving data.

This article helps you fix some of the common issues you might encounter while working with model-driven app forms.

Important

- The tools described in this article are designed for troubleshooting purposes; they aren't meant to be used in day-to-day production scenarios, even though you can use them for troubleshooting issues in production environments.
- These troubleshooting tools only affect the current user session unless otherwise noted (for example, when a browser tab accesses the model-driven app). They don't change system customizations or affect any other users or sessions. After the current session is closed, the effect is no longer applied.
- Most of the tools are available in all the production environments. Some of them mentioned in the article might not have been deployed to your organization yet; new tools are added periodically.
- Tools listed in this article are written in a scenario-driven way. You can use them independently to troubleshoot different types of issues.
- [Use URL parameters to disable various form components](#) and [View registered form event handlers and libraries in Monitor](#) are critical and fundamental tools you'll frequently use to troubleshoot many scenarios.
- For more information on how to use Monitor, see [Use Monitor to troubleshoot model-driven app form behavior](#)

Use URL parameters to disable various form components

When you're troubleshooting issues with forms, you need to use the URL parameters to disable components as you work to isolate the specific component that caused the

issue. We recommend that you use the flags one at a time to narrow down the cause of the issue. You can use the following URL parameters:

- **DisableFormCommandbar**

Disables the command bar on the form. It only disables the command bar on form pages and not supports list (grid), dashboard, etc.

HTTP

```
https://myorg.crm.dynamics.crm/main.aspx?appid=00000000-0000-0000-0000-000000000000&pagetype=entityrecord&id=00000000-0000-0000-0000-000000000000***&flags=DisableFormCommandbar=true
```

- **DisableFormHandlers**

Disables all the form handlers. If you use the **DisableFormHandlers=true** flag, it disables the following event handlers: [OnLoad](#), [OnSave](#), business rule, [OnChange](#), and [TabStateChange](#).

To learn more about obtaining event or library indices for granular controls, see [View registered form event handlers and libraries in monitor](#).

HTTP

```
https://myorg.crm.dynamics.crm/main.aspx?appid=00000000-0000-0000-0000-000000000000&pagetype=entityrecord&id=00000000-0000-0000-0000-000000000000***&flags=DisableFormHandlers=true
```

- **&flags=DisableFormHandlers=eventName**

Disables the form handler by specifying the event name, for example, ****DisableFormHandlers=onload**.

HTTP

```
https://myorg.crm.dynamics.crm/main.aspx?appid=00000000-0000-0000-0000-000000000000&pagetype=entityrecord&id=00000000-0000-0000-0000-000000000000***&flags=DisableFormHandlers=true
```

- **&flags=DisableFormHandlers=eventName_index**

Disables the event handler at the specified index for any supported event name. For example, **DisableFormHandlers=true_0** disables all the event handlers at

index 0. `DisableFormHandlers=onload_2` disables the **OnLoad** event handler at index 2.

- **&flags=DisableFormHandlers=eventName_startIndex_endIndex**

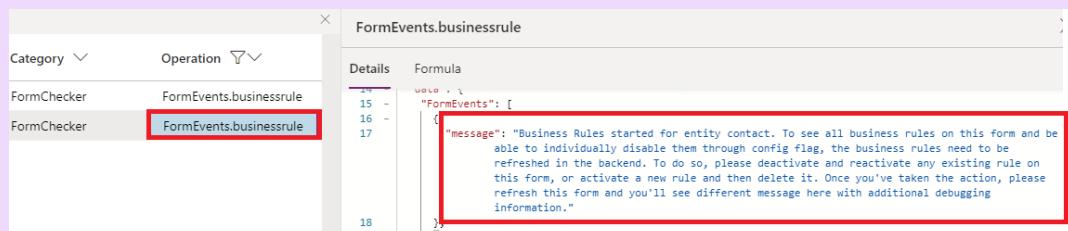
Disables all the event handlers within the given range by specifying `startIndex` and `endIndex` values (both are included). For example,

`DisableFormHandlers=true_0_2` disables all the event handlers of index 0, 1, and 2. `DisableFormHandlers=onload_2_5` disables the **OnLoad** event handler of index 2, 3, 4, and 5. If you have more event handlers, you can use this approach to narrow down problematic handlers quickly.

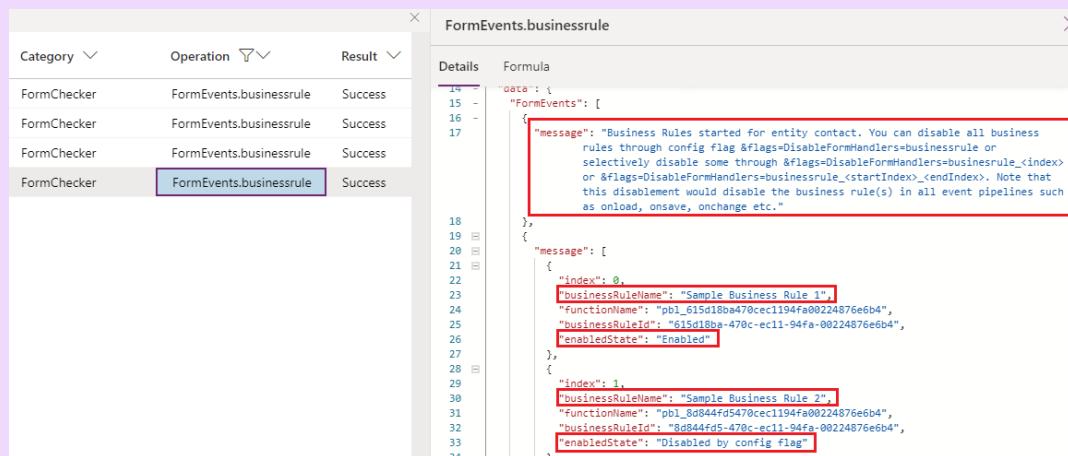
① Note

Business rules are authored in the business rule designer, compiled into the client-side script, and registered in multiple form events, such as `OnLoad`, `OnSave`, and `OnChange`. The way to disable business rules are very similar to other form events. However, there're a few key differences:

- When you use `DisableFormHandlers=true`, `businessrule`, `businessrule_*index*`, or `businessrule_*startIndex_endIndex*`, you're disabling the business rule(s) in all the form events they're registered to.
- For example, the image below shows instructions on refreshing business rule(s) in the backend. You only need to do it once in your organization, and you can revert your changes after troubleshooting.



- After you perform the above action and refresh the form, you'll see different message with additional information, as shown in below:



- **DisableFormLibraries**

Disables form libraries and prevents the libraries from being loaded. To learn more about obtaining event or library indices for granular controls, see [View registered form event handlers and libraries in monitor](#). The usage is similar to

`DisableFormHandlers`, except it does not take an event name as the value.

- `&flags=DisableFormLibraries=true`: Disable all the form libraries.
- `&flags=DisableFormLibraries=index`: Disable form libraries at the specified index.
- `&flags=DisableFormLibraries=startIndex_endIndex`: Disable form libraries in the range of startIndex and endIndex (both included).

- **DisableWebResourceControls**

Disables all the web resource controls on the form.

HTTP

```
https://myorg.crm.dynamics.crm/main.aspx?appid=00000000-0000-0000-0000-000000000000&pagetype=entityrecord&id=00000000-0000-0000-0000-000000000000***&flags=DisableWebResourceControls=true
```

Name * **1**

Owner *  **First name Last name**

WebResource_formcheckerdemo_bad: The control has been disabled with a URL parameter

- **DisableFormControl**

Disables a form control. Specify the control name to disable the control. If you see that the issue goes away with `&flags=DisableWebResourceControls=true`, and there is more than one web resource control on the form, you can use this flag to further identify the control that's causing the issue.

HTTP

```
https://myorg.crm.dynamics.crm/main.aspx?appid=00000000-0000-0000-0000-000000000000&pagetype=entityrecord&id=00000000-0000-0000-0000-000000000000***&flags=DisableFormControl=controlname
```

- **DisableBusinessProcessFlow**

Disables all the business process flows on the form.

HTTP

```
https://myorg.crm.dynamics.crm/main.aspx?appid=00000000-0000-0000-0000-  
000000000000&pagetype=entityrecord&id=00000000-0000-0000-0000-  
000000000000**&flags=DisableBusinessProcessFlow=true
```

- **navbar** This isn't a **flag** parameter; instead, use **navbar=off** in the URL.

You can also add multiple URL parameters separated with a comma (,).

Http

```
https://myorg.crm.dynamics.crm/main.aspx?appid=00000000-0000-0000-0000-  
000000000000&pagetype=entityrecord&id=00000000-0000-0000-0000-  
000000000000**&flags=DisableFormHandlers=true,DisableWebResourceControls=true,  
DisableFormCommandbar=true,DisableBusinessProcessFlow=true&navbar=off
```

① Note

The difference between **DisableFormHandlers** and **DisableFormLibraries** are:

- The **DisableFormHandlers** flag disables form handlers regardless of the containing form libraries. In contrast, the **DisableFormLibraries** flag disables the form libraries (web resources) regardless of the functions (event handlers) included in the libraries. Simply put, **DisableFormLibraries** makes sure the specified JavaScript web resource files are not loaded.
- The **DisableFormHandlers** flag doesn't prevent the containing form library from being loaded. Thus it doesn't stop the JavaScript code present in the library but not registered as an event handler from being executed. For example, if a form library `new_myscript.js` is written in the following way (not recommended practice):
- You should start with **DisableFormHandlers** to see if the issue goes away, and if not, you can try **DisableFormLibraries**. Disabling any script always involves some risks of potentially breaking your form scenarios. However, the latter tend to have more side effects because of the disablement of the entire JavaScript files.

```
1 // new_myscript.js  
2 alert("First alert that's not registered in any form events but will still be triggered when the form loads!");  
3  
4 function myOnloadHandler () {  
5 |   alert("Second alert that's registered in the form onload event!");  
6 }
```

- Assuming the `myOnloadHandler` is registered as an `OnLoad` event handler, the `DisableFormHandlers=true` flag only prevents the second alert, whereas the `DisableFormLibraries=true` flag prevents both alerts.

View registered form event handlers and libraries in monitor

To view registered form event handles and libraries, you can view the `FormEvents` operation in [Monitor](#).

```

{
  "nodeId": null,
  "formulaData": {...},
  "data": {
    "FormEvents": {
      "AllEvents": [
        {
          "eventIndex": 0,
          "function": "onload_0",
          "library": "new_FormLibrary_0.js",
          "enabledState": "Enabled"
        },
        {
          "eventIndex": 1,
          "function": "onload_1",
          "library": "new_FormLibrary_1.js",
          "enabledState": "Enabled"
        }
      ],
      "AllLibraries": [
        {
          "libraryIndex": 0,
          "library": "new_FormLibrary_0.js",
          "disabledByConfigFlag": false
        },
        {
          "libraryIndex": 1,
          "library": "new_FormLibrary_1.js",
          "disabledByConfigFlag": false
        }
      ]
    }
  }
}

```

You'll need the `eventIndex` and `libraryIndex` parameter values when using the `DisableFormHandlers` or `DisableFormLibraries` URL flags. After an event or library is disabled, you'll see the event enabled state in both `FormEvents` operation (an overall view of all registered event handlers of all events), and `FormEvents.eventName` operation (details logged when a specific event happens).

FormChecker	QuickCreateMenu
FormChecker	FormEvents
FormChecker	UnsupportedClient...
FormChecker	XrmNavigation
FormChecker	FormEvents.business...
FormChecker	XrmNavigation
FormChecker	FormEvents.onload
FormChecker	ControlDefaultValue
FormChecker	FormControls
FormChecker	RelatedMenu
FormChecker	QuickCreateMenu
FormChecker	FormEvents
FormChecker	UnsupportedClient...
FormChecker	XrmNavigation
FormChecker	FormEvents.business...
FormChecker	XrmNavigation
FormChecker	FormEvents.onload
FormChecker	RelatedMenu
FormChecker	FormControls
FormChecker	ControlDefaultValue

```

18 - {
19   "eventIndex": 0,
20   "function": "onload_0",
21   "library": "new_FormLibrary_0.js",
22   "enabledState": "Enabled"
23 },
24 {
25   "eventIndex": 1,
26   "function": "onload_1",
27   "library": "new_FormLibrary_1.js",
28   "enabledState": "Disabled by config flag"
29 }

15 - "FormEvents": [
16 - {
17   "message": "onload handler started for entity new_formcheckerdemo.",
18   "event": {
19     "identifier": "onload_0c57b933-8736-4219-9c2c-a8d6d123654c",
20     "eventName": "onload",
21     "attributeName": null,
22     "functionName": "onload_0",
23     "webResourceName": "new_FormLibrary_0.js",
24     "solutionName": "FormCheckerDemo",
25     "solutionVersion": "1.0.0.0",
26     "publisherName": "formcheckerdemopublisher",
27     "type": "onload",
28     "source": "This event handler is defined via the form XML."
29   }
30 },
31 {
32   "message": "onload handler is disabled. Reason: Disabled by config flag",
33   "event": {
34     "identifier": "onload_26d5888f-b277-4299-908b-bf7d4880301a",
35     "eventName": "onload",
36     "attributeName": null,
37     "functionName": "onload_1",
38     "webResourceName": "new_FormLibrary_1.js",
39     "solutionName": "FormCheckerDemo",
40     "solutionVersion": "1.0.0.0",
41     "publisherName": "formcheckerdemopublisher",
42     "tvoe": "onload"
43   }
44 }
]

```

Unexpected behaviors when loading a form

Issue

Some common issues that can cause unexpected behavior when a model-driven app form is loaded are:

- Columns or controls don't have the values you expect.
- Controls aren't disabled or aren't enabled.
- Controls aren't shown or aren't hidden.

How to troubleshoot

There are multiple reasons why unexpected behaviors occur when a form opens. One of the most common is the [OnLoad](#) scripts that run synchronously or asynchronously to change the column or control behavior. To determine whether your script is causing the issue, you can disable the form handlers by appending `&flags=DisableFormHandlers=true` at the end of your app URL.

If the unexpected behavior stops occurring after you disabled the form handler, it is a strong indication that the specific form handler is causing this behavior. If you have identified the problematic script that's causing this behavior, follow up with the script owner to further troubleshoot this issue.

Saving in progress error message

Issue

Sometimes when you save a form, you see a **Saving in Progress** error message.

This error occurs when the form `OnSave` event is triggered before the previous `OnSave` event has been completed. This behavior isn't supported, and the error appears by design because calling the `OnSave` event before the previous `OnSave` event is complete will cause recursive save loops with unintended consequences.

A typical cause for this error is the script that calls the `save()` method in the `OnSave` event handler. Another possible cause might be concurrent `save()` calls in the `setTimeout()` method, which might cause the error to intermittently show up, depending on whether the prior `save()` call was completed before another `save()` call was made.

How to troubleshoot

In [Monitor](#) the `FormEvents.onsave` operation provides all the details that are causing the error (the call stack has been modified for demonstration purpose). The call stack tells what exact web resource, function, line, and row number causing this error. The form checker cannot detect the error if the issue can't be reproduced.

FormEvents.onsave		
Category	Operation	Result
FormChecker	XrmNavigation	Success
FormChecker	ControlStateChange...	Success
FormChecker	FormEvents.onload	Success
FormChecker	RelatedMenu	Success
FormChecker	FormControls	Success
FormChecker	RelatedMenu	Success
FormChecker	FormControls	Success
FormChecker	FormEvents.onsave	Success

Details	Formula	Request	Response
19			
20			
21	}, { "message": "Save is called again while previous save operation is in progress for entity new_formcheckerdemo. Make sure the previous save event handler code does not call client API to save the form.", "callstack": "		
22			

[onsave_saveInProgress \(http://%7b637346779920023008%7d/webresources/new_formcheckerdemo:64:19\)\n at](#)

Follow up with the script owner to further troubleshoot the issue.

Intermittent form errors

Issue

The most common cause of intermittent or random form errors is using unsupported [client API](#) methods. These errors have the following characteristics:

- They occur only for specific records, users, regions, or browsers, or only during periods when the network or service load is high.
- They rarely occur in support instances.
- They might occur once on a computer, and the same error might occur again after you clear the browser cache.
- `formContext.getControl` or `formContext.getControl(arg).getAttribute()` randomly returns null for a valid control or column.

There are many ways to write unsupported client API methods, and they all share a common pattern: they cause a race condition in the form load pipeline. Because they introduce a race condition, the issue only occurs when the custom script executes before the form is fully ready to be accessed via the client API. This can depend on many factors:

- In the JavaScript web resource, code is put into a global scope that's executed immediately when the web resource file is loaded, without waiting for the form to be accessible. Make sure the code is executed inside a valid form handler, such as an [OnLoad](#) handler.

```
1 // Bad custom script example
2
3 // this is a bad place to invoke client API, this API will fail intermittently
4 // Move it into a form handler, such as onload
5 Xrm.Page.getAttribute("myAttribute").setDisabled(true);
6
7 function myOnloadHandler(executionContext) {
8     // this is a recommended place to put your client API code in
9 }
```

- In the Power Apps component framework component script file, client API methods are accessed inside the `init` or `updateView` function. The `init()` and `updateView()` functions are executed immediately when the component is loaded, without waiting for the form to be readily accessible. You can't use unsupported Client API methods in Power Apps component framework components.

Client API is accessed inside a `window.setTimeout()` function in the web resource file. The page state is unpredictable when the `setTimeout()` method executes the wrapped function. Due to the nature of the timer function, when the execution occurs, the page

might be in a transitional state (during page load or save) that's not readily accessible by the Client API.

How to troubleshoot

Using [Monitor](#), you can access information that helps you determine when the unsupported client access occurred and when the access occurred at the wrong time due to a race condition. However, Form Checker will not report such unsupported client access when the unsupported code happens to execute at the right time that does not cause an issue.

Category			Operation				Result	
			Details	Formula	Request	Response		
FormChecker		UnsupportedClientApi	Error					
FormChecker		UnsupportedClientApi	Error					
FormChecker		UnsupportedClientApi	Error					

```
61  {
62    "name": "pageLoadingState",
63    "value": 2
64  },
65  {
66    "name": "callStack",
67    "value": " at onload_controlstate (
      /webresources/new_formcheckerdemo:14:14)\n at
```

Note

The call stack has been modified for illustration purposes. The call stack shows details like web resource, function, and the line causing the error.

Follow up with the script owner to further troubleshoot the issue.

The form or record isn't saved when you try to save the form

Issue

A common cause is an [OnSave](#) event handler that calls the `executionContext.getEventArgs().preventDefault()` method to cancel the save operation.

How to troubleshoot

In [Monitor](#), the `FormEvents.onsave` operation provides all the details of why the save event is canceled details than that are available from the form UI itself.

Category	Operation	FormEvents.onsave	
FormChecker	FormEvents.onsave		
Details	Formula	Request	Response
21			"message": "onsave handler completed for entity new_formcheckerdemo (default prevented, save operation will be canceled).",
22			"event": {
23			"eventName": "onsave",
24			"webResource": "new_formcheckerdemo",
25			"functionName": "onsave_preventDefault",
26			"solutionName": "FormCheckerDemo",
27			"publisherName": "formcheckerdemopublisher"
28			}
29			,
30			"message": "Save operation canceled for entity new_formcheckerdemo."
31			}
32			

Follow up with the script owner to further troubleshoot the issue.

Form freezes, loads slowly, or throws unexplained errors

Issue

There are many possible reasons for a form to freeze, load slowly, or throw a "Web resource method does not exist" script error or an error that isn't a common script error. Some of the possible reasons include:

- Bad `OnLoad` scripts.
- Web resource controls.
- Ribbon button scripts and rules.
- Synchronous network requests.
- Custom plug-ins.
- Business process flow errors.

How to troubleshoot

Determine if the issue reproduces without involving forms. If it does, then there is a broader issue that should be investigated out of the form's context. Actual ownership of the problem depends on the particular details case by case.

- If you believe this issue only occurs on forms, see [Use URL parameters to disable various form components](#) to narrow down the component that's causing the issue.
- If you identified that the issue is caused by certain form libraries/script files, follow up with the owner who made these customizations further to find out the root

cause of the issue.

- If you have identified that web resource controls cause the issue with the **DisableWebResourceControls** flag, then you can use the **DisableFormControl** flag to disable each one-by-one until the problem is longer reproduced. The last disabled control that doesn't reproduce the issue is the one that is causing the issue. Follow up with the owner of the control to further troubleshoot the issue.
- If you have identified that the issue is caused by the command bar/ribbon with the **DisableFormCommandbar** flag, this means this is not an issue with the form but an issue with the command bar. Use [Command Checker](#) to troubleshoot individual commands and identify which one is causing the issue.

A business rule or custom script isn't working

Issue

This issue occurs if a business rule or custom script that used to work in the legacy web client stopped working in Unified Interface. One of the main reasons for this error to occur is when a business rule or script in Unified Interface references a control that isn't available in Unified Interface.

How to troubleshoot

One of the reasons that the business rule or script is not working in Unified Interface is that the controls that are part of them do not exist in Unified Interface. Composite controls exist in the web client, but in Unified Interface composite control is broken down into parts and is stored differently. For example, if the column `fullname` is part of the business rule or custom script, columns `firstname`, `middlename`, or `lastname` should be used instead.

Once you launch form checker, you'll be able to see more details in the `CompositeControl` operation including the composite control that is causing the problem, the columns that can be used in the business rule or custom script instead and a full call stack (the call stack has been modified for demonstration purpose).

Category	Operation ▾	Result
FormChecker	CompositeControl	Error
<div style="border: 1px solid #ccc; padding: 5px;"> <p>Details Formula Request Response</p> <pre> 1 [2 "status": null, 3 "duration": null, 4 "datasource": "Forms.FormChecker.CompositeControl", 5 "responseSize": null, 6 "controlName": null, 7 "propertyName": null, 8 "nodeId": null, 9 "formulaData": { ... 13 }, 14 "data": { 15 "CompositeControl": { 16 "A composite control accessed through a business rule or a custom script": "fullname", 17 "This action is unsupported. Contact the business rule or custom script owner to make the change to use these controls instead": [18 "firstname", 19 "middlename", 20 "lastname" 21], 22 "Source": "Business rule", 23 "callstack": ... </pre> </div>		

Follow up with the corresponding owner of the business rule or custom script to change the control suggested by the form checker.

Related menu/Related tab

Issue

There are many reasons why a related menu item doesn't appear on the **Related** tab or has an incorrect label.

How to troubleshoot

In the following example, a related table `role` (security role) doesn't appear in the `team` form because the `role` table isn't available in Unified Interface.

Category	Operation ▾	Details	Formula	Request	Response
FormChecker	RelatedMenu				
		<pre> 1 [2 "time": 1599864907684, 3 "category": "FormChecker", 4 "name": "RelatedMenu", 5 "data": { 6 "RelatedMenu": [7 { 8 "Id": "navRoles", 9 "Label": null, 10 "Area": "Related - Common", 11 "Primary Entity": "team", 12 "Related Entity": "role", 13 "Relationship": "teamroles_association", 14 "Displayed": true, 15 "Displayed State Reason": " [RelatedMenu_NotDisplayed_RelatedEntityMetadataNotAvailableInUnifiedClient]: The related record is not available in Unified Client. The entity or table may not be included in the App Module, or not supported in Unified Client. Please contact the entity owner to solve this problem." </pre>			

In [Monitor](#), the `RelatedMenu` operation provides all the details that are causing the issue.

There are also a few sources where a record can be included as an option for the **Related** menu tab. The following example contains details that indicate that the label `Activities` in the **Related** menu on an account form comes from the plural display name of the related table.

Category	Operation ▾	Details	Formula	Request	Response
FormChecker	RelatedMenu				
FormChecker	RelatedMenu	<pre> 6 □ 7 □ 8 □ 9 □ 10 □ 11 □ 12 □ 13 □ 14 □ 15 □ 16 □ 17 □ </pre> <p>"RelatedMenu": [</p> <p> {</p> <p> "Id": "navActivities",</p> <p> "Label": "Activities",</p> <p> "Area": "Related - Common",</p> <p> "Primary Entity": "account",</p> <p> "Related Entity": "activitypointer",</p> <p> "Relationship": "Account_ActivityPointers",</p> <p> "Displayed": true,</p> <p> "Displayed State Reason": "[RelatedMenu_Disabled_SetInRelationshipMetadata]: This related menu option is set by the relationship defined for the record or entity. Contact the entity owner to make a change to the relationship to display in the related menu options.",</p> <p> "Label Source": "[RelatedMenu_LabelSource_EntityDisplayCollectionName]: The label comes from the related entity's plural display name. Contact the entity owner to make a change."</p> <p> },</p>			

Follow up with the corresponding owner of the related menu item.

Why a control is disabled/enabled or visible/hidden

Issue

There are many possible reasons why a control might be disabled or hidden when the form is loaded.

How to troubleshoot

You can use [Monitor](#) to view the `FormControls` operation that includes all the details about the initial control state when the form loads.

Category	Operation ▾	FormControls	
FormChecker	FormControls		
Details	Formula	Request	Response
<pre> 22 □ 23 □ 24 □ 25 □ 26 □ 27 □ 28 □ 29 □ 30 □ 31 □ 32 □ 33 □ 34 □ 35 □ 36 □ 37 □ 38 □ </pre> <p> },</p> <p> {</p> <p> "Control Name": "new_disabledinformxml",</p> <p> "Control Label": "Disabled in Form Xml",</p> <p> "Control Label Source": "[Control_LabelSource_ControlLabelInFormDescriptorInUserLanguage]",</p> <p> "Attribute": "new_disabledinformxml",</p> <p> "Disabled": true,</p> <p> "Disabled State Reason": "[Control_Disabled_FormDescriptor]: The control is disabled in the form XML. Contact the form or entity owner to change it."</p> <p> },</p> <p> {</p> <p> "Control Name": "new_disabledbyclientapi",</p> <p> "Control Label": "Disabled by Client Api",</p> <p> "Control Label Source": "[Control_LabelSource_ControlLabelInFormDescriptorInUserLanguage]",</p> <p> "Attribute": "new_disabledbyclientapi",</p> <p> "Disabled": true,</p> <p> "Disabled State Reason": "[Control_Disabled_ClientAPI]: A custom call using the forms client API or a business rule was made to disable this control. This call replaces the form XML configuration and determines the record state."</p> <p> },</p>			

Another place to check is the `ControlStateChange.visible` or `ControlStateChange.disabled` operation that explains why the control disable or visible state is changed at any time on the form. This operation explains the control state before the change, intended state change that may or may not succeed, and the state after the change. Not all control state change attempts are successful. For a control

disabled by form XML, you can enable it through client API in an `OnLoad` event handler. However, if control is disabled for security reasons, it's highly unlikely an attempt to enable it through client API would successfully change the state.

Category	Operation ▾	ControlStateChange.disabled			
		Details	Formula	Request	Response
FormChecker	ControlStateChange.disabled		<pre> 18 }, 19 { 20 "name": "controlLabel", 21 "value": "Disabled by Client Api" 22 }, 23 { 24 "name": "disabled state before change", 25 "value": false 26 }, 27 { 28 "name": "intended disabled state", 29 "value": true 30 }, 31 { 32 "name": "disabled state after change", 33 "value": true 34 }, 35 { 36 "name": "callstack", 37 "value": " </pre>		
					<code>onload_controlstate (%7b637354762440023313%7d/webresources/new_formcheckerdemo:14:52)\n at a1.executeFunction</code>

A control can be disabled by using the following list of rules in order. If a rule is met, then the following rules are ignored. If you want to change whether a control is disabled, you must change the input to the rule used for the result or to a rule earlier in the list.

- If the flags `DisableWebResourceControls=true` or `DisableFormControl=<control name>` are passed and the control is affected by these flags, the control will be disabled.
- If the owning table is read-only in Unified Interface in table definitions, the control is disabled.
- If the table isn't available in offline mode, the control is disabled.
- If the current user doesn't have write permissions on the record, the control is disabled.
- If the column definition has `IsValidforCreate` set to false, the control is disabled.
- If the column definition has `IsValidforUpdate` set to false, the control is disabled.
- If the current user doesn't have `Assign to` privilege, the owner column is disabled.
- If the user doesn't have write permissions on the column defined by field-level security, the control is disabled.
- If the control is disabled or enabled by the Client API script, the control disabled state will honor that setting.
- If the control is disabled in the form designer, the control is disabled.
- If the user doesn't have `Append To` privilege for the lookup control's table, or `Append` privilege on the current record's table, the lookup control is disabled

Finally, if the control passes all the above checks, the record state determines whether the control is disabled. The control is enabled by default on active records and disabled on inactive records.

Note

The difference between `FormControls` and `ControlStateChange` is that the `FormControls` operation reflects the initial control state when the form is loaded, while the `ControlStateChange` operation reflects the state change at any time on the form, whether it's during form load, in `OnChange` or `OnSave` events after the form is loaded.

Important

A control's disabled and hidden state can change multiple times when a form is first loaded. To know the reason why a control is hidden or disabled, make sure to check the `last` operation logged in the monitor. For example, if there are no `ControlStateChange.visible/ControlStateChange.hidden` operations for the control being investigated, the value and reasoning will be in the `FormControls` operation. Otherwise, it will be the value and reason in the `last ControlStateChange.visible/ControlStateChange.hidden` operation. You can order logs by timestamp to search for the last operation.

Why a control has a certain value on form load

Issue

A control may or may not have a specific value on form load as the user expected.

How to troubleshoot

There are many possible reasons why control can have a value when a form loads. The `ControlDefaultValue` operation in [Monitor](#) explains the source of the default values.

Model-driven app session							
Id	Time	Category	Operati...	Result	Re...	Status	
142	17:54:56.764	FormChecker	ControlDefaultValue	Success			

If multiple updates are happening to a control's value, an `Update Sequence` will indicate the final value. For example, here is a control with a default value and then overridden with a value passed with a client API script. There is a call stack provided.

```
--> "donotfax": [
  {
    "Control Name": "donotfax",
    "Control Value": false,
    "Source of the default value": "Autopopulated with the field's default value",
    "Call stack if applicable": "N/A",
    "Update Sequence": 1
  },
  {
    "Control Name": "donotfax",
    "Control Value": "Do Not Allow",
    "Source of the default value": "Value passed in a client API script",
    "Call stack if applicable": "
      at onload : /webresources/
    "
  }
],
{
  "Update Sequence": 2
},
```

There are scenarios where columns are populated based on a relationship column mapping, in which case the event will show that.

```

"ControlDefaultValue": [
    "transactioncurrencyid": [
        {
            "Control Name": "transactioncurrencyid",
            "Control Value": "US Dollar",
            "Source of the default value": "Relationship field mapping",
            "Call stack if applicable": "N/A"
        }
    ],
    "address1_city": [
        {
            "Control Name": "address1_city",
            "Control Value": "Renton",
            "Source of the default value": "Relationship field mapping",
            "Call stack if applicable": "N/A"
        }
    ],
    ...
]

```

Verify where the value is coming from and take action based on the below table:

Source	How to fix
Client API script	Contact the script owner.
Default value	Check the control's configuration.
Relationship column mapping	Check the relationship configuration and update the column mapping.
Value passed by page input data passed via URL	Check the API that opens the specific form with the issue, it is passing the value.

Why a tab or section is visible or hidden

Issue

There are many possible reasons why a tab or section might be hidden or visible.

How to troubleshoot

The `TabStateChange` or `SectionStateChange` operations in [Monitor](#) explain the visible state change, as shown in the following image. If a script causes it, then the call stack would reveal the web resource file, line number, and function name that caused this behavior.

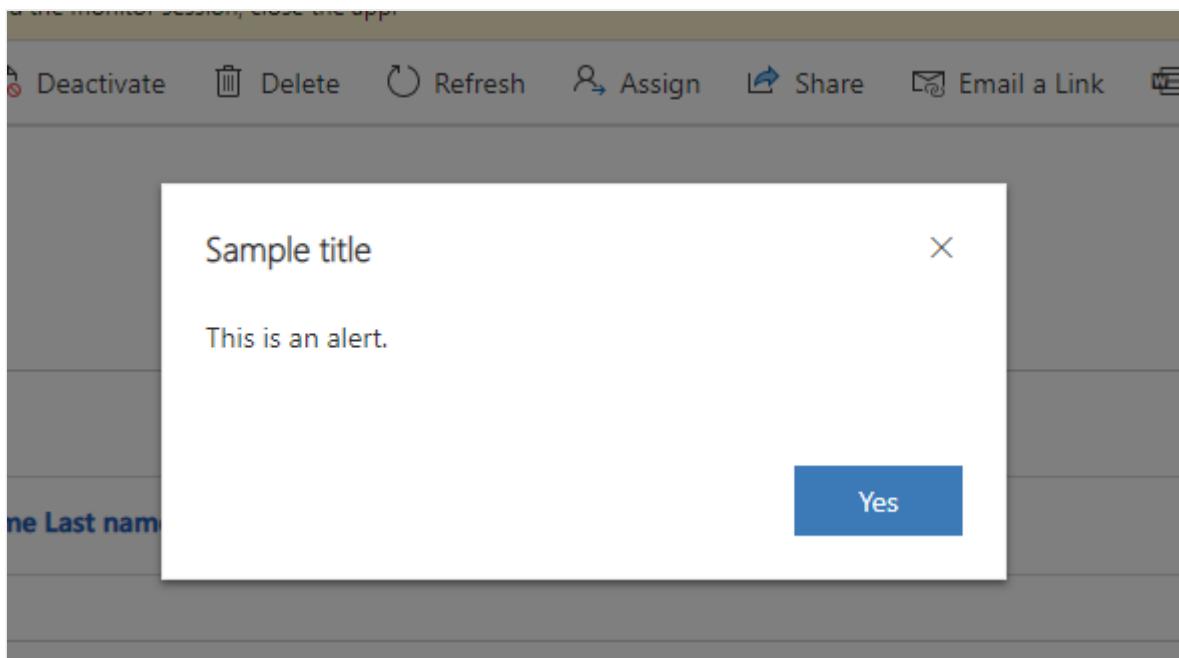
Category	Operation ▾	SectionStateChange.visible			
		Details	Formula	Request	Response
FormChecker	SectionStateChange.visible	15 □ 16 17 18 19 □ 20 21 22 23 □ 24 25 26 27 □ 28 29	<pre>{ "name": "sectionName", "value": "section2" }, { "name": "visible state before change", "value": true }, { "name": "visible state after change", "value": false }, { "name": "callstack", "value": " at m.setVisible (uclient/scripts/8.js?v=1.4.1400-master:41:3082)\n at onField3Change (webresources/new_formcheckerdemo:88:74)\n at " }</pre>		
FormChecker	TabStateChange.visible				

Follow up according to the suggestion in the state reason or the owner of the web resource/business rules to change or fix the behavior.

Unexpected dialogs or navigation

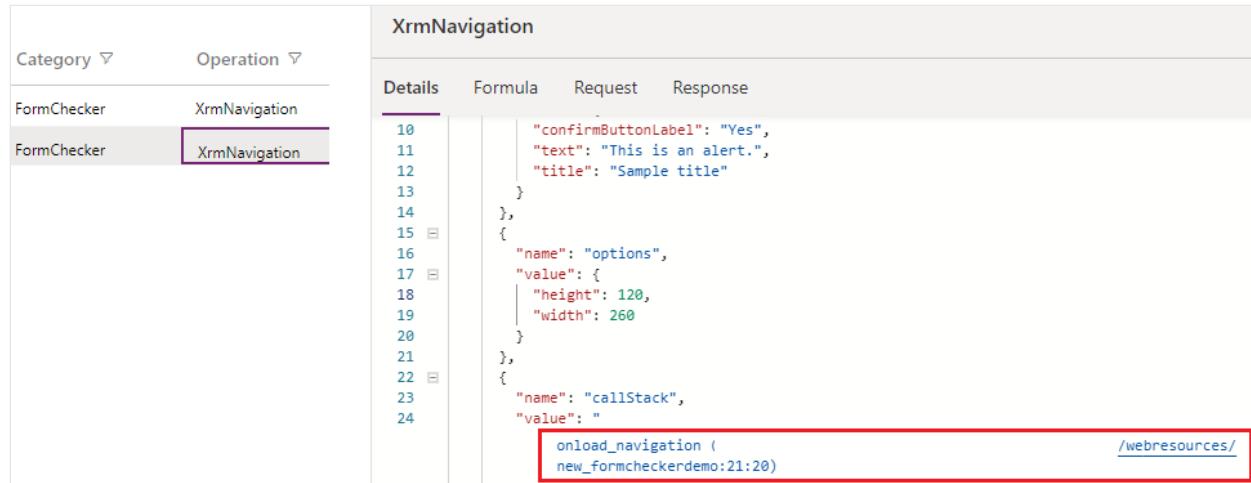
Issue

There are many possible reasons why a dialog appears, or navigation happens unexpectedly. One of the common causes is the [Xrm.Navigation API](#) methods are called to open a record or a form by a custom script. For example, when you open a form, an alert appears, as shown in the following image.



How to troubleshoot

The `XrmNavigation` operation in [Monitor](#) contains call stack that helps you identify the script that's causing unexpected behavior.



The screenshot shows the `XrmNavigation` operation details in the [Monitor](#) tool. The code editor displays the following JavaScript code:

```
10 |     "confirmButtonLabel": "Yes",
11 |     "text": "This is an alert.",
12 |     "title": "Sample title"
13 |
14 |
15 | },
16 | {
17 |     "name": "options",
18 |     "value": {
19 |         "height": 120,
20 |         "width": 260
21 |     }
22 | },
23 | {
24 |     "name": "callStack",
25 |     "value": "
26 |         onload_navigation (
27 |             new_formcheckerdemo:21:20)
28 |     
```

A red box highlights the line `onload_navigation (/webresources/`, indicating it is a web resource URL.

Follow up with the owner of the web resource to change or fix the behavior.

Opening another form instead of a quick create form

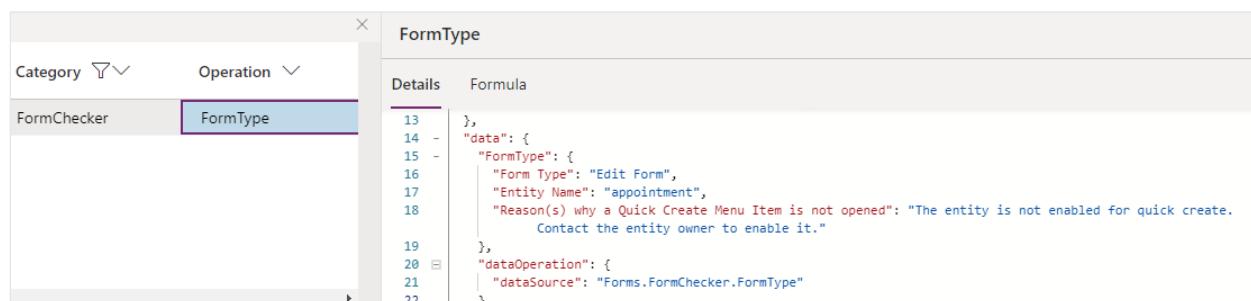
Issue

When opening a quick create form from a lookup or a grid, another form may open (edit or main form) instead of a quick create form. There are few reasons why this can happen:

- The main form dialog force flag is being set.
- Quick create form is not available.

How to troubleshoot

You can use [Monitor](#) to view the `FormType` operation that includes all the reasons why a quick create form is not opened.



The screenshot shows the `FormType` operation details in the [Monitor](#) tool. The code editor displays the following JavaScript code:

```
13 | },
14 | "data": {
15 |     "FormType": {
16 |         "Form Type": "Edit Form",
17 |         "Entity Name": "Appointment",
18 |         "Reason(s) why a Quick Create Menu Item is not opened": "The entity is not enabled for quick create.
19 |             Contact the entity owner to enable it."
20 |     },
21 |     "dataOperation": {
22 |         "dataSource": "Forms.FormChecker.FormType"
23 |     }
24 | }
```

You'll need to follow up with the table owner who has disabled quick create through table definitions (metadata).

Table doesn't appear in the quick create menu flyout

Issue

When opening the global quick create menu flyout, not all tables are available. There are few reasons why the tables are filtered in this list:

- There is no quick create form available for the table.
- Table is not enabled for quick create form.
- Table is not enabled for the new Unified Interface.
- Table is read-only in Unified Interface.
- Table's mobile client visibility cannot be modified.
- Table is not part of the app module.
- User does not have a create privilege on the table.
- The create privilege is not supported for the table.

How to troubleshoot

You can use [Monitor](#) to view the `QuickCreateMenu` operation, which includes all the tables and reasons why they are filtered from the quick create menu flyout.

See the examples below to understand the reasons for filtering. Based on the explanations, contact the responsible party or make changes accordingly.

```
  "post": [
    "The entity is not enabled for quick create. Contact the entity owner to enable it.",
    "The entity is not enabled for Unified Client. Contact the entity owner to enable it.",
    "The entity's mobile client visibility cannot be modified. Contact the entity owner to fix it."
  ],
```

```
  "new_a1": [
    "There is no quick create form available for the entity. Contact the entity owner to make sure there is a quick create form added."
  ]
```

```
  "new_a6": [
    "The entity is not part of the app module. Contact the app owner to add the entity."
  ],
```

Unexpected unsaved changes message

Issue

When working on forms, you get the *unsaved changes* message on the form footer when you navigate from the current form or save the form without any changes.

How to troubleshoot

The *unsaved changes* error appears when you change the form and when the changes are not saved. If you haven't made any changes manually, they could come from a JavaScript, plug-in, or business rule. You can use [Monitor](#) to view the `UnsavedChanges` operation that helps find the source of the changes. You can filter by OperationType `UnsavedChanges`.

The `all attributes modified` section includes a quick summary of the columns causing the unsaved changes error and their values. The `unsaved changes` section shows what happened to the columns in detail. For every column, a list of controls are provided that could be causing a change. The value change is also displayed (`previousValue`, `newValue`), and a call stack.

The screenshot below shows the root cause of the issue. You can see that the change has come from the `OnLoad` script.

```

"data": {
  "UnsavedChanges": [
    {
      "Reported from": "Form footer"
    },
    {
      "All attributes modified": {
        "Account": {
          "fax": 123,
          "name": "New account name"
        }
      }
    },
    {
      "Unsaved changes": {
        "Account": {
          "fax": [
            {
              "message": "Attribute with name 'fax' has been modified. The change could be coming from any of the below controls:",
              "controlNames": [
                "fax"
              ],
              "previousValue": null,
              "newValue": 123,
              "callStack": "at onload /webresources/new_onload:3:33" at Object.runFormOnLoadScripts
            }
          ],
          "name": [
            {
              "message": "Attribute with name 'name' has been modified. The change could be coming from any of the below controls:",
              "controlNames": [
                "name"
              ],
              "previousValue": null,
              "newValue": "Account name",
              "callStack": "The change was manually done by the user."
            },
            {
              "message": "Attribute with name 'name' has been modified. The change could be coming from any of the below controls:",
              "controlNames": [
                "name"
              ],
              "previousValue": "Account name",
              "newValue": "New account name",
              "callStack": "The change was manually done by the user."
            }
          ]
        }
      }
    }
  ]
}

```

! Note

If the user has manually made the changes on the form, a call stack will not be provided.

Verify where the change is coming from and if it's expected behavior or not. If a script causes the change, the original web resource can be traced back in the call stack. In most cases, it will either be a script. Make a call based on the web resource itself.

Business required column validation does not behave as expected

Issue

Business required columns by default block the form save operation if the value is empty. However, in many by-design scenarios, a business-required column may not

block the save operation when the value is empty or block the save when you don't believe it should.

How to troubleshoot

The `RequiredFieldValidation` operation is logged when a save is attempted, regardless of whether save is successful or not. This operation explains why each business-required column blocks or doesn't block the save operation.

Below is an example of this operation. The message explains how to read the detailed reports of each required column. In this example, `fax` column is bound to one control, and the control of the same name is read-only hence will not trigger required column validation.

Category	Operation	Res	Details	Formula
FormChecker	FormControls	S		
FormChecker	QuickCreateMenu	S		
FormChecker	FormEvents	S		
FormChecker	FormEvents.business...	S		
FormChecker	FormEvents.onload	S		
FormChecker	ControlDefaultValue	S		
FormChecker	FormControls	S		
FormChecker	RelatedMenu	S		
FormChecker	QuickCreateMenu	S		
FormChecker	FormEvents	S		
FormChecker	FormEvents.business...	S		
FormChecker	FormEvents.onload	S		
FormChecker	ControlDefaultValue	S		
FormChecker	FormControls	S		
FormChecker	RelatedMenu	S		
FormChecker	RequiredFieldValida...	S		
FormChecker	CompositeControl	E		
FormChecker	CompositeControl	F		

Below is another example that `jobtitle` is a business-required column on the business process flow but not on the form, and the column is not modified. Thus it does not block the save operation even when it's empty.

FormChecker	RelatedMenu	S	42	"attributeName": "jobtitle", "isDirty": false, "isValueEmpty": true, "finalValidationResult": "Required field validation is skipped because the attribute is not placed on the form. This field is also put on the BPF however required field validation is skipped on BPF controls if the field is not dirty."
FormChecker	FormControls	S	43	
FormChecker	RequiredFieldValida...	S	44	
FormChecker	FormEvents.onsave	S	45	

Follow up

Most times, the behavior is by design, and the `RequiredFieldValidation` operation explains why this column behaves in a certain way in form save operation. If the required column validation is skipped on a column because the control is disabled or

hidden, as the first example illustrated, see the form checker suggestions for further analysis.

This may lead to another troubleshooting scenario such as [Why a control is disabled/enabled or visible/hidden](#).

Some columns are not displayed on the merge dialog

Issue

The merge dialog uses the default main form definition for the table and selectively renders most, but not all the columns in the dialog. This Form Checker operation explains why some of the columns are not displayed on the merge dialog, even that they may be displayed on the main form.

How to troubleshoot

The `MergeDialog.load` operation below explains the reason why some columns are not displayed.

In this example, `parentcustomerid` column on the contact form is not supported in the merge dialog. The business card column is not displayed because the containing section is hidden in the main form XML definition. Even though showing the owning section in the main form via client API is possible, the merge dialog honors the form XML configuration as event handlers are not supported on the merge dialog.

MergeDialog.load

Category	Operation
FormChecker	MergeDialog.load

Details Formula

```
34 },  
35 {  
36   "label": "Company Name",  
37   "fieldName": "parentcustomerid",  
38   "owningSectionInMainForm": "name = CONTACT_INFORMATION, label = null",  
39   "displayedState": false,  
40   "displayStateReason": "this control is not displayed because parentcustomerid of contact is not  
41     supported on the merge dialog."  
42 },  
43 {  
44   "label": "Address 1",  
45   "fieldName": "address1_composite",  
46   "owningSectionInMainForm": "name = CONTACT_INFORMATION, label = null",  
47   "displayedState": false,  
48   "displayStateReason": "this control is not displayed because this attribute is not valid for  
49     update. Please contact entity owner for further investigation or explanation of why  
50       address1_composite.IsValidForUpdate is false."  
51 },  
52 {  
53   "label": null,  
54   "fieldName": "mapcontrol",  
55   "owningSectionInMainForm": "name = MapSection, label = null",  
56   "displayedState": false,  
57   "displayStateReason": "this control is not displayed because the metadata for attribute mapcontrol  
58     is not found in the entity metadata."  
59 },  
60 {  
61   "label": "Business Card",  
62   "fieldName": "businesscard",  
63   "owningSectionInMainForm": "name = BusinessCard, label = null",  
64   "displayedState": false,  
65   "displayStateReason": "This control is not displayed because the containing section (name =  
66     BusinessCard, label = null) is hidden in the main form."
```

Provide feedback

Navigating to a custom page using client API

Article • 12/16/2022

This article provides examples of navigating from a model-driven app page to a custom page using [Client API](#).

This article outlines the steps to use client API to open a custom page as a full-page, dialog, or pane. It provides examples of **custom** as a `pageType` value in [navigateTo \(Client API reference\)](#).

ⓘ Important

Custom pages are a new feature with significant product changes and currently have a number of known limitations outlined in [Custom Page Known Issues](#).

Navigating from a model page to a custom page

Finding the logical name

Each of the following client API examples takes the logical name of the custom page as the required parameter. The logical name is the **Name** value of the page in the solution explorer.

Solutions > Customer Management					
Display name ▾	Name	Type ▾	Managed...	Modified	
Customer Landing Page	... contoso_customerlandingpage_30b6d	Canvas app	⋮	-	
Customer Management ⌂	... contoso_CustomerManagement	Model-driven a	⋮	 1 min ago	
Customer Management ⌂	... contoso_CustomerManagement	Site map	⋮	1 d ago	

Open as an inline full page without context

Within a model-driven app client API event handler, call the following code and update the **name** parameter to be the logical name of the page.

JavaScript

```
// Inline Page
var pageInput = {
    pageType: "custom",
    name: "<logical name of the custom page>",
};
var navigationOptions = {
    target: 1
};
Xrm.Navigation.navigateTo(pageInput, navigationOptions)
.then(
    function () {
        // Called when page opens
    }
).catch(
    function (error) {
        // Handle error
    }
);

```

Open as an inline full page with a record context

This example uses the `recordId` parameter within the [NavigateTo](#) function to provide the custom page with the record to use.

JavaScript

```
// Inline Page
var pageInput = {
    pageType: "custom",
    name: "<logical name of the custom page>",
    entityName: "<logical name of the table>",
    recordId: "<record id>",
};
var navigationOptions = {
    target: 1
};
Xrm.Navigation.navigateTo(pageInput, navigationOptions)
.then(
    function () {
        // Called when page opens
    }
).catch(
    function (error) {
        // Handle error
    }
);

```

The `Param` function within the custom page retrieves the value and uses the `Lookup` function to retrieve the record.

PowerApps Formula

```
App.OnStart=Set(RecordItem,
    If(IsBlank(Param("recordId")),
        First(<entity>),
        LookUp(<entity>, <entityIdField> = GUID(Param("recordId"))))
    )
```

Open as a centered dialog

Within a model-driven app client API event handler, call the following code and update the **name** parameter to be the logical name of the custom page. This mode supports the sizing parameters similar to the [Main form dialogs](#).

JavaScript

```
// Centered Dialog
var pageInput = {
    pageType: "custom",
    name: "<logical custom page name>",
};
var navigationOptions = {
    target: 2,
    position: 1,
    width: {value: 50, unit:"%"},
    title: "<dialog title>"
};
Xrm.Navigation.navigateTo(pageInput, navigationOptions)
.then(
    function () {
        // Called when the dialog closes
    }
).catch(
    function (error) {
        // Handle error
    }
);
```

Open as a side dialog

Within a model-driven app client API event handler, call the following code and update the **name** parameter to be the logical name of the custom page.

JavaScript

```
// Side Dialog
var pageInput = {
```

```

    pageType: "custom",
    name: "<logical page name>",
};

var navigationOptions = {
    target: 2,
    position: 2,
    width: {value: 500, unit: "px"},
    title: "<dialog title>"
};

Xrm.Navigation.navigateTo(pageInput, navigationOptions)
.then(
    function () {
        // Called when the dialog closes
    }
).catch(
    function (error) {
        // Handle error
    }
);

```

Open from a grid primary field link as a full page with record ID

This example uses the `recordId` parameter within the `navigateTo` function to provide the custom page with the record to use. The `Param` function within the custom page retrieves the value and uses the `Lookup` function to retrieve the record.

A more complete example of this can be found at [Override the default open behavior of data rows in an entity-bound grid](#).

1. Create a web resource of type **JScript** and update the **name** parameter to be the logical page name. Add the following code to the web resource.

JavaScript

```

function run(selectedItems)
{
    let selectedItem = selectedItems[0];

    if (selectedItem) {
        let pageInput = {
            pageType: "custom",
            name: "<logical page name>",
            entityName: selectedItem.TypeName,
            recordId: selectedItem.Id,
        };
        let navigationOptions = {
            target: 1
        };
    }
}

```

```

        Xrm.Navigation.navigateTo(pageInput, navigationOptions)
            .then(
                function () {
                    // Handle success
                }
            ).catch(
                function (error) {
                    // Handle error
                }
            );
    }
}

```

2. Customize the table ribbon **CommandDefinition** for **OpenRecordItem** to call the function above and include the **CrmParameter** with the value **SelectedControlSelectedItemReferences**.

XML

```

<CommandDefinition Id="Mscrm.OpenRecordItem">
    <Actions>
        <JavaScriptFunction FunctionName="run"
Library="$webresource:cr62c_OpenCustomPage">
            <CrmParameter
Value="SelectedControlSelectedItemReferences" />
        </JavaScriptFunction>
    </Actions>
</CommandDefinition>

```

3. In the custom page, override the App's **OnStart** property to use the **Param** function to get the **recordId** and lookup record.

PowerApps Formula

```

App.OnStart=Set(RecordItem,
If(IsBlank(Param("recordId")),
First(<entity>),
LookUp(<entity>, <entityIdField> = GUID(Param("recordId"))))
)

```

(!) Note

After changing the **OnStart** property, you'll need to run **OnStart** from the App context menu. This function performs only once within a session.

4. Then, the custom page can use the **RecordItem** parameter as a record. Below is an example on how to use it in an [EditForm](#).

```
PowerApps Formula
```

```
EditForm.Datasource=<datasource name>
EditForm.Item=RecordItem
```

Open from a selected record in editable grid as a centered dialog with record ID

Editable grid can be used to trigger [OnRecordSelect](#) event for scenarios where you want to perform an action when a particular record is selected in a view. This example uses the `recordId` parameter within the [navigateTo](#) function to provide the custom page with the record to use. The record ID is retrieved using the `getId` method in [GridEntity](#) object. The `Param` function within the custom page retrieves the value and uses the `lookup` function to retrieve the record.

1. [Enable editable grid](#) control in the table.
2. Create a web resource of type **JScript** and update the **name** parameter to be the logical page name. Add the following code to the web resource.

```
JavaScript
```

```
function RunOnSelected(executionContext) {
    // Retrieves the record selected in the editable grid
    var selectedRecord = executionContext.getFormContext().data.entity;
    var Id = selectedRecord.getId().replace(/\[{}\]/g, "");

    // Centered Dialog
    var pageInput = {
        pageType: "custom",
        name: "<logical page name>",
        recordId: Id,
    };
    var navigationOptions = {
        target: 2,
        position: 1,
        width: { value: 50, unit: "%" },
        title: "<dialog title>"
    };
    Xrm.Navigation.navigateTo(pageInput, navigationOptions)
        .then(
            function () {
                // Called when the dialog closes
            }
        )
}
```

```
    ).catch(
        function (error) {
            // Handle error
        }
    );
}
```

3. In the custom page, override the **App's OnStart** property to use the **Param** function to get the **recordId** and lookup record.

PowerApps Formula

```
App.OnStart=Set(RecordItem,
    If(IsBlank(Param("recordId")),
        First(<entity>),
        LookUp(<entity>, <entityIdField> = GUID(Param("recordId"))))
    )
```

ⓘ Note

After changing the **OnStart** property, you'll need to run **OnStart** from the App context menu. This function performs only once within a session.

4. Then, the custom page can use the **RecordItem** parameter as a record. Below is how to use it in an **EditForm**.

PowerApps Formula

```
EditForm.Datasource=<datasource name>
EditForm.Item=RecordItem
```

Related articles

[Model-driven app custom page overview](#)

[Add a custom page to your model-driven app](#)

[navigateTo \(client API reference\)](#)

Send in-app notifications within model-driven apps

Article • 06/23/2023

Developers of model-driven apps can configure notifications to be displayed to app users as a toast or within the notification center. Your model-driven app automatically polls the system for new notifications and displays them to the user. The notification sender or your system administrator can configure how the notification is shown and how it can be dismissed. Notifications appear in the notification center until the recipient dismisses them or they expire. By default, a notification expires after 14 days but your administrator can override this setting.

Notifications are user-specific. Each notification is intended for a single user, identified as the recipient when the notification is sent. Sending a notification to a team isn't supported. If you need to send notifications to multiple users, you must create notifications for each individual user.

This article outlines the steps for how to send in-app notifications to a specific user. To see how these notifications appear in applications, see [In-app notifications in model-driven apps](#).

Enable the in-app notification feature

To use the in-app notification feature, you need to enable the **In-app notifications** setting. This setting is stored within the model-driven app.

1. Sign in to [Power Apps](#).
2. Open the solution that contains the model-driven app.
3. Select the model-driven app and select the **Edit** split menu to open using the modern app designer.
4. Open **Settings** and switch to **Features**.
5. Enable **In-app notifications**.

In-app notifications

Enables the app to poll for new in-app notifications and display those notifications as a toast or within the notification center. [Learn more](#)



Yes

[Reset to environment value](#)

6. Select **Save** to save the settings change.

7. Select **Publish** on the model-driven app.

Send basic in-app notifications

Notifications can be sent using the `SendAppNotification` message.

See [SendAppNotification Action](#) for information on the message and parameters.

The `SendAppNotification` message doesn't currently have request and response classes in the Dataverse SDK for .NET. To get strongly typed classes for this message, you must generate classes or use the underlying [OrganizationRequest](#) and [OrganizationResponse](#) classes. More information: [Use messages with the Organization service](#).

The `SendAppNotification` message uses open types, enabling dynamic properties on the in-app notification. For example, a notification can have zero to many actions, and each action may have different action types. Open types enable having dynamic properties for the actions depending on the action types selected. More information: [Use open types with custom APIs](#)

The following basic examples show how to use the API to send in-app notifications.

11 minutes ago



Welcome

Welcome to the world of app notifications!

This example uses the custom `Example.SendAppNotificationRequest` function described in [Creating a function for your client script](#) below.

JavaScript

```
var SendAppNotificationRequest = new Example.SendAppNotificationRequest(  
    title = "Welcome",  
    recipient = "/systemusers(<GUID of the user>)",  
    body = "Welcome to the world of app notifications!",  
    priority = 200000000,  
    iconType = 100000000,  
    toastType = 200000000,  
);  
  
Xrm.WebApi.online.execute(SendAppNotificationRequest).then(function  
(response) {  
    if (response.ok) {  
        console.log("Status: %s %s", response.status,  
response.statusText);  
  
        return response.json();  
    }  
})  
.then(function (responseBody) {  
    console.log("Response Body: %s", responseBody.NotificationId);  
})  
.catch(function (error) {  
    console.log(error.message);  
});
```

Notification polling

In-app notifications use polling to retrieve notifications periodically when the app is running. New notifications are retrieved at start of the model-driven app and when a page navigation occurs as long as the last retrieval is more than one minute ago. If a user stays on a page for a long duration, new notifications aren't retrieved until the user navigates to another page.

Notification table

Notifications sent using the `SendAppNotification` message are stored in the [Notification \(appnotification\) table](#) (Web API `appnotification`). The following are selected columns for the table.

Display Name	Schema Name	Description
Title	Title	The title of the notification.
Owner	OwnerId	The <i>user</i> who receives the notification. While this column can be set to either a user or team, you must only set this to a user. The notification can't be set to a team.
Body	Body	Details about the notification.
IconType	IconType	The list of predefined icons. The default value is <code>Info</code> . For more information, go to Changing the notification icon later in this article.
Toast Type	ToastType	The list of notification behaviors. The default value is <code>Timed</code> . For more information, go to Changing the notification behavior later in this article.
Priority	Priority	Enables prioritization of notifications, which determines the order in which the notifications are displayed in the notification center. For more information, see Changing the notification behavior later in this article.
Expiry (seconds)	TTLInSeconds	The number of seconds from when the notification should be deleted if not already dismissed.
Data	Data	JSON that's used for extensibility and parsing richer data into the notification. The maximum length is 5,000 characters.

ⓘ Note

The `appmoduleid` field is not used in the table.

Customizing the notification

In addition to the basic properties of the notification, you have options for customizing the notification delivered to the user. Customizing the notification involves changing the styles in the `Title` and `Body` of the notification, customizing the notification icon, and changing the behavior of the notification.

Using markdown in Title and Body

The `Title` and `Body` parameters of the `SendAppNotification` message don't support markdown defined within the properties. You can adjust the styles of these properties

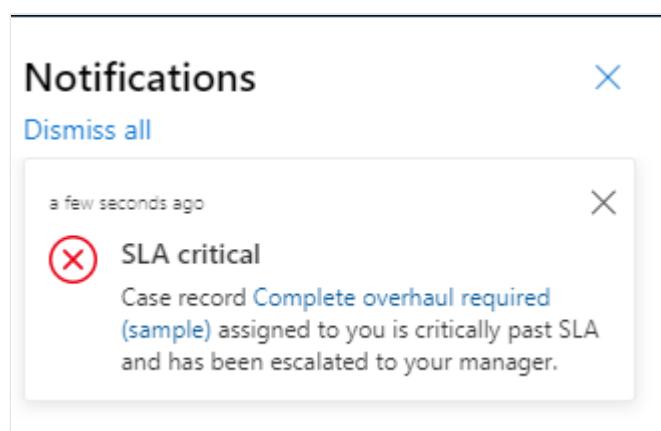
using markdown in the `OverrideContent` property. This field supports overriding the `Title` and `Body` simple strings with a limited subset of markdown styles.

The following is the supported markdown.

Text Style	Markdown
Bold	<code>**Bold**</code>
Italic	<code>_Italic_</code>
Bullet list	<code>- Item 1\r- Item 2\r- Item 3</code>
Numbered list	<code>1. Green\r2. Orange\r3. Blue</code>
Hyperlinks	<code>[Title](url)</code>

Newlines can be included with the body using `\n\n\n\n`.

This example shows how to create a notification by adding a custom body definition that includes an inline link.



Client API

This example uses the custom `Example.SendAppNotificationRequest` function described in [Creating a function for your client script](#) below.

JavaScript

```
var SendAppNotificationRequest = new  
Example.SendAppNotificationRequest(title = "SLA critical",  
    recipient = "/systemusers(<GUID of the user>)",  
    body = "Record assigned to you is critically past SLA.",  
    iconType = 100000003,  
    toastType = 200000000,  
    overrideContent = {  
        "@odata.type": "#Microsoft.Dynamics.CRM.expando",
```

```

        "title": "**SLA critical**",
        "body": "Case record [Complete overhaul required (sample)](?
pagetype=entityrecord&etn=incident&id=0a9f62a8-90df-e311-9565-
a45d36fc5fe8) assigned is critically past SLA and has been escalated to
your manager."
    }

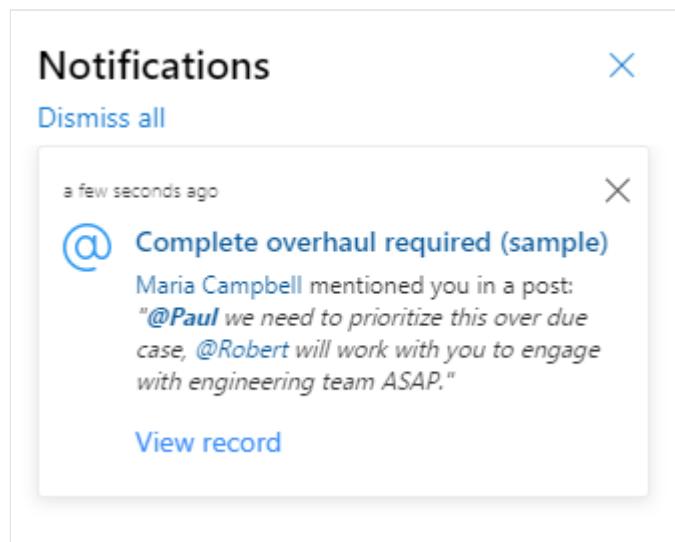
);

Xrm.WebApi.online.execute(SendAppNotificationRequest).then(function
(response) {
    if (response.ok) {
        console.log("Status: %s %s", response.status,
response.statusText);

        return response.json();
    }
})
.then(function (responseBody) {
    console.log("Response Body: %s", responseBody.NotificationId);
})
.catch(function (error) {
    console.log(error.message);
});

```

This example adds a custom title and a body definition that allows multiple links, bold formatting, and italic formatting.



Client API

This example uses the custom `Example.SendAppNotificationRequest` function described in [Creating a function for your client script](#) below.

JavaScript

```

var SendAppNotificationRequest = new
Example.SendAppNotificationRequest(title = "Complete overhaul required
(sample)",
    recipient = "/systemusers(<GUID of the user>)",
    body = "Maria Campbell mentioned you in a post.",
    priority = 200000000,
    iconType = 100000004,
    toastType = 200000000,
    expiry = 120000,
    overrideContent = {
        "@odata.type": "#Microsoft.Dynamics.CRM.expando",
        "title": "[Complete overhaul required (sample)](?pagetype=entityrecord&etn=incident&id=0a9f62a8-90df-e311-9565-a45d36fc5fe8)",
        "body": "[Maria Campbell](?pagetype=entityrecord&etn=contact&id=43m770h2-6567-ebm1-ob2b-000d3ac3kd6c) mentioned you in a post: _\\"**[@Paul](?pagetype=entityrecord&etn=contact&id=03f770b2-6567-eb11-bb2b-000d3ac2be4d)** we need to prioritize this overdue case, [@Robert](?pagetype=entityrecord&etn=contact&id=73f970b2-6567-eb11-bb2b-000d3ac2se4h) will work with you to engage with engineering team ASAP.\\"_"
    }
);
;

// Use the request object to execute the function
Xrm.WebApi.online.execute(SendAppNotificationRequest).then(function
(response) {
    if (response.ok) {
        console.log("Status: %s %s", response.status,
response.statusText);

        // Use response.json() to access the content of the response
body.
        return response.json();
    }
})
.then(function (responseBody) {
    console.log("Response Body: %s", responseBody.NotificationId);
})
.catch(function (error) {
    console.log(error.message);
    // handle error conditions
});

```

① Note

`OverrideContent` is not supported in Power Fx with the `xSendAppNotification` function.

Changing the notification behavior

You can change in-app notification behavior by setting `ToastType` to one of the following values.

Toast Type	Behavior	Value
Timed	The notification appears for a brief duration (the default is four seconds) and then disappears.	<code>200000000</code>
Hidden	The notification appears only in the notification center and not as a toast notification.	<code>2000000001</code>

Changing the notification icon

You can change the in-app notification icon by setting `IconType` to one of the following values. When using a custom icon, specify the `iconUrl` parameter within the `OverrideContent` parameter.

Icon Type	Value	Image
Info	<code>100000000</code>	
Success	<code>100000001</code>	
Failure	<code>100000002</code>	
Warning	<code>100000003</code>	
Mention	<code>100000004</code>	
Custom	<code>100000005</code>	

The following example demonstrates using Web API to send a notification with a custom icon.

HTTP

```

POST [Organization URI]/api/data/v9.2/SendAppNotification
HTTP/1.1
Content-Type: application/json; charset=utf-8
OData-MaxVersion: 4.0
OData-Version: 4.0
Accept: application/json

{
    "Title": "Welcome",
    "Body": "Welcome to the world of app notifications!",
    "Recipient": "/systemusers(<Guid of the user>)",
    "IconType": 100000005, // custom
    "OverrideContent": {
        "@odata.type": "#Microsoft.Dynamics.CRM.expando",
        "iconUrl": "/WebResources/cr245_AlertOn" //URL of the image file to be used for the notification icon
    }
}

```

Setting the notification priority

You can change the order in which notifications display in the notification center by setting the **Priority**. The following are the optional values:

Priority	Value
Normal	200000000
High	200000001

The default value is **Normal**. Notifications are sorted in the notification center by **Priority** and **Created On** date, descending. High priority notifications display at the top of the list in the notification center.

Notification actions

In-app notifications support zero to many actions on the notification card. There are three supported action types:

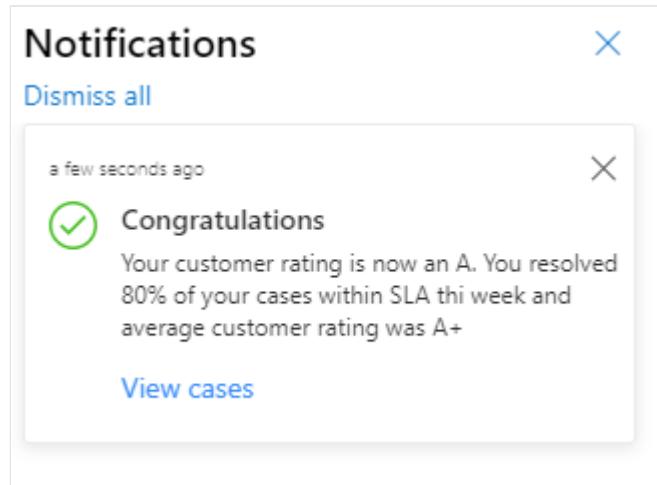
- **URL**: When the action is selected the web browser navigates to the defined URL.
- **Side Pane**: When the action is selected, a side pane is opened in the app and loads the defined context in the pane.
- **Teams Chat**: When the action is selected, a Teams chat is initiated with defined users in the context of a Dynamics 365 record.

Defining a URL action

The URL action type enables navigation from the action on the app notification to a defined URL. The following are the parameters for this action type:

Parameter	Required	Data type	Description
url	Yes	String	The URL of the web address to be opened when the action is selected.
navigationTarget	No	String	This parameter controls where a navigation link opens. The options are: <ul style="list-style-type: none">dialog: Opens in the center dialog.inline: Default. Opens in the current page.newWindow: Opens in a new browser tab.

The following example shows how to create a notification with a single URL action.



Client API

This example uses the custom `Example.SendAppNotificationRequest` function described in [Creating a function for your client script](#) below.

JavaScript

```
var SendAppNotificationRequest = new
Example.SendAppNotificationRequest(title = "Congratulations",
recipient = "/systemusers(<GUID of the user>)",
body = "Your customer rating is now an A. You resolved 80% of your
cases within SLA thi week and average customer rating was A+",
iconType = 100000001,
toastType = 200000000,
overrideContent = {
```

```

        "@odata.type": "#Microsoft.Dynamics.CRM.expando",
        "title": "**SLA critical**",
        "body": "Case record [Complete overhaul required (sample)](?
pagetype=entityrecord&etn=incident&id=0a9f62a8-90df-e311-9565-
a45d36fc5fe8) assigned is critically past SLA and has been escalated to
your manager."
    },
    actions = {
        "@odata.type": "Microsoft.Dynamics.CRM.expando",
        "actions@odata.type":
"#Collection(Microsoft.Dynamics.CRM.expando)",
        "actions": [
            {
                "title": "View cases",
                "data": {
                    "@odata.type": "#Microsoft.Dynamics.CRM.expando",
                    "type": "url",
                    "url": "?"
pagetype=entitylist&etn=incident&viewid=00000000-0000-0000-00aa-
000010001028&viewType=1039",
                    "navigationTarget": "newWindow"
                }
            }
        ]
    }
);

Xrm.WebApi.online.execute(SendAppNotificationRequest).then(function
(response) {
    if (response.ok) {
        console.log("Status: %s %s", response.status,
response.statusText);

        return response.json();
    }
})
.then(function (responseBody) {
    console.log("Response Body: %s", responseBody.NotificationId);
})
.catch(function (error) {
    console.log(error.message);
});

```

Defining a side pane action

A side pane action enables opening a side pane to load a defined page when the action is selected from the app notification. More information: [Creating side panes by using a client API](#) for more information.

When using the side pane action type, you have control over the options of the side pane itself, and the page that loads in the side pane.

- See [createPane](#) for the optional parameters for the pane that is created.
- See [navigateTo \(Client API reference\)](#) for the parameters to be defined for the page that is loaded in the side pane.

The following example shows creating an app notification with a two side pane actions.

Client API

This example uses the custom `Example.SendAppNotificationRequest` function described in [Creating a function for your client script](#) below.

JavaScript

```
var SendAppNotificationRequest = new
Example.SendAppNotificationRequest(title = "New task",
    recipient = "/systemusers(<GUID of the user>)",
    body = "A new task has been assigned to you to follow up on the
Contoso account",
    iconType = 100000000,
    toastType = 200000000,
    actions = {
        "@odata.type": "Microsoft.Dynamics.CRM.expando",
        "actions@odata.type":
"#Collection(Microsoft.Dynamics.CRM.expando)",
        "actions": [
            {
                "title": "View task",
                "data": {
                    "@odata.type": "#Microsoft.Dynamics.CRM.expando",
                    "type": "sidepane",
                    "paneOptions": {
                        "@odata.type":
"#Microsoft.Dynamics.CRM.expando",
                        "title": "Task Record",
                        "width": 400
                    },
                    "navigationTarget": {
                        "@odata.type":
"#Microsoft.Dynamics.CRM.expando",
                        "pageType": "entityrecord",
                        "entityName": "task",
                        "entityId": "<Task ID>"
                    }
                }
            }
        ]
    );
});
```

```

Xrm.WebApi.online.execute(SendAppNotificationRequest).then(function
(response) {
    if (response.ok) {
        console.log("Status: %s %s", response.status,
response.statusText);

        return response.json();
    }
})
.then(function (responseBody) {
    console.log("Response Body: %s", responseBody.NotificationId);
})
.catch(function (error) {
    console.log(error.message);
});

```

Defining a Teams chat action

A Teams chat action enables scenarios where a Teams chat is initiated from the app notification. This action uses the embedded Teams feature for Dynamics 365 apps, which provides sellers and agents the ability to chat in Microsoft Teams from within the customer engagement apps, such as Sales Hub, Customer Service Hub, and custom apps.

Note

Microsoft Teams chat in Dynamics 365 must be enabled to use the Teams chat action type. See [Work with Microsoft Teams chat in Dynamics 365](#) for more information.

The action type provides the following options:

- Create a new chat session or open an existing chat session.
- Link the chat session to a Dynamics 365 record or create an unlinked chat.

The following are the parameters for defining a Teams chat action on the app notification.

Parameter	Data type	Description
chatId	String	Define a value for the chat ID to open an existing chat. This is the ID of the Teams chat session, which can be obtained from the id property of the chat entity in Microsoft Graph. More information:

Parameter	Data type	Description
		<p>Get chat for more information. For Teams chat sessions that have been linked to Dynamics 365 records, the association is stored in the Microsoft Teams chat association entity (msdyn_teamschatassociation) table in Dataverse. The ID for the chat session is stored in the Teams Chat Id property of this table.</p> <p>Leave this parameter blank to initiate a new chat session.</p>
<code>memberIds</code>	GUID[]	An array of the Microsoft Azure Active Directory (AAD) user ID values of each of the participants that to be included in a new chat session. Member ID values shouldn't be defined if a value has been defined for the chatId parameter. If the chatId has been defined, then the existing chat is opened, and the members of the existing chat are included in the chat when opened.
<code>entityContext</code>	Expando	<p>The entity context provides the Dynamics 365 record to which the chat session should be linked. For example, if the chat session is regarding a specific customer account record, define the account record in this parameter to have the chat session linked to the account and display in the account's timeline.</p> <p>The entity context includes the entityName and recordId parameters, which must be defined to identify the record for the entity context.</p> <p>An entity context shouldn't be defined if a value has been defined for the chatId parameter. If the chatId has been defined, then the existing chat is opened, and the <code>entityContext</code>, whether linked or unlinked, will already have been defined for the existing chat. If the action is creating a new chat session (that is, the chatId parameter hasn't been provided), and the entity context isn't defined, then the new chat session won't be linked to a Dynamics 365 record.</p>
<code>entityName</code>	String	Part of the entity context, the logical name of the Dataverse table for the record to which the chat is related to.
<code>recordId</code>	GUID	Part of the entity context, this is the ID property of the table defined in the entityName parameter for the record to which the chat will be linked.
<code>chatTitle</code>	String	The title of the Teams chat.
<code>initialMessage</code>	String	The text of an introduction message you may optionally provide that will be automatically sent when the chat is created.

The following example shows creating an app notification with a single Teams chat action. When the action is selected on the toast notification it initiates the chat with the

participants defined. The chat is linked to a defined account record.

Client API

This example uses the custom `Example.SendAppNotificationRequest` function described in [Creating a function for your client script](#) below.

JavaScript

```
var SendAppNotificationRequest = new
Example.SendAppNotificationRequest(title = "New order posted",
    recipient = "/systemusers(<GUID of the user>)",
    body = "A new sales order has been posted for Contoso",
    iconType = 100000000,
    toastType = 200000000,
    actions = {
        "@odata.type": "Microsoft.Dynamics.CRM.expando",
        "actions@odata.type":
"#Collection(Microsoft.Dynamics.CRM.expando)",
        "actions": [
            {
                "title": "Chat with sales rep",
                "data": {
                    "@odata.type": "#Microsoft.Dynamics.CRM.expando",
                    "type": "teamsChat",
                    "memberIds@odata.type": "#Collection(String)",
                    "memberIds": ["<AAD User ID 1>", "<AAD User ID 2>"],
                    "entityContext": {
                        "@odata.type":
"#Microsoft.Dynamics.CRM.expando",
                        "entityName": "account",
                        "recordId": "<Account ID value>"
                    }
                }
            }
        ]
    }
);

Xrm.WebApi.online.execute(SendAppNotificationRequest).then(function
(response) {
    if (response.ok) {
        console.log("Status: %s %s", response.status,
response.statusText);

        return response.json();
    }
})
.then(function (responseBody) {
    console.log("Response Body: %s", responseBody.NotificationId);
})
.catch(function (error) {
```

```
        console.log(error.message);
    });
}
```

Creating a function for your client script

The client API examples in this topic provide examples of client scripting to send in-app notifications. The examples call a SendAppNotificationRequest function. To complete the examples, you will need to construct the reusable function in your environment. Below is an example of the function.

JavaScript

```
var Example = window.Example || {};
Example.SendAppNotificationRequest = function (
    title,
    recipient,
    body,
    priority,
    iconType,
    toastType,
    expiry,
    overrideContent,
    actions)
{
    this.Title = title;
    this.Recipient = recipient;
    this.Body = body;
    this.Priority = priority;
    this.IconType = iconType;
    this.ToastType = toastType;
    this.Expiry = expiry;
    this.OverrideContent = overrideContent;
    this.Actions = actions;
};

Example.SendAppNotificationRequest.prototype.getMetadata = function () {
    return {
        boundParameter: null,
        parameterTypes: {
            "Title": {
                "typeName": "Edm.String",
                "structuralProperty": 1
            },
            "Recipient": {
                "typeName": "mscrm.systemuser",
                "structuralProperty": 5
            },
            "Body": {
                "typeName": "Edm.String",
                "structuralProperty": 1
            }
        }
    };
}
```

```

        },
        "Priority": {
            "typeName": "Edm.Int",
            "structuralProperty": 1
        },
        "IconType": {
            "typeName": "Edm.Int",
            "structuralProperty": 1
        },
        "ToastType": {
            "typeName": "Edm.Int",
            "structuralProperty": 1
        },
        "Expiry": {
            "typeName": "Edm.Int",
            "structuralProperty": 1
        },
        "OverrideContent": {
            "typeName": "mscrm.expando",
            "structuralProperty": 5
        },
        "Actions": {
            "typeName": "mscrm.expando",
            "structuralProperty": 5
        },
    },
    operationType: 0,
    operationName: "SendAppNotification",
};

}

```

For more examples of client scripting using client API, see [Walkthrough: Write your first client script](#).

Managing security for notifications

The in-app notification feature uses three tables. A user needs to have the correct security roles to receive notifications and to send notifications to themselves or other users.

In addition to the appropriate table permissions, a user must be assigned the **Send In-App Notification** `prvSendAppNotification` privilege to execute the `SendAppNotification` message. The privilege is granted to the **Environment Maker** role by default. This privilege is required for sending in-app notifications. It isn't required to receive notifications.

Usage	Required table privileges
User has no in-app notification bell and receives no in-app notification	None: Read privilege on the app notification table.
User can receive in-app notifications	<ul style="list-style-type: none"> • Basic: Read privilege on the app notification table. • Create, Read, Write, and Append privileges on the model-driven app user setting. • Read and AppendTo privileges on setting definition.
User can send in-app notifications to self	Basic: Create and Read privileges on the app notification table, and Send In-App Notification privilege.
User can send in-app notifications to others	Read privilege with Local, Deep, or Global access level on the app notification table based on the receiving user's business unit, and Send In-App Notification privilege.
User can delete in-app notifications	Global: Delete privileges on the app notification table.

Notification storage

Because the app notification table uses the organization's database storage capacity, it's important to consider the volume of notifications sent and the expiration setting. More information: [Microsoft Dataverse storage capacity](#)

In-app notifications vs. push notifications

The Power Apps Notification connector is for push notifications, which are separate from in-app notification. Push notifications only appear on the mobile device notifications list to open the app. In-app notifications appear when the app is open. We recommend limiting the use of push notifications to high-priority items, to avoid overwhelming the user. For more information, go to:

- [Power Apps Notification connector](#)
- [Power Apps Notification V2 connector](#)
- [Create push notifications for Power Apps Mobile](#)

See also

[SendAppNotification Action](#)

[Create a table row using the Web API](#)

[createRecord \(Client API reference\)](#)

[In-app notifications in model-driven apps](#)

[appnotification EntityType](#)

[Notification \(appnotification\) table/entity reference](#)

Creating side panes by using a client API

Article • 08/04/2023

Developers can create and manage app side panes within a model-driven app by using the [Xrm.App.sidePanes](#) API, which represents the collection of side panes. Calling the [createPane](#) method adds a new pane that allows navigation to any model-driven app form or custom page. Pages displayed in the side pane must fit within the minimum width of 300 pixels and resize to larger widths based on the pane width.

Tabs are listed in the side pane in two groups: nonclosable and closable. Within each group, the tabs are listed in the order they were created in. The top group contains the panes that a user can't close, and the bottom group has user-closable panes. The nonclosable group is populated when the app is opened, whereas the closable group is added based on user actions within the app.

You can use a platform-provided header with the title and Close button, or you can use a custom header.

You can add a badge to the side pane to indicate to the user that a change needs attention. The badge supports three modes: a simple dot, a count, or an image. By default, the badge is cleared when the user switches to the pane. You can control when the badge is cleared.

App side panes are only supported in web browsers and are prevented within native players.

ⓘ Note

This API is not available for Dynamics 365 Customer Engagement on-premise deployments. More information: [Client APIs not supported in Customer Engagement \(on-premises\)](#)

Examples

Showing a default view in the side pane

The examples in this section show how to display the default view of a table in the app side pane. A reservation list and product list are opened as nonclosable panes.

JavaScript

```
Xrm.App.sidePanes.createPane({
    title: "Reservations",
    imageSrc: "WebResources/sample_reservation_icon",
    paneId: "ReservationList",
    canClose: false
}).then((pane) => {
    pane.navigate({
        pageType: "entitylist",
        entityName: "sample_reservation",
    })
});
```

Reservations

Active Reservations

Initials	Name	Item	Date
AP	Ammar Peterson	Notebook	8/3/2018
AG	Anthony Glaze	LCD Monitor	4/28/2018
AP	Antonio Pinto	Notebook	12/7/2018
BM	Betsy McCarthy	Laptop	11/6/2018
CB	Carlene Blankenship	Wired keyboard	10/5/2018
DS	Deborah Schultz	Laptop	3/27/2018
GE	Graciela Esteban	Laptop	5/30/2018
LC	Lila Clay	LCD Monitor	6/11/2018
LC	Luisa Cummings	Laptop	4/15/2018

JavaScript

```
Xrm.App.sidePanes.createPane({
    title: "Products",
    imageSrc: "WebResources/sample_product_icon",
    paneId: "ProductList",
    canClose: false
}).then((pane) => {
    pane.navigate({
        pageType: "entitylist",
        entityName: "sample_product",
    })
});
```

Products

Active Products

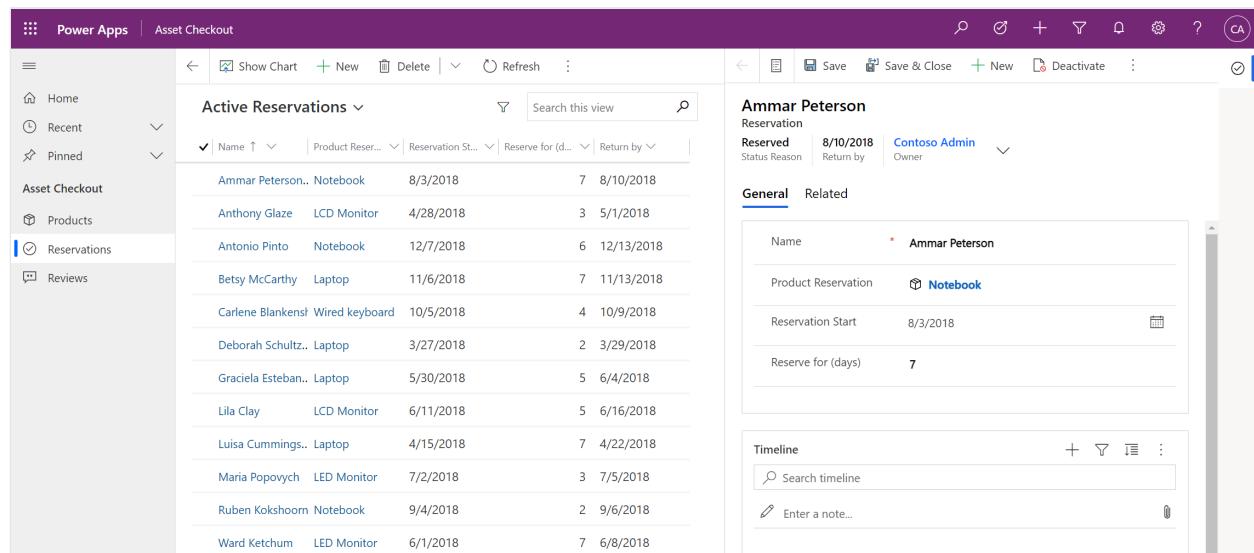
Image	Name	Category	Color	More
Laptop	C1	White	⋮	
LCD Monitor	C4	Black	⋮	
LED Monitor	C4	White	⋮	
Notebook	C1	Black	⋮	
Wired keyboard	C3	Black	⋮	
Wired mouse	C2	Black	⋮	
Wireless keyboard	C3	White	⋮	
Wireless mouse	C2	White	⋮	

Showing a table row

This example shows how to display a row in the side pane. A reservation row is opened in a side pane where the default header is hidden and the width is customized to 600 pixels.

```
JavaScript

Xrm.App.sidePanes.createPane({
    title: "Reservation: Ammar Peterson",
    imageSrc: "WebResources/sample_reservation_icon",
    hideHeader: true,
    canClose: true,
    width: 600
}).then((pane) => {
    pane.navigate({
        pageType: "entityrecord",
        entityName: "sample_reservation",
        entityId: "d4034340-4623-e811-a847-000d3a30c619",
    })
});
```



Showing a web resource

This example shows how to display a webresource in the side pane. A custom html page is opened in a side pane where the default header is visible & close button is hidden.

```
JavaScript

Xrm.App.sidePanes.createPane({
    title: "Registration Page",
    imageSrc: "WebResources/msfp_SurveyIcon_32",
    paneId: "RegistrationPage",
    canClose: false
});
```

```
        }).then((pane) => {
            //navigate to webresource
            pane.navigate({
                pageType: "webresource",
                webresourceName: "new_RegistrationPage",
            })
        });
    );
```

Managing side panes

In addition to creating side panes and showing rows or views within the side pane, you can also:

- Use the `state` method to collapse the side pane programmatically:

```
Xrm.App.sidePanes.state = 0;
```

- Use the `state` method to expand the side pane programmatically:

```
Xrm.App.sidePanes.state = 1;
```

- Change properties by retrieving the selected pane:

```
var lastPane = Xrm.App.sidePanes.getSelectedPane();
lastPane.width = 400;
```

- Retrieve a specific pane by using the `paneId` parameter:

```
var reservationPane = Xrm.App.sidePanes.getPane("ReservationList");
reservationPane.close();
```

- Enable the badge property on a pane:

```
Xrm.App.sidePanes.getSelectedPane().badge = 1;
```

Use with Xrm.App.panels.loadPanel

The [Xrm.Panels.loadPanel](#) API is being replaced with [Xrm.App.sidePanes.createPane](#) because the former only supports a single pane while the latter supports multiple panes. To enable transitioning from `loadPanel` to `createPane`, the two can work together with some limitations. If only `loadPanel` is used within a model-driven app, then the experience remains the same. However if both `loadPanel` and `createPane` are used, the first limitation is that a placeholder icon is shown for the `loadPanel`. The second limitation is that when the user switches from the `loadPanel` to the `createPane`, the

`loadPanel` content is unloaded to save memory and is reloaded on switch back without the state. This tab switch behavior is the same used within the multi-session app mode to manage the memory used by the app. Most page types restore correctly however when an external site or web resource is opened the state isn't restored.

By switching to use `createPane`, both limitations can be avoided by providing an icon and by enabling `alwaysRender`. The `alwaysRender` keeps the pane content when the user switches but does take more memory so should be used sparingly.

Related articles

[sidePanes \(Client API reference\)](#)

[loadPanel \(Client API Reference\)](#)

[Walkthrough: Write your first client script](#)

Get or update a setting value using client API

Article • 02/03/2023

Settings are solution components that enable developers to quickly configure apps to provide a customized experience. Settings can be used to enable or disable features or configure feature behavior for a single app or all apps within an environment. More information: [Use settings to provide customized app experiences](#)

The following functions can be used to get or update a setting value using client API.

getCurrentAppSetting

Gets the value of a setting for the current app.

Syntax

JavaScript

```
var settingValue =  
Xrm.Utility.getGlobalContext().getCurrentAppSetting(settingName);
```

Parameters

Name	Type	Required	Description
settingName	String	Yes	The name of the setting to get the value for.

Return value

Type: Same as the type of the setting: *Number*, *String*, or *Yes/No*

Description:

- If the setting is *Overridable* and *Value can be overridden* is set to *Environment and app* the setting app value is returned. If a setting app value does not exist, then the setting environment value is returned. If a setting environment value does not exist, the default value as specified in the setting definition is returned.

- If the setting is *Overridable* and "Value can be overridden" is set to "Environment only" the setting environment value is returned. If a setting environment value does not exist, the default value as specified in the setting definition is returned.
- If the setting is *Overridable* and "Value can be overridden" is set to "App only" the setting app value is returned. If a setting app value does not exist, the default value as specified in the setting definition is returned.
- If the setting is not *Overridable*, the default value as specified in the setting definition is returned.
- If the setting name is incorrect or the setting could not be found, the return value is null.

saveSettingValue

Adds or updates the setting app value for the current app or the setting environment value for the current environment.

Syntax

JavaScript

```
var appOverrideScope = 2; // Add or update a setting app value
var saveSettingOptions = {overrideScope: appOverrideScope,
solutionUniqueName: mySolutionName};
Xrm.Utility.getGlobalContext().saveSettingValue(settingName, value,
saveSettingOptions).then(successCallback, errorCallback);
```

Parameters

Name	Type	Required	Description
settingName	String	Yes	The name of the setting to update the value of.
value	Number, String, or Yes/No	Yes	The value to update the setting to.

Name	Type	Required	Description
saveSettingOptions	String	No	<p>Options when updating the value. It contains two parameters</p> <ul style="list-style-type: none"> • overrideScope <ul style="list-style-type: none"> ◦ Use 1 to add or update a setting environment value ◦ Use 2 to add or update a setting app value. ◦ If not specified it is set to environment. • solutionUniqueName <ul style="list-style-type: none"> ◦ The solution to which the setting environment value or setting app value should be added. ◦ If not specified the default solution is used.
successCallback	String	Yes	A function to call if the update is successful.
errorCallback	String	Yes	A function to call if the update fails.

Return value

On success, returns a promise object.

Web APIs

You can also use the following Web APIs to get or update a setting value

- [RetrieveSetting Function](#)
- [SaveSettingValue Action](#)

See also

[Solutions overview](#)

[Use settings to provide customized app experiences](#)

Client API Reference for model-driven apps

Article • 11/30/2022

This section contains reference documentation for client API object model that can be used with JavaScript libraries.

Important

- The Client API object model also contains the **Xrm.Internal** namespace, and use of the objects/methods in this namespace isn't supported. These objects, and any parts of the HTML Document Object Model (DOM), are subject to change without notice. We recommend that you don't use these functions or any script that depends on the DOM.
- Also, while debugging, you may find methods and objects in the Client API object model that aren't documented. Only documented objects and methods are supported.
- Most of the client scripting APIs available in this documentation also apply to Dynamics 365 Customer Engagement (on-premises). For a list of client scripting APIs not available for Customer Engagement (on-premises), see [Client scripting reference for Dynamics 365 Customer Engagement \(on-premises\)](#).

The topics under this section are organized as follows:

- Starts with reference for all the events, collections, and the execution context object.
- Continues on to provide information about methods for **attributes** and **controls** in Customer Engagement that are actually collections that appear under different objects in the Client API object model.
- Provides reference for properties and methods for the **formContext** and **gridContext** objects.
- Finally provides reference for namespaces in the **Xrm** object model.

Related topics

[Apply business logic using client scripting in model-driven apps](#)

[Understand the Client API object model](#)

[Model-driven apps Developer Overview](#)

Events (Client API reference)

Article • 11/30/2022

Events occur in Custom Engagement forms and grids whenever a form or grid loads, data is changed, or saved. You execute your JavaScript code by associating it an events so that it is executed when the event occurs. More information: [Events in forms and grids](#)

Column event

- [OnChange](#)

Control event

- [OnOutputChange](#)

Form events

- [OnLoad](#)
- [OnSave](#)

Form data event

- [OnLoad](#)

Grid and subgrid events

- [OnChange](#)
- [OnLoad](#)
- [OnRecordSelect](#)
- [OnSave](#)

IFRAME control event

- [OnReadyStateComplete](#)

Knowledge base search control events

- [OnResultOpened](#)
- [OnSelection](#)
- [PostSearch](#)

Lookup control event

- [OnLookupTagClick](#)
- [PreSearch](#)

Process events

- [OnProcessStatusChange](#)
- [OnStageChange](#)
- [OnStageSelected](#)

Tab event

- [TabStateChange](#)

Related topics

[Events in forms and grids](#)

Column OnChange event (Client API reference)

Article • 11/30/2022

The `OnChange` event occurs in the following situations:

- Data in a form column has changed and focus is lost. There is an exception to this behavior that applies to Yes/No columns that are formatted to use radio buttons or check boxes. In these cases the event occurs immediately.
- Data changes on the server are retrieved to update a column when the form is refreshed, such as after a record is saved.
- The `attribute.fireOnchange` method is used.

All columns support the `OnChange` event. Data in the column is validated before and after the `OnChange` event.

The `OnChange` event does not occur if the column is changed programmatically using the `attribute.setValue` method. If you want event handlers for the `OnChange` event to run after you set the value you must use the `formContext.data.entity.attribute.fireOnchange` method in your code. The `OnChange` event also does not occur if the column is changed programmatically when discarding changes if the user is navigating away from a dirty form.

ⓘ Note

Although the **Status** column supports the `OnChange` event, the column is read-only on the form so the event cannot occur through user interaction. Another script could cause this event to occur by using the `fireOnchange` method on the column.

ⓘ Note

`OnChange` events are synchronous. You should **not** use asynchronous code in an `OnChange` event handler that needs an action to be taken or handled on the resolution of the async code. This causes issues if the resolution handler expects the app context to remain the same as it was when the asynchronous code was started. You should also **not** make synchronous network requests in an `OnChange` event handler. This can cause an unresponsive app.

Methods supported for this event

There are three methods you can use to work with the `OnChange` event for a column:

- [addOnChange](#)
- [fireOnChange](#)
- [removeOnChange](#)

Related topics

[Columns \(Client API reference\)](#)

Form OnLoad event

Article • 05/14/2023

This event occurs whenever the form is loaded, specifically:

- On initial page load.
- After a new record is first saved (created).

Use the `formContext.ui.addOnLoad` and `formContext.ui.removeOnLoad` methods to manage event handlers for this event.

ⓘ Note

Controls in a form may not be ready when a form's `OnLoad` event occurs. Use the `OnLoad` event of the control to wait for it to be ready. More information: [Add or remove event handler function to event using UI](#)

Asynchronous OnLoad event handler support

The `OnLoad` event handler has the ability to wait for promises returned by event handlers to settle before loading a form which allows for an `OnLoad` event to be asynchronous ("async"). The `OnLoad` event becomes `async` when the event handler returns a promise.

The form loads when each promise returned by the event handler is resolved. For any promises that are returned, there is a 10 second limit for each promise. After that, the platform considers promises to be timed out. This timeout is applied per promise. For example, if you have five promises returned, the total wait time is 50 seconds. Suppose the promise is rejected or timed out. In that case, the form load operation behaves similarly to the current script errors.

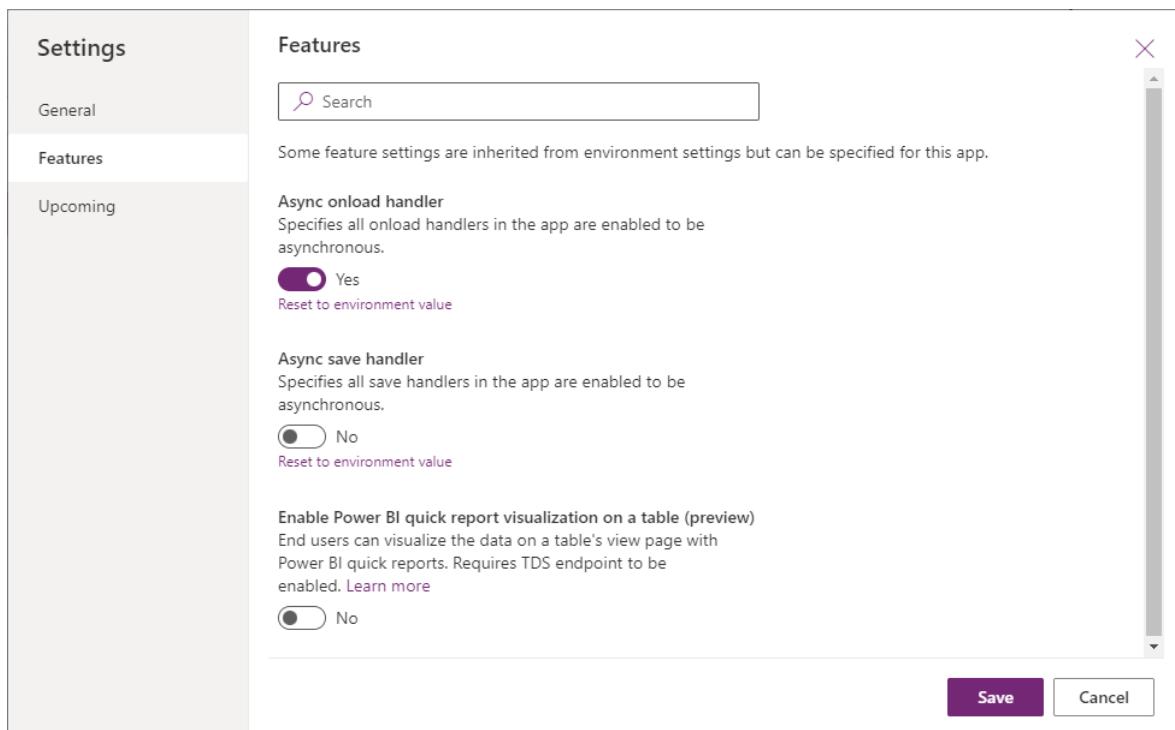
The `OnLoad` event will only wait for one promise returned per handler. If multiple promises are required, it is recommended to wrap all the promises in the `Promise.all()` method and return the single resulting promise. For multiple handlers that return a promise, we recommend that you create one handler that calls all the events and return a single promise that wraps all required promises. This is to minimize wait times caused by the timeout.

Enable Async OnLoad using app setting

To use async onLoad handlers, you will need to enable it through the app setting. An app setting is a platform component that allows you to turn supported features on or off for your app. To enable the async Onload event handlers for a specific app:

1. Go to <https://make.powerapps.com>.
2. Make sure you select the correct environment.
3. Select **Apps** from the left navigation pane.
4. Select the app and then select ... (ellipses). Select **Edit**.
5. Select **Settings** in the command bar.
6. When the dialog opens, select **Features**.
7. Turn on **Async onload handler**.

8. Select **Save**.



Async OnLoad timeouts

When using an async handler, a form load will wait for the promise to be fulfilled, but only up to 10 seconds. This is to ensure that the form loads within a reasonable amount of time.

There may be scenarios where you want to pause OnLoad for a longer period of time. An example is opening a dialog in the async OnLoad handler and waiting for the user's input before saving. To make sure the async operation will wait, you can invoke the

event argument **disableAsyncTimeout** as follows

`executioncontext.getEventArgs().disableAsyncTimeout()`. When **disableAsyncTimeout** is set, the timeout for that handler will not be applied. It will continue to wait for that handler's promise to be fulfilled.

This should be used with caution as it might affect the performance of the form load.

Form OnSave event (Client API reference) in model-driven apps

Article • 07/26/2023

The `OnSave` event occurs when:

- The user selects the **Save** or **Refresh** button in the command bar, even when there's no changed data to be saved.
- Code executes the `formContext.data.entity.save` method, even when there's no changed data to be saved.
- The user navigates away from the form and there's unsaved data in the form.
- The AutoSave option is enabled, 30 seconds after data has changed and there's unsaved data in the form.
- Code executes the `formContext.data.save` method and there's unsaved data in the form.
- Code executes the `formContext.data.refresh` method passing a true value as the first parameter and there's unsaved data in the form.

ⓘ Note

The `OnSave` event for appointment, recurring appointment, or service activity records will cancel the save operation and use the `Book` message to persist the change rather than `Create` or `Update`. Because of this, `OnSave` and `PostSave` event handlers for these tables will not work.

To determine which button was clicked to perform the save, use the [getSaveMode method](#) method.

You can cancel the save action by using the `preventDefault` method within the event arguments object. The `preventDefault` method is accessible by using the `getEventArgs` method that is part of the execution context. Execution context is automatically passed to the form event handler.

Asynchronous event handler support

The OnSave event has ability to wait for promises returned by event handlers to settle before saving, allowing the `OnSave` event to be asynchronous ("async").

The `OnSave` event becomes async when the `OnSave` event handler returns a promise. Saving of the record happens when each promise returned by a handler is resolved. For any promises that are returned, there's a 10-second limit for each promise, after that the platform considers promises to be timed out. This timeout is applied per promise. For example, if we have five promises returned, the total wait time is 50 seconds.

If the promise is rejected or timed out, the save operation continues to behave similarly to the current script errors. Use the `preventDefault` method within the event arguments object in that particular handler if you want to prevent the save event to happen if there's a script error/rejected promise or handler times out.

You can also cancel the save operation irrespective of the error in the handler or not using the `preventDefault` method within the event arguments object. If this method is called, the Async OnSave event waits for all the promises to settle, but the save won't occur. Calling this method means the logic within `.then()` & `.catch()` will execute.

The `OnSave` event waits for one promise returned per handler. If multiple promises are required, it's recommended to wrap all the promises in the `Promise.all()` method and return the single resulting promise. For multiple handlers that all return a promise, we recommend you to create one handler that calls all the events and return a single promise that wraps all required promises. This practice is to minimize wait times caused by the timeout.

Example scenario on when to use async OnSave handlers

Consider creating a Work Order Service Task, you need to validate that the Customer Asset selected has the same account listed in the Work Order. Fetching the account on the Work Order and Customer Asset are both asynchronous processes and need to be completed before the validation can occur.

In this scenario, since there are multiple async processes and both calls return a single promise by wrapping both in the `Promise.all()` method.

ⓘ Note

The `preventDefault` method can only be used synchronously.

For example:

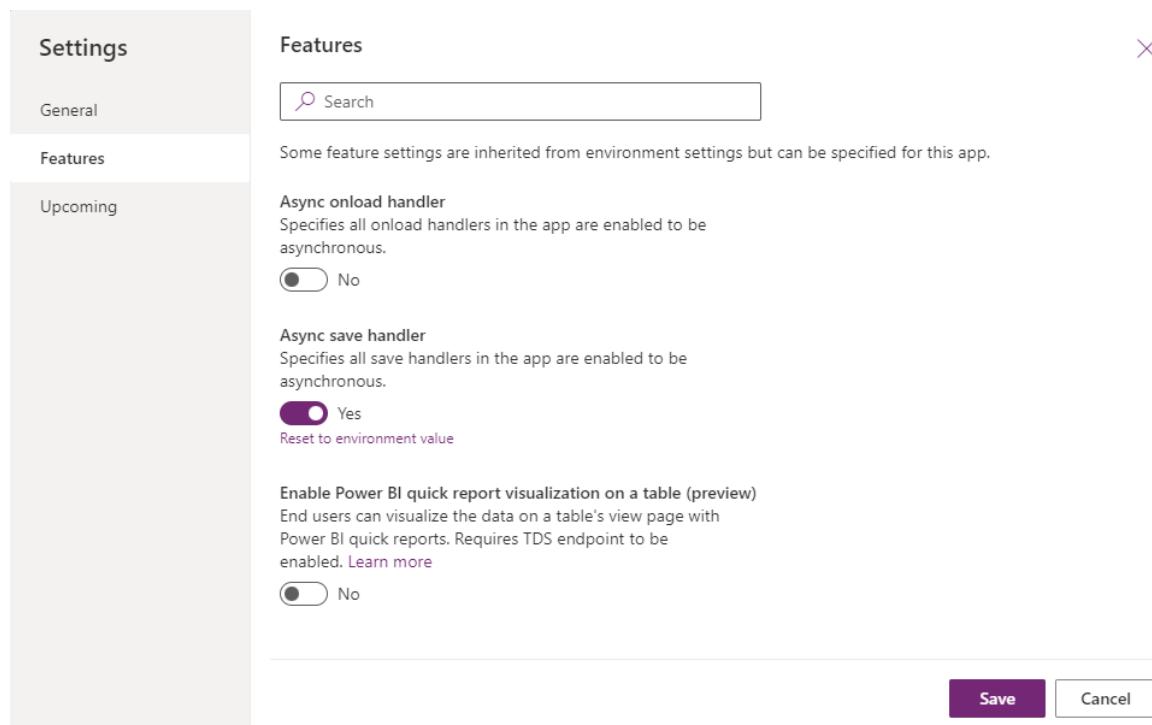
```
JavaScript
```

```
function myHandler(context) {
    return new Promise((resolve) => {
        setTimeout( () => {
            context.getEventArgs().preventDefault();
        }, 1000);
    });
}
```

Enable Async OnSave using app setting

To use async onSave handlers, you need to enable it through an app setting:

1. Go to <https://make.powerapps.com>.
2. Make sure select the correct environment.
3. Select **Apps** from the left navigation pane.
4. Select the app and then select ... (ellipses). Select **Edit**.
5. Select **Settings** from the command bar.
6. When the dialog opens, select **Features**.
7. Turn on **Async onSave handler**.
8. Select **Save**.



Async OnSave timeouts

When you use an async `OnSave` handler, the form waits for the promise returned by the handler to be fulfilled. To ensure that the form save completes in a timely manner, the handler will throw a timeout exception after 10 seconds to let you know to tune the async `OnSave` handler for better performance.

There are scenarios where pausing the `OnSave` handler for longer than 10 seconds is needed. An example is opening a dialog and waiting for the user's input before continuing to save. To make sure the async operation waits for the promise to resolve, use the `disableAsyncTimeout` method.

ⓘ Note

You must call `disableAsyncTimeout` before any await statements or async calls.

For example:

JavaScript

```
async function myHandler(context) {
    context.getEventArgs().disableAsyncTimeout();
    // The 10000ms time out will not be disabled if the above line does not
    // come before all async awaits
    await Xrm.Navigation.openConfirmDialog({ text: "Are you sure you want
        to save?" });
}
```

When `disableAsyncTimeout` is called, the timeout for that handler isn't applied. It continues to wait for that handler's promise to be fulfilled.

This pattern should be used with caution as it might affect the performance of the form save.

Related article

[Grid OnSave Event](#)

Form data OnLoad event (Client API reference)

Article • 11/30/2022

This event occurs whenever form data is loaded, specifically:

- On initial page load.
- When the page data is explicitly refreshed using `formContext.data.refresh` method.
- When the data is refreshed on a page on saving a record, if there are any changes.

Use the `formContext.data.addOnLoad` and `formContext.data.removeOnLoad` methods to manage event handlers for this event.

Grid OnChange event (Client API reference)

Article • 12/16/2022

The `OnChange` event occurs when a value is changed in a cell in the editable grid and the cell loses focus. This event can also occur when an attribute value is updated using the `setValue` method.

Grid OnRecordSelect event (Client API reference)

Article • 07/12/2023

The `OnRecordSelect` event occurs when a single row (record) is selected in an editable grid. This event doesn't occur if a user selects different cells in the same row, or selects multiple rows.

Example: Override the default open behavior in model-driven grids

When you want to customize the way that a table record opens from the [Power Apps grid control](#), you can control how this opens with a JavaScript function associated with the grid `OnRecordSelect` event.

The following example ensures that the record opens using the form specified by the `pageInput formId` value using the [Xrm.Navigation.navigateTo](#) method. In this example, the form and grid must belong to the same entity.

Step 1: Create a web resource

Create, save, and publish a JavaScript (JS) web resource that contains the following code:

JavaScript

```
var Example = window.Example || {};
(function () {
    this.OnSelect = function (executionContext) {
        var pageInput = {
            pageType: "entityrecord",
            entityName: executionContext.getEventSource().getEntityName(),
            entityId: executionContext.getEventSource().getId(),
            formId: "420786E3-D342-4A9A-914B-AA331FF2D25E"
        };
        Xrm.Navigation.navigateTo(pageInput);
    }
}).call(Example);
```

More information: [Create or edit model-driven app web resources](#)

Step 2: Enable Power Apps Grid Control

Follow these steps to enable the **Power Apps grid control** as the main grid (table view) or within a model-driven form subgrid:

- [Use as main grid](#)
- [Use as subgrid](#)

Step 3: Register the custom behavior on OnRecordSelect Event

When you enable the **Power Apps grid control**, an **Events** tab appears. Select the **Events** tab:

1. Under the **Form Libraries** section, add the Form Library from the web resource created.
2. Under the **Event Handlers** section, select the event **OnRecordSelect** and select **Add** and a popup appears.
3. In the popup, select the form library just added and the function name `Example.OnSelect`. This is the name of the JavaScript function created in the web resource. Make sure to check the option **Pass execution context as first parameter**.

More information: [Power Apps grid control](#)

Grid OnSave event (Client API reference)

Article • 12/16/2022

The `OnSave` event occurs before sending the updated information to the server, and when any of the following occurs:

- There is a change in the record selection.
- The user explicitly triggers a save operation using the editable grid's save button.
- The user applies a sort, filter, group, pagination, or navigation operation from the editable grid while there are pending changes.

Some important points to consider for the `OnSave` event:

- If a user edits multiple columns of the same record in sequence, the `OnSave` event will only be fired once to ensure optimal performance and form behavior compatibility.
- Editable grid and the parent form have separate save buttons. Selecting the save button in one will not save changes in the other.
- Editable grid does not save pending changes when navigation operations are performed outside of its context. If the control has unsaved data, that data may be lost. Consequently, the `OnSave` event may not fire. For example, this could happen when navigating to a different record using a form lookup column or through the ribbon.
- Selecting the refresh button in the editable grid causes it to discard any pending changes, and the `OnSave` event won't be fired.
- Editable grid control does not implement an auto-save timer. Editable grid suppresses duplicate detection rules.

ⓘ Note

The `OnSave` event for appointment, recurring appointment, or service activity records will cancel the save operation and use the `Book` message to persist the change rather than `Create` or `Update`. Because of this, `OnSave` and `PostSave` event handlers for these tables will not work.

Related topic

[Form OnSave Event](#)

OnLookupTagClick Event (Client API reference)

Article • 12/16/2022

This event occurs when the user clicks the tag in a lookup control.

An execution context object is passed to event handlers for this event. You can use the [getEventArgs](#) method to retrieve an object that has the [getTagValue](#) method.

The [getTagValue](#) method returns an object with the following properties:

- **name**. String. Name of the tag.
- **id**: String. ID of the tag.
- **entityType**. String. Table type of the tag.
- **fieldName**. String. The originating lookup column that raised the event.

Methods supported for this event

- [addOnLookupTagClick](#) method to add event handlers for this event.
- [removeOnLookupTagClick](#) method to remove event handlers for this event.

OnOutputChange event (Client API reference)

Article • 08/18/2022

This event occurs when an output property control changes.

Use the [addOnOutputChange](#) and [removeOnOutputChange](#) methods to add and remove event handlers respectively for this event.

OnProcessStatusChange event (Client API reference)

Article • 12/16/2022

This event occurs when the status of a process instance changes.

Use the [formContext.data.process.addOnProcessStatusChange](#) method to add event handlers for this event and the [formContext.data.process.removeOnProcessStatusChange](#) method to remove them.

onPreProcessStatusChange event (Client API reference)

Article • 12/16/2022

Applies to Dynamics 365 (online), version 9.x

This event occurs **before** the status of a process instance changes.

Use the [formContext.data.process.addOnPreProcessStatusChange](#) method to add event handlers for this event and the

[formContext.data.process.removeOnPreProcessStatusChange](#) method to remove them.

From within a web resource script registered to the onPreProcessStatusChange event, a developer can invoke the following on the executionContext object passed into the web resource script:

```
executionContext.getEventArgs().preventDefault();
```

When you invoke `preventDefault`:

- The state change will not be processed. The process instance will remain on the original stage in the original state.
- The save of the main form will not be processed. If the main form was in a dirty state, it would remain in a dirty state.
- Any web resources that registered onProcessStatusChange will not be invoked.

This client API is only supported on the unified client. The legacy web client does not support this client API.

Methods supported for this event

- [formContext.data.process.addOnPreProcessStatusChange](#) method to add event handlers for this event.
- [formContext.data.process.removeOnPreProcessStatusChange](#) method to remove event handlers for this event.

OnPreStageChange event (Client API reference)

Article • 12/16/2022

This event occurs **Before** the stage of a business process flow control changes. This event occurs after the user selects the **Next Stage**, **Move to previous stage** or **Set Active Stage** buttons in the user interface or when a developer uses the

`formContext.data.process.moveNext`, `formContext.data.process.movePrevious`, or
`formContext.data.process.setActiveStage` methods.

ⓘ Note

The OnPreStageChange event is supported only on Unified Interface.

From within a web resource script registered to the `onPreStageChange` event, a developer can invoke the following on the `executionContext` object passed into the web resource script:

```
executionContext.getEventArgs().preventDefault();
```

When you invoke `preventDefault`:

- The stage navigation will not be processed. The process instance will remain on the original stage.
- In a cross-table navigation, the form of the table of the destination stage will not open.
- The save of the main form will not be processed. If the main form was in a dirty state, it would remain in a dirty state.
- Any web resources that registered `onStageChange` will not be invoked.

An execution context object is passed to event handlers for this event. You can use the [getEventArgs](#) method to retrieve an object that has the following methods:

- **getDirection**: Returns a string that is either `Next` or `Previous` to show the direction of the stage change.
- **getStage**: Returns a stage object. Except when the navigation moves to a new table, the stage returned represents the destination stage object—that is, the next active stage. When the navigation moves to a new table, the stage is the stage being navigated from—that is, the previous active stage object. More information: [Stage methods](#).

Methods supported for this event

- `formContext.data.process.addOnPreStageChange` method to add event handlers for this event.
- `formContext.data.process.removeOnPreStageChange` method to remove event handlers for this event.

IFRAME OnReadyStateComplete event (Client API reference)

Article • 12/16/2022

The `OnReadyStateComplete` event indicates that the content of the IFRAME has loaded and can be accessed in code. Use this event when referencing IFRAME controls within your scripts.

OnResultOpened event (Client API reference)

Article • 12/16/2022

This event occurs when a knowledge base article is opened in the knowledge base search control in line or through the pop-out action. Use the [addOnResultOpened](#) and [removeOnResultOpened](#) methods to manage event handlers for this event.

OnSelection event (Client API reference)

Article • 12/16/2022

This event occurs when a knowledge base article is selected in the knowledge base search control.

Use the [addOnSelection](#) and [removeOnSelection](#) methods to add and remove event handlers respectively for this event.

OnStageChange Event (Client API reference)

Article • 12/16/2022

This event occurs when the stage of a business process flow control changes. This event occurs when the user clicks the **Next Stage** or **Move to previous stage** buttons in the user interface or when a developer uses the `formContext.data.process.moveNext` or `formContext.data.process.movePrevious` methods. You can't cancel the stage change using code in a handler for this event.

An execution context object is passed to event handlers for this event. You can use the [getEventArgs](#) method to retrieve an object that has the following methods:

- **getDirection**: Returns a string that is either "next" or "previous" to show the direction of the stage change.
- **getStage**: Returns a stage object. Except when the navigation moves to a new table, the stage returned represents the destination stage object, that is, the next active stage. When the navigation moves to a new table, the stage is the stage being navigated from, that is, the previous active stage object. More information: [Stage methods](#).

Methods supported for this event

- `formContext.data.process.addOnStageChange` method to add event handlers for this event.
- `formContext.data.process.removeOnStageChange` method to remove event handlers for this event.

OnStageSelected event (Client API reference)

Article • 12/16/2022

This event occurs when a stage of a business process flow control is selected. You can't cancel the stage selection using code in a handler for this event.

Note

The chevron arrow buttons in the business process flow bar can be used to move through and view all the stages. However, these forward and backward chevron buttons do not select the stage and do not change the active stage. Therefore, OnStageSelected event does not occur while navigating and viewing stages when using the chevron arrow buttons.

You can use the [getEventArgs](#) method to retrieve an object that has the following method:

getStage: Returns a stage object representing the selected stage. More information: [Stage methods](#).

Methods supported for this event

- [formContext.data.process.addOnStageSelected](#) method to add event handlers for this event.
- [formContext.data.process.removeOnStageSelected](#) method to remove event handlers for this event.

PostSave Event

Article • 12/16/2022

PostSave event occurs after the `OnSave` event is complete. This event is used to support or execute custom logic using web resources to perform after `Save` actions when the `save` event is successful or failed due to server errors.

ⓘ Note

The `OnSave` event for appointment, recurring appointment, or service activity records will cancel the save operation and use the `Book` message to persist the change rather than `Create` or `Update`. Because of this, `OnSave` and `PostSave` event handlers for these tables will not work.

Use the `addOnPostSave` and `removeOnPostSave` methods to manage event handlers for this event.

ⓘ Note

This method is supported only on Unified Interface

Syntax

```
formContext.data.entity.addOnPostSave(myFunction)
```

Parameter

Name	Type	Required	Description
myFunction	function reference	Yes	The function to add to the PostSave event. The <code>execution context</code> is automatically passed as the first parameter to this function.

Example

The following sample code displays organization unique name as form notification.

JavaScript

```
function addMessageToOnPostSave(executionContext) {
    var formContext = executionContext.getFormContext();
    formContext.data.entity.addOnPostSave(displayOrgName);
}

// function to display organization unique name.

function displayOrgName(executionContext)
{
    var formContext = executionContext.getFormContext();
    var orgName =
Xrm.Utility.getGlobalContext().organizationSettings.uniqueName;
    var myuniqueId = "_myUniqueId";
    formContext.ui.setFormNotification(orgName, "INFO", myuniqueId);
    window.setTimeout(function () {
formContext.ui.clearFormNotification(myUniqueId); }, 10000);

}
```

Relates articles

[getEntityReference](#)
[getIsSaveSuccess](#)
[getSaveErrorInfo](#)

PostSearch event (Client API reference)

Article • 12/16/2022

This event occurs when the search is complete in a knowledge base search control, and the results are displayed.

Use the [addOnPostSearch](#) and [removeOnPostSearch](#) methods to manage event handlers for this event.

Lookup Control PreSearch event (Client API reference)

Article • 12/16/2022

This event occurs just before the lookup control launches a dialog to search for records. There is no UI to set event handlers for this event. You must use the [addPreSearch](#) and [removePreSearch](#) methods on the lookup control to add or remove event handlers for this event.

Use this event with other lookup control methods to change the results displayed in a lookup based on the form data just before the lookup control shows search results for a user to choose from.

Related topics

[addCustomFilter](#)

Subgrid OnLoad event (Client API reference)

Article • 12/16/2022

This event occurs every time the subgrid refreshes. This includes when users sort values in subgrid by selecting the column headings.

Use the GridControl.addOnLoad and GridControl.removeOnLoad methods to manage event handlers, usually in the form Onload event.

TabStateChange event (Client API reference)

Article • 12/16/2022

This event occurs when the **DisplayState** of the tab changes due to user interaction or when the [setDisplayState](#) method is applied in code.

Use this event when you want to change the **src** property of an IFRAME within the tab. If you set the **IFrame.src** property in the **OnLoad** event for an IFRAME within a collapsed tab, the value will be overwritten when the tab is expanded.

Use the [addTabStateChange](#) method to add event handlers for this event and the [removeTabStateChange](#) method to remove them.

Collections (Client API reference)

Article • 12/16/2022

Collections are structures to provide access to data that represent an array, but without the ability to modify the data in the array. More information: [Collections in formContext object model](#)

These methods are available to all the collections in the `formContext` object model.

Method	Description
forEach	Applies the action contained in a delegate function.
get	Get one or more objects from the collection depending on the arguments passed.
getLength	Gets the count of items in the collection.

Related topics

[Client API form context](#)

forEach method for collections (Client API reference)

Article • 12/16/2022

Applies the action contained in a delegate function.

Syntax

```
collection.forEach(delegate function(attribute, index))
```

Parameters

Delegate function with parameters for column and index. |

Related topics

[Collections in Client API](#)

[get](#)

[getLength](#)

get method for collections (Client API reference)

Article • 12/16/2022

Get one or more objects from the collection depending on the arguments passed.

Syntax

```
collection.get([String][Number][delegate function(attribute, index)])
```

Parameters

Parameter	Return value	Return type
None	All the objects in the collection	Array
String	The object where the name matches the argument The objects returned in the <code>formContext.data.process</code> namespace don't contain names. So, using the string parameter for this method returns no objects.	Object
Number	The object where the index matches the number	Object
delegate function(attribute, index)	Any objects that cause the delegate function to return true.	Array

Related topics

[Collections in Client API](#)

[forEach](#)

[getLength](#)

getLength method for collections (Client API reference)

Article • 12/16/2022

Gets the count of items in the collection.

Syntax

```
collection.getLength()
```

Return value

Type: Number

Description: Count of items in the collection.

Related topics

[Collections in Client API](#)

[forEach](#)

[get](#)

GetGlobalContext function and ClientGlobalContext.js.aspx (Client API reference)

Article • 11/30/2022

Use the **GetGlobalContext** function when programming with [web resources](#) to gain access to the global context information such as the information specific to the client, organization or user for your model-driven apps instance.

To get access to the **GetGlobalContext** function in your HTML web resource, include a reference to [ClientGlobalContext.js.aspx](#).

ⓘ Important

Including a reference to [ClientGlobalContext.js.aspx](#) does not make the `Xrm` object available in HTML web resources. Therefore, scripts containing `xrm.*` methods aren't supported in HTML web resources. `parent.Xrm.*` will work if the HTML web resource is loaded in a form container. However, for other places, such as loading an HTML web resource as part of the SiteMap, `parent.Xrm.*` also won't work.

GetGlobalContext function

The **GetGlobalContext** function returns the same context object as returned by the `Xrm.Utility.getGlobalContext` method, which implies that the context object will have the same properties and methods as available for `Xrm.Utility.getGlobalContext`. More information: [Xrm.Utility.getGlobalContext](#)

ClientGlobalContext.js.aspx

You must include a reference to the [ClientGlobalContext.js.aspx](#) page located at the root of the web resources directory to be able to use the **GetGlobalContext** function.

- If you are not using slash characters in HTML web resource names to simulate a folder structure, you can include this script by directly referring to it. For example:

HTML

```
<head>
  <title>HTML Web Resource</title>
  <script src="ClientGlobalContext.js.aspx" type="text/javascript" >
</script>

</head>
```

- If you are using backslash characters in HTML web resource names to simulate a directory structure, you must reflect this in your script element. The following example is for an HTML web resource named **sdk/_Contoso.htm** and a JavaScript web resource named **sdk/_Scripts/ContosoScript.js** with a CSS web resource named **sdk/_Styles/ContosoStyles.css**.

HTML

```
<head>
  <title>HTML Web Resource</title>
  <script src="..../ClientGlobalContext.js.aspx" type="text/javascript" >
</script>

  <script src="Scripts/ContosoScript.js" type="text/javascript">
</script>
  <link href="Styles/ContosoStyles.css" rel="stylesheet"
type="text/css" />
</head>
```

① Note

Using a relative path including the root WebResources folder, for example, `/WebResources/ClientGlobalContext.js.aspx`, is not recommended because it can cause the page to lose organization context in a multi-tenant environment.

The `ClientGlobalContext.js.aspx` page will include some global event handlers. These event handlers will cancel the `onselectstart` , `contextmenu` , and `ondragstart` events.

Related topics

[Xrm.Utility.getGlobalContext](#)

[Understand Client API object model](#)

[Web resources for model-driven apps](#)

Execution context (Client API reference)

Article • 11/30/2022

The execution context defines the event context in which your code executes. More information: [Client API execution context](#).

The execution context object provides the following methods.

Method	Description
getDepth	Returns a value that indicates the order in which this handler is executed.
getEventArgs	Returns an object with methods to manage this handler.
getEventSource	Returns a reference to the object that the event occurred on.
getFormContext	Returns a reference to the form or an item on the form depending on where the method was called.
getSharedVariable	Retrieves a variable set using the setSharedVariable method.
setSharedVariable	Sets the value of a variable to be used by a handler after the current handler completes.

Related topics

[Client API execution context](#)

[Save event arguments](#)

[Understand Client API object model](#)

getDepth (Client API reference)

Article • 12/16/2022

Returns a value that indicates the order in which this handler is executed.

Syntax

```
ExecutionContextObj.getDepth()
```

Return value

Type: Number

Description: The order in which this handler is executed. The order begins with 0.

Related topics

[Execution context](#)

getEventArgs (Client API reference)

Article • 01/14/2023

Returns an object with methods to manage the events.

Syntax

```
ExecutionContextObj.getEventArgs()
```

Return value

Type: Object

Description: When a specified event occurs, you can use the `getEventArgs` method of the execution context object to retrieve an object that may contain additional methods you can use. The table below describes the methods:

Events	Return Object Methods
OnChange	None
OnDataLoad	<code>getDataLoadState</code> : Gets the state of the data load. It returns an enum with the following values: - <code>InitialLoad</code> = 1 - <code>Save</code> = 2 - <code>Refresh</code> = 3
OnGridDataLoad	None
OnLoad	<code>getDataLoadState</code> : Gets the state of the data load. It returns an enum with the following values: - <code>InitialLoad</code> = 1 - <code>Save</code> = 2 - <code>Refresh</code> = 3
OnLookupTagClick	- <code>getTagValue</code> : Gets the selected tag value. The value returned for the <code>getTagValue</code> method is a <code>LookupValue</code> . - <code>preventDefault</code> : Cancels the save operation, but all remaining handlers for the event will still be executed. - <code>isDefaultPrevented</code> : Returns a value indicating whether the save event has been canceled because the <code>preventDefault</code> method was used in this event handler or a previous event handler.
OnPostSearch	None

Events	Return Object Methods
OnPostSave Event	<ul style="list-style-type: none"> - getEntityReference: Returns a lookup value that references the record. - getIsSaveSuccess: Returns data about whether the save operation succeeded. - getSaveErrorInfo: If the save operation failed, returns data about why it failed.
OnProcessStatusChange	<ul style="list-style-type: none"> - getStatus: Returns the Business Process Flow status: <code>Active</code>, <code>Finished</code>, or <code>Aborted</code>.
OnPreProcessStatusChange	<ul style="list-style-type: none"> - getStatus: Returns the Business Process Flow status: <code>Active</code>, <code>Finished</code>, or <code>Aborted</code>.
OnPreStageChange event	<ul style="list-style-type: none"> - getStage: Gets the stage object corresponding to the event triggered. Returns the selected stage in for the <code>OnStageSelected</code> event and next or previous stage objects for the <code>OnStageChange</code> event depending on direction moved. More information: Stage methods.
OnReadyStateComplete	None
OnRecordSelect	None
OnResultOpened	None
OnSave	<ul style="list-style-type: none"> - getSaveMode: Returns a value indicating how the save event was initiated by the user. - preventDefault: Cancels the save operation, but all remaining handlers for the event will still be executed. - isDefaultPrevented: Returns a value indicating whether the save event has been canceled because the <code>preventDefault</code> method was used in this event handler or a previous event handler. <p>If the event is async and async handlers are enabled, the following are also available:</p> <ul style="list-style-type: none"> - preventDefaultOnError: Cancels the save operation if the event handler has a script error. - disableAsyncTimeout: Disables the default async handler timeout.
OnSelection	None
OnStageChange	<ul style="list-style-type: none"> - getStage: Gets the stage object corresponding to the event triggered. Returns the selected stage in for the <code>OnStageSelected</code> event and next or previous stage objects for the <code>OnStageChange</code> event depending on direction moved. More information: Stage methods. - getDirection: Gets the direction of the stage advance action. It returns a string value <code>Next</code> or <code>Previous</code>.

Events	Return Object Methods
OnStageSelected	<ul style="list-style-type: none"> - getStage: Gets the stage object corresponding to the event triggered. Returns the selected stage in for the <code>OnStageSelected</code> event and next or previous stage objects for the <code>OnStageChange</code> event depending on direction moved. More information: Stage methods. - getDirection: Gets the direction of the stage advance action. It returns a string value <code>Next</code> or <code>Previous</code>.
OnTabStateChange	None
PreSearch	None

Related topics

[Execution context](#)

getEventSource (Client API reference)

Article • 01/11/2023

Returns a reference to the object that the event occurred on.

Syntax

```
ExecutionContextObj.getEventSource()
```

Return value

Type: Object

Description: Returns the object from the **Xrm** object model that is the source of the event, not an HTML DOM object. For example, in an [OnChange](#) event, this method returns the `formContext.data.entity` object that represents the changed column.

Events	Return Object
OnChange	column
OnDataLoad	<code>formContext.data.entity</code>
OnGridDataLoad	SubGrid control
OnLoad	<code>formContext.ui</code>
OnLookupTagClick	Lookup control
OnPostSearch	<code>kbSearch</code> control
OnProcessStatusChange	<code>formContext.data.process</code>
OnProcessStatusChange	<code>formContext.data.process</code>
OnReadyStateComplete	IFrame control
OnRecordSelect	<code>formContext.data.entity</code>
OnResultOpened	<code>kbSearch</code> control
OnSave	<code>formContext.data.entity</code>
OnPostSave	None
OnSelection	None

Events	Return Object
OnStageChange	formContext.data.process
OnStageSelected	formContext.data.process
OnTabStateChange	tab object
PreSearch	Lookup control

Related topics

[Execution context](#)

getFormContext (Client API reference)

Article • 12/16/2022

Returns a reference to the form or an item on the form depending on where the method was called.

Syntax

```
ExecutionContextObj.getFormContext()
```

Return value

Type: Object

Description: Returns a reference to the form or an item on the form such as editable grid depending on where the method was called. This method enables you to create common event handlers that can operate either on a form or an item on the form depending on where its called.

Example

The following sample code demonstrates how you can create a method that sets notification on a form column or editable grid cell depending on where you registered the script ([Column OnChange](#) event or editable grid [OnChange](#) event):

JavaScript

```
function commonEventHandler(executionContext) {
    var formContext = executionContext.getFormContext();
    var telephoneAttr =
formContext.data.entity.attributes.get('telephone1');
    var isNumberWithCountryCode = telephoneAttr.getValue().substring(0,1)
==== '+';

    // telephoneField will be a form control if invoked from a form OnChange
    event;
    // telephoneField will be a editable grid GridCell object if invoked
    from editable grid OnChange event.
    var telephoneField = telephoneAttr.controls.get(0);

    if (!isNumberWithCountryCode) {
        telephoneField.setNotification('Please include the country code
beginning with '+'.', 'countryCodeNotification');
    }
}
```

```
    else {
        telephoneField.clearNotification('countryCodeNotification');
    }
}
```

Related topics

[Execution context](#)

[Form context](#)

getSharedVariable (Client API reference)

Article • 11/30/2022

Retrieves a variable set using the [setSharedVariable](#) method.

Syntax

```
ExecutionContextObj.getSharedVariable(key)
```

Parameters

key

Type: String

Description: The name of the variable.

Return value

Type: Object

Description: The specific type depends on what the value object is.

Related topics

[setSharedVariable](#)

[Execution context](#)

setSharedVariable (Client API reference)

Article • 11/30/2022

Sets the value of a variable to be used by a handler after the current handler completes.

Syntax

```
ExecutionContextObj.setSharedVariable(key, value)
```

Parameters

- **key:** String. The name of the variable
- **Value:** Object. The values to set

Return value

Type: Object

Description: The specific type depends on what the value object is.

Related topics

[getSharedVariable](#)

[Execution context](#)

Save event arguments (Client API reference)

Article • 01/14/2023

When the form [OnSave](#) event occurs, you can use the [getEventArgs](#) method of the execution context object to retrieve an object that contains methods you can use to manage the save event.

Method	Description
getSaveMode	Returns a value indicating how the save event was initiated by the user.
isDefaultPrevented	Returns a value indicating whether the save event has been canceled because the preventDefault method was used in this event handler or a previous event handler.
preventDefault	Cancels the save operation, but all remaining handlers for the event will still be executed.
preventDefaultOnError	Cancels the save operation if the event handler has a script error.
disableAsyncTimeout	Disables the default async handler timeout.

PostSave event arguments

When the form [OnPostSave](#) event occurs, you can use the [getEventArgs](#) method of the execution context object to retrieve an object that contains methods you can use to manage the postsave event.

Method	Description
getEntityReference	Use this method to know information about a table being saved. It returns the table logical name, record id, and table name if save was successful.
getIsSaveSuccess	Use this method to know whether the save operation was successful or failed.
getSaveErrorInfo	Use this method to know the error details on why save failed.

Related topics

[Client API execution context](#)

Execution context methods

EventArgs.getEntityReference (Client API reference)

Article • 08/04/2023

Use this method to know information about a table being saved/updated. It returns table ID, and table name if success.

ⓘ Note

This API is not available for Dynamics 365 Customer Engagement on-premise deployments. More information: [Client APIs not supported in Customer Engagement \(on-premises\)](#)

Syntax

```
executionContext.getEventArgs().getEntityReference();
```

Related articles

[PostSave](#)

getIsSaveSuccess (Client API reference)

Article • 08/04/2023

Use this method to know whether the `OnSave` operation is successful or failed.

ⓘ Note

This API is not available for Dynamics 365 Customer Engagement on-premise deployments. More information: [Client APIs not supported in Customer Engagement \(on-premises\)](#)

Syntax

```
executionContext.getEventArgs().getIsSaveSuccess();
```

Related articles

[PostSave](#)

getSaveMode (Client API reference)

Article • 12/16/2022

Returns a value indicating how the save event was initiated by the user.

Syntax

```
executionContext.getEventArgs().getSaveMode()
```

Return Value

Type: Number

Description: The following table describes the supported values returned to detect different ways table records may be saved by the user.

Value	Save mode	Table
1	Save	All
2	Save and Close	All
5	Deactivate	All
6	Reactivate	All
7	Send	Email
15	Disqualify	Lead
16	Qualify	Lead
47	Assign	User or Team owned tables
58	Save as Completed	Activities
59	Save and New	All
70	Auto Save	All

Remarks

This method is essential if you want to enable auto-save for most forms in an organization but disable it for specific forms.

Example

The following code registered for the **OnSave** event with the execution context passed to it will prevent any saves that initiate from an auto-save but allow all others. With auto-save enabled, navigating away is equivalent to **Save and Close**. This code will prevent any saves that are initiated by the 30 second timer or when people navigate away from a form with unsaved data.

JavaScript

```
function preventAutoSave(executionContext) {  
    var eventArgs = executionContext.getEventArgs();  
    if (eventArgs.getSaveMode() == 70 || eventArgs.getSaveMode() == 2) {  
        eventArgs.preventDefault();  
    }  
}
```

To save a record the user must select the **Save** icon at the bottom of the form or a custom **Save** command needs to be added to the command bar.

Related topics

[isDefaultPrevented](#)

[preventDefault](#)

getSaveErrorInfo (Client API reference)

Article • 08/04/2023

Use this method to know the error details on why a table save failed.

ⓘ Note

This API is not available for Dynamics 365 Customer Engagement on-premise deployments. More information: [Client APIs not supported in Customer Engagement \(on-premises\)](#)

Syntax

```
executionContext.getEventArgs().getSaveErrorInfo();
```

Related articles

[PostSave](#)

isDefaultPrevented (Client API reference)

Article • 12/16/2022

Returns a value indicating whether the save event has been canceled because the [preventDefault](#) method was used in this event handler or a previous event handler.

Syntax

```
executionContext.getEventArgs().isDefaultPrevented();
```

Return Value

Type: Boolean

Description: **true** if the save event has been canceled because the [preventDefault](#) method was used; **false** otherwise.

Related topics

[getSaveMode](#)

[preventDefault](#)

preventDefault (Client API reference)

Article • 12/16/2022

Cancels the save operation, but all remaining handlers for the event will still be executed.

Syntax

```
executionContext.getEventArgs().preventDefault();
```

Important

When you use `preventDefault` on a form with business process flows, the stage navigation may throw this error: **Unable to save form data due to web resource registered onSave invoking preventDefault**. Use [OnPreStageChange](#) to prevent this error.

Related topics

[getSaveMode](#)

[isDefaultPrevented](#)

preventDefaultOnError (Client API reference)

Article • 08/04/2023

Cancels the save operation if the event handler has a script error, returns a rejected promise for an async event handler or the operation times out.

ⓘ Note

This API is not available for Dynamics 365 Customer Engagement on-premise deployments. More information: [Client APIs not supported in Customer Engagement \(on-premises\)](#)

Syntax

```
executionContext.getEventArgs().preventDefaultOnError();
```

Related topics

[getSaveMode](#)

[isDefaultPrevented](#)

Columns (Client API reference)

Article • 01/10/2023

Columns contain data in the model-driven apps form or grids. Use the `formContext.data.entity.attributes` collection or the `formContext.getAttribute` shortcut method to access a collection of columns. For more information about collections, see [Collections \(Client API reference\)](#).

To access a column within the collection, you pass either the name (string) or the index value (number) of the column as an argument to the method. For example: `formContext.getAttribute(arg)`. If no argument is specified, it will return a collection of all attributes on the form.

Columns are categorized by type. You can determine the type of the column by using the `getAttributeType` method. Certain column methods are only available for specific types of columns.

This topic provides information about the methods available per column type.

All column types

- | | | |
|---|--|---|
| <ul style="list-style-type: none">• <code>controls</code>• <code>addOnChange</code>• <code>fireOnChange</code>• <code>getAttributeType</code>• <code>getFormat</code>• <code>getIsDirty</code> | <ul style="list-style-type: none">• <code>getName</code>• <code>getParent</code>• <code>getRequiredLevel</code>• <code>getSubmitMode</code>• <code>getUserPrivilege</code>• <code>getValue</code> | <ul style="list-style-type: none">• <code>isValid</code>• <code>removeOnChange</code>• <code>setRequiredLevel</code>• <code>setSubmitMode</code>• <code>setValue</code>• <code>setIsValid</code> |
|---|--|---|

Boolean column type

In addition to the methods available for all column types as explained earlier, the following method is available only for the `boolean` column:

- `getInitialValue`

Lookup column type

In addition to the methods available for all column types as explained earlier, the following method is available only for the `lookup` column:

- [getIsPartyList](#)

Choices and choice column types

In addition to the methods available for all column types as explained earlier, the following methods are available only for the **choices** and **choice** columns:

- [getInitialValue](#)
- [getOption](#)
- [getOptions](#)
- [getSelectedOption](#)
- [getText](#)

Number column type (decimal, double, integer, money)

The following methods are available only for the **decimal**, **double**, and **integer** columns:

- [getMax](#)
- [getMin](#)
- [getPrecision](#)
- [setPrecision](#)

String column type

In addition to the methods available for all column types as explained earlier, the following method is available only for the **string** column:

- [getMaxLength](#)

Related topics

[Understand Xrm object model](#)

[Controls \(Client API reference\)](#)

Controls collection (Client API reference)

Article • 12/16/2022

Use the Controls collection to access controls associated with columns.

Because each column may be represented more than one time on the page, the controls collection provides access to all controls representing that column. If the column is represented by only one control in the page, the length of this collection will be 1. When you use the control getName method, the name of the first control will be the same as the name of the column. The second instance of a control for that column will be <columnName>1. The pattern <columnName>+N will continue for each additional control added to the form for a specific column.

When a form displays a business process flow control in the header, additional controls will be added for each column that is displayed in the business process flow. These controls have a unique name like the following: **header_process_<column name>**.

When performing actions on controls that are tied to a column, you should always consider that the control may be included on the page more than once and you should generally perform the same actions for each control for the column. You can do this by looping through the column controls collection and perform the actions on each control.

Related topics

[Columns \(Client API reference\)](#)

addOnChange (Client API reference)

Article • 11/30/2022

Sets a function to be called when the **OnChange** event occurs.

ⓘ Note

If the `addOnChange` method is used on the form `onLoad` event handler, you should ensure that it's called when necessary. You can use the `getEventArgs` to conditionally call the `addOnChange` method based on the data load state.

Column types supported

All

Syntax

```
formContext.getAttribute(arg).addOnChange(myFunction)
```

Parameters

Parameter	Type	Description
Name		
myFunction	Function reference	Specifies the function to be executed on the column OnChange event. The execution context is automatically passed as the first parameter to this function.

Related topics

[removeOnChange](#)

[Column OnChange Event](#)

fireOnChange (Client API reference)

Article • 11/30/2022

Causes the `OnChange` event to occur on the column so that any script associated to that event can execute.

Column types supported

All

Syntax

```
formContext.getAttribute(arg).fireOnChange()
```

Related topics

[Column OnChange Event](#)

getAttributeType (Client API reference)

Article • 01/10/2023

Returns a string value that represents the type of column.

Column types supported

All

Syntax

```
formContext.getAttribute(arg).getAttributeType()
```

Return Value

This method will return one of the following **string** values:

- boolean
- datetime
- decimal
- double
- file
- image
- integer
- lookup
- memo
- money
- multiselectoptionset (choices)
- optionset (choice)
- string

getFormat (Client API reference)

Article • 11/30/2022

Returns a string value that represents formatting options for the column.

Column types supported

All

Syntax

```
formContext.getAttribute(arg).getFormat()
```

Return Value

This method will return one of the following **string** values or "null":

- date
- datetime
- duration
- email
- language
- none
- phone
- text
- textarea
- tickersymbol
- timezone
- url

Note

This format information generally represents the format options of the application column. Format options for Boolean columns are not provided.

The following table lists the format string values to expect for each type of column type and format option.

Application column Type	Format Option	Column Type	Format Value
Date and Time	Date Only	datetime	date
Date and Time	Date and Time	datetime	datetime
Whole Number	Duration	integer	duration
Single Line of Text	E-mail	string	email
Whole Number	Language	choice	language
Whole Number	None	integer	none
Single Line of Text	Text Area	string	textarea
Single Line of Text	Text	string	text
Single Line of Text	Ticker Symbol	string	tickersymbol
Single Line of Text	Phone	string	phone
Whole Number	Time Zone	choice	timezone
Single Line of Text	Url	string	url

getInitialValue (Client API reference)

Article • 11/30/2022

Returns a value that represents the value set for a Yes/No, Choice or Choices column when the form is opened.

Column types supported

Yes/No, Choice, Choices

Syntax

```
formContext.getAttribute(arg).getInitialValue()
```

Return Value

Type: Number

Description: The initial value for the column.

attribute.getIsDirty (Client API reference)

Article • 11/30/2022

Returns a boolean value indicating if there are unsaved changes to the column value. An unsaved change to a column value means the client value is different from the last known committed value retrieved from Dataverse by the client from runtime.

Column types supported

All

Syntax

```
formContext.getAttribute(arg).getIsDirty()
```

Return Value

Type: Boolean.

Description: True if there are unsaved changes, otherwise false.

getIsPartyList (Client API reference)

Article • 11/30/2022

Returns a Boolean value indicating whether the lookup represents a partylist lookup. Partylist lookups allow for multiple records to be set, such as the **To:** column for an email table record.

Column types supported

Lookup

Syntax

```
formContext.getAttribute(arg).getIsPartyList()
```

Return Value

Type: Boolean.

Description: True if the lookup column is a partylist, otherwise false.

getMax (Client API reference)

Article • 11/30/2022

Returns a number indicating the maximum allowed value for a column.

Column types supported

decimal, integer, double, money

Syntax

```
formContext.getAttribute(arg).getMax()
```

Return Value

Type: Number.

Description: The maximum allowed value for the column.

getMaxLength (Client API reference)

Article • 11/30/2022

Returns a number indicating the maximum length of a string or memo column.

Column types supported

string, memo

Syntax

```
formContext.getAttribute(arg).getMaxLength()
```

Return Value

Type: Number.

Description: The maximum allowed length of a string for this column.

ⓘ Note

The email form description column is a memo column, but it does not have a `getMaxLength` method.

getMin (Client API reference)

Article • 11/30/2022

Returns a number indicating the minimum allowed value for a column.

Column types supported

Decimal, integer, double, money

Syntax

```
formContext.getAttribute(arg).getMin()
```

Return Value

Type: Number.

Description: The minimum allowed value for the column.

attribute.getName (Client API reference)

Article • 11/30/2022

Returns a string representing the logical name of the column.

Column types supported

All

Syntax

```
formContext.getAttribute(arg).getName()
```

Return Value

Type: String.

Description: The logical name of the column.

Related topics

[setSubmitMode \(Client API reference\)](#)

getOption (Client API reference)

Article • 11/30/2022

Returns an option object with the value matching the argument (label or enumeration value) passed to the method.

Column types supported

Choice, Choices

Syntax

```
formContext.getAttribute(arg).getOption(value)
```

Parameters

String (label of the option) or **Number** (enumeration value of the option).

Return Value

Type: Option object.

Description: The logical name of the column.

attribute.getOptions (Client API reference)

Article • 05/08/2023

Returns an array of option objects representing valid options for a column.

Column types supported

Choice, Choices

Syntax

```
formContext.getAttribute(arg).getOptions()
```

Return Value

Type: Array of option objects.

Description: The array of option objects representing valid options.

Options have two properties:

Property	Type	Description
text	string	The localized label for the option.
value	number	The integer value of the option.

attribute.getParent (Client API reference)

Article • 11/30/2022

Returns the `formContext.data.entity` object that is the parent to all the columns.

Column types supported

All

Syntax

```
formContext.getAttribute(arg).getParent()
```

Return Value

Type: `formContext.data.entity` object.

Description: The parent object.

getPrecision (Client API reference)

Article • 11/30/2022

Returns the number of digits allowed to the right of the decimal point.

Column types supported

Money, decimal, double, and integer

Syntax

```
formContext.getAttribute(arg).getPrecision()
```

Return Value

Type: Number.

Description: The number of digits allowed to the right of the decimal point.

Related topics

[setPrecision](#)

getRequiredLevel (Client API reference)

Article • 11/30/2022

Returns a string value indicating whether a value for the column is required or recommended.

Column types supported

All

Syntax

```
formContext.getAttribute(arg).getRequiredLevel()
```

Return Value

Type: String.

Description: Returns one of the following values:

- none
- required
- recommended

Related topic

[setRequiredLevel \(Client API reference\)](#)

getSelectedOption (Client API reference)

Article • 11/30/2022

Returns the option object or an array of option objects selected in a **choice** or **choices** column respectively.

Column types supported

choice, choices

Syntax

```
formContext.getAttribute(arg).getSelectedOption()
```

Return Value

Type: Option object for choice; array of option objects for choices.

Description: Returns the object with text and value properties.

Related topics

[getInitialValue \(Client API reference\)](#)

[getOption \(Client API reference\)](#)

[getOptions \(Client API reference\)](#)

[getText \(Client API reference\)](#)

getSubmitMode (Client API reference)

Article • 11/30/2022

Returns a string indicating when data from the column will be submitted when the record is saved.

Column types supported

All

Syntax

```
formContext.getAttribute(arg).getSubmitMode()
```

Return Value

Type: String.

Description: Returns one of the following values:

- always
- never
- dirty

Related topic

[setSubmitMode \(Client API reference\)](#)

getText (Client API reference)

Article • 11/30/2022

Returns a string value of the text for the currently selected option for a **choice** or **choices** column.

Column types supported

choice, choices

Syntax

```
formContext.getAttribute(arg).getText()
```

Return Value

Type: String.

Description: The **text** value of the selected option.

ⓘ Note

When no option is selected, it will return null.

Related topics

[getInitialValue \(Client API reference\)](#)

[getOption \(Client API reference\)](#)

[getOptions \(Client API reference\)](#)

[getSelectedOption \(Client API reference\)](#)

getUserPrivilege (Client API reference)

Article • 11/30/2022

Returns an object with three boolean properties corresponding to privileges indicating if the user can create, read or update data values for a column. This function is intended for use when Field Level Security modifies a user's privileges for a particular column.

Column types supported

All

Syntax

```
formContext.getAttribute(arg).getUserPrivilege()
```

Return Value

Type: Object.

Description: The object has three boolean properties:

- canRead
- canUpdate
- canCreate

attribute.getValue (Client API reference)

Article • 11/30/2022

Retrieves the data value for a column.

Column types supported

All

Syntax

```
formContext.getAttribute(arg).getValue()
```

Return Value

Type: Depends on the type of column. The value may be null.

Column Type	Return Type
boolean	Boolean
datetime	Date To get the string version of a date using the Power Apps user's locale preferences, use the format and localeFormat methods. Other methods will format dates using the operating system locale rather than the user's Power Apps locale preferences.
decimal	Number
Double	Number
integer	Number

Column	Return Type
Type	
lookup	Array ↗ An array of lookup objects. NOTE: Certain lookups allow for multiple records to be associated in a lookup, such as the To: column for an email table record. Therefore, all lookup data values use an array of lookup objects – even when the lookup column does not support more than one record reference to be added. Each lookup has the following properties: - <i>entityType</i> : String. The name of the table displayed in the lookup. - <i>id</i> : String: The string representation of the GUID value for the record displayed in the lookup. - <i>name</i> : String: The text representing the record to be displayed in the lookup.
memo	String ↗
money	Number ↗
choices	Array ↗ An array of numbers.
choice	Number ↗
string	String ↗

Related topic

[setValue \(Client API reference\)](#)

attribute.isValid (Client API reference)

Article • 11/30/2022

Returns a boolean value to indicate whether the value of a column is valid.

Column types supported

All

Syntax

```
formContext.getAttribute(arg).isValid();
```

Return Value

Type: Boolean.

Description: true if the column value is valid; false otherwise.

removeOnChange (Client API reference)

Article • 11/30/2022

Removes a function from the **OnChange** event handler for a column.

Column types supported

All

Syntax

```
formContext.getAttribute(arg).removeOnChange(myFunction)
```

Parameters

Parameter	Type	Description
Name		
myFunction	Function reference	Specifies the function to be removed from the OnChange event.

Related topics

[addOnChange](#)

[Column OnChange Event](#)

setIsValid (Client API reference)

Article • 11/30/2022

Sets a value for a column to determine whether it is valid or invalid with a message.

Column types supported

All

Syntax

```
formContext.getAttribute(arg).setIsValid(bool, message);
```

Parameters

Name	Type	Required	Description
bool	Boolean	Yes	Specify false to set the column value to invalid and true to set the value to valid.
message	String	No	The message to display.

setPrecision (Client API reference)

Article • 11/30/2022

Sets the number of digits allowed to the right of the decimal point.

Column types supported

Money, decimal, double, and integer

Syntax

```
formContext.getAttribute(arg).setPrecision(value);
```

Parameter

Parameter Name	Type	Description
value	Number	Number of digits allowed to the right of the decimal point.

Related topics

[getPrecision](#)

setRequiredLevel (Client API reference)

Article • 11/30/2022

Sets whether data is required or recommended for the column before the record can be saved.

ⓘ Important

Reducing the required level of a column can cause an error when the page is saved. If the column is required by the server, an error will occur if there is no value for the column.

Column types supported

All

Syntax

```
formContext.getAttribute(arg).setRequiredLevel(requirementLevel)
```

Parameters

Type: String.

Description: Set the level to one of the following values:

- none
- required
- recommended

Related topic

[getRequiredLevel \(Client API reference\)](#)

setSubmitMode (Client API reference)

Article • 11/30/2022

Sets whether data from the column will be submitted when the record is saved.

Column types supported

All

Syntax

```
formContext.getAttribute(arg).setSubmitMode(mode)
```

Parameters

Type: String.

Description: Set one of the following mode values:

- **always**: The data is always sent with a save.
- **never**: The data is never sent with a save. When this value is used, the column(s) in the form for this column cannot be edited.
- **dirty**: Default behavior. The data is sent with the save when it has changed.

Remarks

Use this method to control when data for a column is submitted when a record is created or saved. For example, you may have a column on the form that is only intended to control logic in the form. You are not interested in capturing the data in it. You might set it so that the data is not saved. Or you may have a Plugin that depends on the value always being included. You may want to set the column so that it will always be included.

Note

Data in a column will always be refreshed after save operation, even if the column's submit mode is set to `never`. For example, if a column's value in the server is null and the column's submit mode is set to `never`, and the column is modified with

some value by the user, after the user saves the form the column's value will be replaced with null.

Columns that do not get updated after the initial save of the record, such as `createdby`, are set so that they will not be submitted on save. To force a column value to be submitted whether it has changed or not, use this method with the `mode` parameter set to "always".

See also

[getSubmitMode \(Client API reference\)](#)

setValue (Client API reference)

Article • 08/22/2023

Sets the data value for a column.

Column types supported

All

Syntax

```
formContext.getAttribute(arg).setValue(value)
```

Parameters

Depends on the type of column.

Column Type	Parameters Type
boolean	Boolean ↗
datetime	Date ↗ In Unified Interface, date values are assumed to be UTC. In the legacy web client, date values are assumed to be in the user's time zone.
decimal	Number ↗
double	Number ↗
Integer	Number ↗
lookup	Array ↗ An array of lookup objects. Certain lookups, known as 'partylist' lookups, allow for multiple records to be associated in a lookup, such as the To: column for an email table record. Therefore, all lookup data values use an array of lookup objects – even when the lookup column doesn't support more than one record reference to be added. Each lookup has the following properties: <ul style="list-style-type: none">- <i>entityType</i>: String. The name of the table displayed in the lookup.- <i>id</i>: String: The string representation of the GUID value for the record displayed in the lookup. The value should match the following format: {XXXXXXXX-XXXX-XXXX-XXXX-XXXX}

Column Type	Parameters Type
	XXXXXXXXXXXXX} - <i>name</i> : String: The text representing the record to be displayed in the lookup.
memo	String ↗
money	Number ↗
choice	Number ↗
string	String ↗
memo	String ↗
money	Number ↗
choice, choices	The getOptions method returns option values as strings. You must use parseInt ↗ to convert them to numbers before you can use those values to set the value of a choice column. Valid statuscode (Status Reason) options depend on the current statecode of the record. The statecode (Status) column can't be set in form scripts. To understand which statecode values are valid, refer to the column definitions. For custom tables, use the table definitions browser. Finally, also consider any custom state transitions that have been applied to the column. More information: Define status reason transitions for the Case or custom tables .
String	String ↗ A String column with the email format requires that the string represents a valid email address.

ⓘ Note

Updating a column using **setValue** will not cause the **OnChange** event handlers to run. If you want the **OnChange** event handlers to run you must use **fireOnChange** in addition to **setValue**.

Related article

[getValue \(Client API reference\)](#)

Controls (Client API reference)

Article • 01/10/2023

A control represents an HTML element present on the form. Some controls are bound to a specific column, whereas others may represent unbound controls such as an IFRAME, web resource, or a subgrid that has been added to the form.

The **control** object provides methods to change the presentation or behavior of a control and identify the corresponding column. You access controls using one of the following collections:

- **formContext.ui.controls**
- **formContext.ui Section.controls**
- **formContext.data.entity Attribute.controls**

The **formContext.getControl** method is a shortcut method to access **formContext.ui.controls.get**.

Controls are categorized by type. You can determine the type of a control by using the **getControlType** method. Certain control methods are only available for specific types of controls.

This article provides information about the methods available per control type.

standard control type

These are the methods available for a Standard control.

- | | | |
|---|--|--|
| <ul style="list-style-type: none">• addNotification• clearNotification• getAttribute• getControlType• getDisabled | <ul style="list-style-type: none">• getLabel• getName• getOutputs• getParent• getVisible | <ul style="list-style-type: none">• setDisabled• setFocus• setLabel• setNotification• setVisible |
|---|--|--|

The following methods for the Standard control are **deprecated** in this release: `addOnKeyPress`, `fireOnKeyPress`, and `removeOnKeyPress`.

iframe control type

These are the methods available for an IFRAME control.

- [getContentWindow](#)
- [getControlType](#)
- [getDisabled](#)
- [getInitialUrl](#)
- [getLabel](#)
- [getName](#)
- [getObject](#)
- [getParent](#)
- [getSrc](#)
- [getVisible](#)
- [setDisabled](#)
- [setFocus](#)
- [setLabel](#)
- [setSrc](#)
- [setVisible](#)

kbsearch (Knowledge base search) control type

These are the methods available for knowledge base search control.

- [addOnPostSearch](#)
- [addOnResultOpened](#)
- [addOnSelection](#)
- [getControlType](#)
- [getDisabled](#)
- [getLabel](#)
- [getName](#)
- [getParent](#)
- [getSearchQuery](#)
- [getSelectedResults](#)
- [getTotalResultCount](#)
- [getVisible](#)
- [openSearchResult](#)
- [removeOnPostSearch](#)
- [removeOnResultOpened](#)
- [removeOnSelection](#)
- [setFocus](#)
- [setLabel](#)
- [setSearchQuery](#)
- [setVisible](#)

Note

When the knowledge base search control is added to the social pane, the name of the control will be "searchwidgetcontrol_notescontrol". This name can't be changed.

lookup control type

These are the methods available for a lookup control.

- [addCustomFilter](#)
- [addCustomView](#)
- [addNotification](#)
- [addOnLookupTagClick](#)
- [addPreSearch](#)
- [clearNotification](#)
- [getAttribute](#)
- [getControlType](#)
- [getDefaultView](#)
- [getDisabled](#)
- [getEntityTypes](#)
- [getLabel](#)
- [getName](#)
- [getParent](#)
- [getVisible](#)
- [removeOnLookupTagClick](#)
- [removePreSearch](#)
- [setDefaultView](#)
- [setDisabled](#)
- [setEntityTypes](#)
- [setFocus](#)
- [setLabel](#)
- [setNotification](#)
- [setVisible](#)

choices and choice control types

Both choices and choice controls have the same set of methods available.

- | | | |
|---|---|--|
| <ul style="list-style-type: none"> • addNotification • addOption • clearNotification • clearOptions • getAttribute • getControlType | <ul style="list-style-type: none"> • getDisabled • getLabel • getName • getOptions • getParent • getVisible | <ul style="list-style-type: none"> • removeOption • setDisabled • setFocus • setLabel • setNotification • setVisible |
|---|---|--|

quickform control type

See [formContext.ui.quickForms](#) for information about methods supported for this control type.

subgrid control type

See [Grids and subgrids](#) for information methods supported for this control type.

timelinewall control type

The timeline control presents the Posts, Activities, and Notes in a unified view. These are the methods available for this control type.

- | | | |
|--|---|--|
| <ul style="list-style-type: none"> • getControlType • getDisabled • getLabel • getName | <ul style="list-style-type: none"> • getParent • getVisible • refresh • setDisabled | <ul style="list-style-type: none"> • setFocus • setLabel • setVisible |
|--|---|--|

timer control type

These are the methods available for the timer control.

- | | | |
|--|--|---|
| <ul style="list-style-type: none"> • getControlType • getDisabled • getLabel • getName | <ul style="list-style-type: none"> • getParent • getState • getVisible • refresh | <ul style="list-style-type: none"> • setDisabled • setFocus • setLabel • setVisible |
|--|--|---|

web resource control type

A web resource control has the same set of methods available as the iframe control. See [iframe control type](#)

The Silverlight web resource has these additional methods:

- [getData](#)
- [setData](#)

💡 Tip

If you want to modify all the controls bound to a column on a form, use the controls collection inside the column type. For example, to add notification to each control bound to the `name` column, you can do the following:

JavaScript

```
const notification = { messages: ['Sample Notification on Name Controls'], notificationLevel: 'RECOMMENDATION', uniqueId: 'my_unique_id' };
formContext.getAttribute("name").controls.forEach(control =>
control.addNotification(notification));
```

Form component control type

A form component control type has the same set of methods available as the `formContext` on a main form. See [Form component behavior > Client API](#)

Related topics

[Columns](#)

addCustomFilter (Client API reference)

Article • 12/16/2022

Adds filters to the results displayed in the lookup. Each filter will be combined with any previously added filters as an `AND` condition.

Control types supported

Lookup

Syntax

```
formContext.getControl(arg).addCustomFilter(filter, entityLogicalName)
```

Parameters

- **filter:** String. The fetchXml filter element to apply. For example:

XML

```
<filter type="and">
    <condition attribute="address1_city" operator="eq" value="Redmond" />
</filter>
```

- **entityLogicalName:** (Optional) String. If this is set, the filter only applies to that table type. Otherwise, it applies to all types of tables returned.

Remarks

This method can only be used in a function in an event handler for the [Lookup Control PreSearch Event](#).

Example

The following code sample is for the Opportunity form **Account** (`parentaccountid`) lookup. When the `Sdk.setParentAccountIdFilter` function is set in the form **Onload** event handler, the `Sdk.filterCustomAccounts` function is added to the **PreSearch** event for that lookup. Remember to select the option to pass in the execution context when setting the function in the form **Onload** event handler. The result is that only accounts

with the **Category** (accountcategorycode) value of **Preferred Customer** (1) will be returned.

JavaScript

```
// A namespace defined for SDK sample code
// You should define a unique namespace for your libraries
var Sdk = window.Sdk || {};

// set 'Sdk.setParentAccountIdFilter' in the Opportunity form onload event
// handler
Sdk.setParentAccountIdFilter = function (executionContext) {

    // get the form context
    formContext = executionContext.getFormContext();

    formContext.getControl("parentaccountid").addPreSearch(Sdk.filterCustomerAcc
ounts);
}

Sdk.filterCustomerAccounts = function () {

    // Only show accounts with the type 'Preferred Customer'
    var customerAccountFilter = "<filter type='and'><condition
attribute='accountcategorycode' operator='eq' value='1' /></filter>";

    formContext.getControl("parentaccountid").addCustomFilter(customerAccountFil
ter, "account");
}
```

[addPreSearch](#)

[formContext](#)

addCustomView (Client API reference)

Article • 12/16/2022

Adds a new view for the lookup dialog box.

Control types supported

Lookup

Syntax

```
formContext.getControl(arg).addCustomView(viewId, entityName, viewDisplayName,  
fetchXml, layoutXml, isDefault)
```

Parameters

- **viewId**: String. The string representation of a GUID for a view.

(!) Note

This value is never saved and only needs to be unique among the other available views for the lookup. A string for a non-valid GUID will work, for example "00000000-0000-0000-0000-000000000001". It's recommended that you use a tool like **guidgen.exe** to generate a valid GUID.

- **entityName**: String. The name of the table.
- **viewDisplayName**: String. The name of the view.
- **fetchXml**: String. The fetchXml query for the view.
- **layoutXml**: String. The XML that defines the layout of the view.
- **isDefault**: Boolean: Indicates whether the view should be the default view.

Remarks

This method doesn't work with **Owner** lookups. Owner lookups are used to assign user-owned records.

addNotification (Client API reference)

Article • 12/16/2022

Displays an error or recommendation notification for a control, and lets you specify actions to execute based on the notification. When you specify an error type of notification, a red "X" icon appears next to the control. When you specify a recommendation type of notification, an "i" icon appears next to the control. On Dynamics 365 mobile clients, tapping on the icon will display the message, and let you perform the configured action by clicking the **Apply** button or dismiss the message.

Control types supported

All

Syntax

```
formContext.getControl(arg).addNotification(notification);
```

Parameters

Name	Type	Required	Description
notification	Object	Yes	<p>The notification to add. The object contains the following parameters:</p> <ul style="list-style-type: none">• actions: (Optional) Array of objects. A collection of objects with the following parameters:<ul style="list-style-type: none">◦ message: (Optional) String. The body message of the notification to be displayed to the user. Limit your message to 100 characters for optimal user experience.◦ actions: (Optional) Array of functions. The corresponding actions for the message.• messages: Array of Strings. The message to display in the notification. In the current release, only the first message specified in this array will be displayed. The string that you specify here appears as bold text in the notification, and is typically used for title or subject of the notification. You should limit your message to 50 characters for optimal user experience.• notificationLevel: String. Defines the type of notification. Valid values are ERROR or RECOMMENDATION.

- **uniqueId**: String. The ID to use to clear this notification when using the **clearNotification** method.

Return Value

Type: Boolean

Description: Indicates whether the method succeeded.

Remarks

In web client the **addNotification** method displays a notification with the messages you specified and two standard buttons: **Apply** and **Dismiss**. Clicking **Apply** executes the action you define; clicking **Dismiss** closes the notification message.

In Unified Interface:

- There is no **Dismiss** button.
- The **Apply** button only appears when the notification level is set to **RECOMMENDATION**, not **ERROR**.

Example

The following sample code displays a notification on the **Account Name** column of the account form to set the **Ticker Symbol** if the **Account Name** column contains "Microsoft", and the ticker symbol is not already set to "MSFT". Clicking **Apply** in the notification will set the **Ticker Symbol** column to "MSFT".

JavaScript

```
function addTickerSymbolRecommendation(executionContext) {  
    var formContext = executionContext.getFormContext();  
    var myControl = formContext.getControl('name');  
    var accountName = formContext.data.entity.attributes.get('name');  
    var tickerSymbol =  
        formContext.data.entity.attributes.get('tickersymbol');  
  
    if (accountName.getValue() == 'Microsoft' && tickerSymbol.getValue() !=  
        'MSFT') {  
        var actionCollection = {  
            message: 'Set the Ticker Symbol to MSFT?',  
            actions: null  
        };  
    };
```

```

actionCollection.actions = [function () {
    tickerSymbol.setValue('MSFT');
    myControl.clearNotification('my_unique_id');
}];

myControl.addNotification({
    messages: ['Set Ticker Symbol'],
    notificationLevel: 'RECOMMENDATION',
    uniqueId: 'my_unique_id',
    actions: [actionCollection]
});
}
else
    console.log("Notification not set");
}

```

This how the notification appears in model-driven apps:

ACCOUNT INFORMATION

💡 Account Name * Microsoft

Recommendation

Set Ticker Symbol
Set the Ticker Symbol to MSFT?

Apply

Related topics

[clearNotification](#)

[setNotification](#)

addOnLookupTagClick (Client API reference)

Article • 12/16/2022

Adds an event handler to the [OnLookupTagClick](#) event.

Control types supported

Lookup

Syntax

```
formContext.getControl(arg).addOnLookupTagClick(myFunction);
```

Parameter

Name	Type	Required	Description
myFunction	Function reference	Yes	<p>The function to add to the OnLookupTagClick event. The execution context is automatically passed as the first parameter to this function along with eventArgs that contain the tag value. More information: OnLookupTagClick event.</p> <p>You should use a reference to a named function rather than an anonymous function if you later want to remove the event handler.</p>

Related topics

[removeOnLookupTagClick](#)

[OnLookupTagClick event](#)

addOnOutputChange (Client API reference)

Article • 11/30/2022

Adds an event handler to the [OnOutputChange](#) event.

Control types supported

Standard controls

Syntax

JavaScript

```
var control = formContext.getControl("<name>");  
control.addOnOutputChange(myFunction);
```

Parameters

Name	Type	Required	Description
myFunction	Function	Yes	The function to add to the OnOutputChange event. The execution context is automatically passed as the first parameter to this function.

Related topics

[OnOutputChange event](#)

[removeOnOutputChange](#)

addOnPostSearch (Client API reference)

Article • 12/16/2022

Adds an event handler to the [PostSearch](#) event.

Control types supported

knowledge base search control

Syntax

JavaScript

```
var kbSearchControl = formContext.getControl("<name>");  
kbSearchControl.addOnPostSearch(myFunction);
```

Parameters

Name	Type	Required	Description
myFunction	Function	Yes	The function to add to the PostSearch event. The execution context is automatically passed as the first parameter to this function.

Related topics

[PostSearch event](#)

[removeOnPostSearch](#)

addOnResultOpened (Client API reference)

Article • 12/16/2022

Adds an event handler to the [OnResultOpened](#) event.

Control types supported

knowledge base search control

Syntax

JavaScript

```
var kbSearchControl = formContext.getControl("<name>");  
kbSearchControl.addOnResultOpened(myFunction);
```

Parameters

Name	Type	Required	Description
myFunction	Function	Yes	The function to add to the OnResultOpened event. The execution context is automatically passed as the first parameter to this function.

Related topics

[OnResultOpened event](#)

[removeOnResultOpened](#)

addOnSelection (Client API reference)

Article • 12/16/2022

Adds an event handler to the [OnSelection](#) event.

Control types supported

Knowledge base search control

Syntax

JavaScript

```
var kbSearchControl = formContext.getControl("<name>");  
kbSearchControl.addOnSelection(myFunction);
```

Parameters

Name	Type	Required	Description
myFunction	Function	Yes	The function to add to the OnSelection event. The execution context is automatically passed as the first parameter to this function.

Related topics

[OnSelection event](#)

[removeOnSelection](#)

addOption (Client API reference)

Article • 12/16/2022

Adds an option to a control.

Control types supported

Choice, Choices

Syntax

```
formContext.getControl(arg).addOption(option, index);
```

Parameters

Name	Type	Required	Description
option	Object	Yes	The option to add. The object contains the following: - text : String. The label for the option. - value : Number. The value for the option.
index	Number	No	The index position to place the new option in. If not provided, the option will be added to the end.

Related topics

[clearOptions](#)

[removeOption](#)

addPreSearch (Client API reference)

Article • 08/18/2023

Applies changes to lookups based on values current just as the user is about to view results for the lookup.

Control types supported

Lookup

Syntax

```
formContext.getControl(arg).addPreSearch(myFunction)
```

Parameters

Name	Type	Required	Description
myFunction	Function	Yes	The function that is run just before the search to provide results for a lookup occurs. You can use this function to call one of the other lookup control functions and improve the results to be displayed in the lookup. The execution context is automatically passed as the first parameter to this function.

Example

In the following example, the `onLoad` function is set for the form onload event. It modifies the search filter for all the lookup controls associated with the `primaryid` lookup attribute because there may be more than one.

It adds the `myPreSearchCallBack` function using the `addPreSearch` method. This example requires all the contact records returned to have the `firstname` value of 'Eric'.

JavaScript

```
function onLoad(executionContext) {
    var fromContext = executionContext.getFormContext()
    var attribute = fromContext.getAttribute("primarycontactid")
    attribute.controls.forEach(control =>
        control.addPreSearch(myPreSearchCallBack))
```

```
}

function myPreSearchCallBack (executionContext){
    var control = executionContext.getEventSource();
    var filter = "<filter><condition attribute='firstname' operator='eq'
value='Eric' /></filter>";
    control.addCustomFilter(filter);
}
```

Related articles

[PreSearch event](#)

[removePreSearch](#)

control.clearNotification (Client API reference)

Article • 12/16/2022

Remove a message already displayed for a control.

Control types supported

All

Syntax

```
formContext.getControl(arg).clearNotification(uniqueId);
```

Parameters

Name	Type	Required	Description
uniqueId	String	No	The ID to use to clear a specific message that was set using setNotification or addNotification . If the uniqueId parameter isn't specified, the currently displayed notification will be cleared.

Return Value

Type: Boolean

Description: Indicates whether the method succeeded.

Related topics

[addNotification](#)

[setNotification](#)

clearOptions (Client API reference)

Article • 12/16/2022

Clears all options from a control.

Control types supported

Choice, Choices

Syntax

```
formContext.getControl(arg).clearOptions();
```

Related topics

[addOption](#)

[removeOption](#)

control.getAttribute (Client API reference)

Article • 12/16/2022

Returns the column that the control is bound to.

Controls that aren't bound to a column (subgrid, web resource, and IFRAME) don't have this method. An error will be thrown if you attempt to use this method on one of these controls.

Control types supported

Standard, Lookup, OptionSet

Syntax

```
formContext.getControl(arg).getAttribute();
```

Return Value

Type: Object

Description: A column

Remarks

The constituent controls within a [quick view control](#) are included in the controls collection and these controls have the `getAttribute` method. However, the column is not part of the column collection for the table. While you can retrieve the value for that column using `getValue` and even change the value using `setValue`, changes you make will not be saved with the table.

The following code shows using the value the contact `mobilephone` column when displayed on an account form using a quick view control named `contactQuickForm`. This code hides the control when the value of the column is `null`.

```
JavaScript
```

```
var quickViewMobilePhoneControl =
formContext.getControl("contactQuickForm_contactQuickForm_contact_mobilephon
e");
if (quickViewMobilePhoneControl.getAttribute().getValue() == null) {
    quickViewMobilePhoneControl.setVisible(false);
}
```

Quick view control

Columns

getContentWindow (Client API reference)

Article • 08/19/2023

Returns the content window that represents an IFRAME or web resource.

ⓘ Note

This method is supported only on **Unified Interface**.

Control types supported

iframe, web resource

Syntax

JavaScript

```
formContext.getControl(arg).getContentWindow().then(successCallback,  
errorCallback);
```

Parameters

Name	Type	Required	Description
successCallback	Function	No	A function to call when operation is executed successfully. A content window instance representing the IFRAME or web resource is passed to the function.
errorCallback	Function	No	A function to call when the operation fails.

Return Value

On success, returns a promise that contains a content window instance representing an IFRAME or web resource.

Example

The following example shows how you can use this method with an HTML Web resource (new_myWebResource.htm).

First, add the following code in your HTML web resource:

```
JavaScript

// This script should be in the HTML web resource.
// No usage of Xrm or formContext should happen until this method is called.
function setClientApiClient(xrm, formContext) {
    // Optionally set Xrm and formContext as global variables on the page.
    window.Xrm = xrm;
    window._formContext = formContext;

    // Add script logic here that uses xrm or the formContext.
}
```

Next, add the following code in the form OnLoad event handler:

```
JavaScript

// This should be in a script loaded on the form.
// form_onload is a handler for the form onload event.
function form_onload(executionContext) {
    var formContext = executionContext.getFormContext();
    var wrControl = formContext.getControl("WebResource_CustomName");
    if (wrControl) {
        wrControl.getContentWindow().then(
            function (contentWindow) {
                contentWindow.setClientApiClient(Xrm, formContext);
            }
        )
    }
}
```

Similar initialization code should be added to a [TabStateChange event](#) handler if such initialization is necessary. Any initialization code should be idempotent if it's reused. For performance reasons, the form may destroy and reinitialize the control during tab navigation.

formContext.getControl (Client API reference)

Article • 12/16/2022

Gets a control on the form.

Syntax

```
formContext.getControl(arg);
```

The `formContext.getControl(arg)` method is a shortcut method to access `formContext.ui.controls.get`.

Parameter

`arg`: Optional. You can access a control on a form by passing an argument as either the **name** or the **index value** of the control on a form. For example:

```
formContext.getControl("firstname") or formContext.getControl(0).
```

When the `arg` value is not provided, it returns an array of all the controls on the form. If the `arg` name is spelled wrong and is not on the form, it simply returns null value.

Return Value

Type: Object or Object collection.

Description: Object if you use the method with parameter; object collection if you use the method without any parameters.

Tip

If you want to modify the all the controls bound to a column on a form, use the controls collection inside the column type. For example, to add notification to each control bound to the `name` column, you can do the following:

JavaScript

```
const notification = {
  messages: ['Sample Notification on Name Controls'],
  notificationLevel: 'RECOMMENDATION',
```

```
uniqueId: 'my_unique_id'});  
formContext.getAttribute("name").controls.forEach(control =>  
control.addNotification(notification));
```

Related topics

[formContext](#)

control.getControlType (Client API reference)

Article • 01/10/2023

Returns a value that categorizes controls.

Control Types supported

All

Syntax

```
getControl(arg).getControlType();
```

Return Value:

Type: String

Return Value	Description
standard	A standard control
iframe	An IFRAME control
kbsearch	A knowledge base search control
lookup	A lookup control
choices	A choices control
notes	A notes control
choice	A choice control
quickform	A quick view control
formcomponent	A form component control
subgrid	A subgrid control
timercontrol	A timer control
timelinewall	A timeline control (for Unified Interface)
webresource	A web resource control

Return Value	Description
customcontrol: < <i>namespace</i> >. < <i>name</i> >	A custom control for Dynamics 365 mobile clients (phones and tablets)
customsubgrid:< <i>namespace</i> >. < <i>name</i> >	A custom dataset control for Dynamics 365 mobile clients (phones and tablets)

Related topics

[Controls](#)

[getControl](#)

control.getData (Client API reference)

Article • 12/16/2022

Returns the value of the data query string parameter passed to a Silverlight web resource.

Control types supported

Web resource

Syntax

```
formContext.getControl(arg).getData();
```

Return Value

Type: String

Description: The data value passed to the Silverlight web resource.

Related topics

[setData](#)

getDefaultView (Client API reference)

Article • 12/16/2022

Returns the ID value of the default lookup dialog view.

Control types supported

Lookup

Syntax

```
formContext.getControl(arg).getDefaultView();
```

Return Value

Type: String

Description: ID of the default view.

Related topics

[setDefaultView](#)

control.getDisabled (Client API reference)

Article • 12/16/2022

Returns whether the control is disabled.

Syntax

```
formContext.getControl(arg).getDisabled();
```

Return Value

Type: Boolean.

Description: true if disabled; false otherwise.

Related topics

[setDisabled](#)

getEntityTypes (Client API reference)

Article • 12/16/2022

Gets the types of tables allowed in the lookup control.

Control types supported

Lookup control

Syntax

```
formContext.getControl(arg).getEntityTypes();
```

Return Value

Type: Array of String

Description: The logical names of the tables allowed in this control.

Related topics

[setEntityTypes](#)

getInitialUrl (Client API reference)

Article • 12/16/2022

Returns the default URL that an IFRAME control is configured to display.

Control types supported

iframe

Syntax

```
formContext.getControl(arg).getInitialUrl();
```

Return Value

Type: String

Description: The initial URL.

Related topics

[Controls](#)

control.getLabel (Client API reference)

Article • 12/16/2022

Returns the label for the control.

Control types supported

All

Syntax

```
formContext.getControl(arg).getLabel();
```

Return Value

Type: String

Description: The label of the control.

Related topics

[setLabel](#)

control.getName (Client API reference)

Article • 12/16/2022

Returns the name assigned to the control.

ⓘ Note

The name assigned to a control is not determined until the form loads. Changes to the form may change the name assigned to a given control.

Control types supported

All

Syntax

```
formContext.getControl(arg).getName();
```

Return Value

Type: String

Description: The name of the control.

Related topics

[Controls](#)

getObject (Client API reference)

Article • 12/16/2022

Returns the object in the form that represents an IFRAME or web resource.

Control types supported

iframe, webresource

Syntax

```
formContext.getControl(arg).getObject();
```

ⓘ Note

This method should be used after the `onreadystatechangecomplete` event is triggered. If this is used before, then there is chance that the call to `getObject()` will fail.

Return Value

Type: Object

Description: Object depends on the type of control:

- An IFRAME and HTML web resource returns the [IFrame](#) element from the Document Object Model (DOM).

control.getOptions (Client API reference)

Article • 12/16/2022

Returns an array of option objects representing valid options available for a control, including a blank option and excluding any options that have been removed from the control using [removeOption](#).

Control types supported

Choice, Choices

Syntax

```
formContext.getControl(arg).getOptions()
```

Return Value

Type: Array of option objects.

Description: The array of option objects representing valid options where each option object has the following attributes:

- **text**: String. Label of the option.
- **value**: Number. Enumeration value of the option.

control.getOutputs (Client API reference)

Article • 09/21/2022

Returns a dictionary of the output properties of the control.

Control types supported

Standard controls.

Syntax

```
formContext.getControl(arg).getOutputs();
```

Return Value

Type: Dictionary

Description: The output dictionary of the control.

Related topics

[OnOutputChange event](#)

control.getParent (Client API reference)

Article • 12/16/2022

Returns a reference to the section object that contains the control.

Control types supported

All

Syntax

```
formContext.getControl(arg).getParent();
```

Return Value

Type: [formContext.ui.tabs section](#) object

getSearchQuery (Client API reference)

Article • 12/16/2022

Gets the text used as the search criteria for the knowledge base management control.

Control types supported

knowledge base search control

Syntax

JavaScript

```
var kbSearchControl = formContext.getControl("<name>");  
var searchQuery = kbSearchControl.getSearchQuery();
```

Return Value

Type: String

Description: The text of the search query.

Related topics

[setSearchQuery](#)

getSelectedResults (Client API Reference)

Article • 12/16/2022

Use this method to get the currently selected result of the search control. The currently selected result also represents the result that is currently open.

Control types supported

knowledge base search control

Syntax

JavaScript

```
var kbSearchControl = formContext.getControl("<name>");  
var kbSearchResult = kbSearchControl.getSelectedResults();
```

Return Value

KB SearchResult. The currently selected result.

KB SearchResult properties

Property	Type	Description
answer	String	The HTML markup containing the content of the article. You could pass this content to a custom action that could include it in an email to send to the customer.
articleId	String	The article ID that is used as an alternate key. You can use it to see if this article already exists in Microsoft Dataverse or not.
articleUid	String	The unique article ID. This value is used as an alternate key. This ID is needed to create a new KB record while associating an article if one doesn't exist already.
attachmentCount	Number	Number of attachments in the article.

Property	Type	Description
createdOn	Date	The date the article was created. This value will be in the current user's time zone and format. You may want to use the age of the article in your business logic.
expiredDate	Date	The date the article was or will be expired. You can compare this date to the current data to determine whether the article has expired or not. The value uses the current user's time zone and format.
folderHref	String	The link to the folder path of the article.
href	String	The direct link to the article.
isAssociated	Boolean	Indicates whether the article is associated with the parent record or not. You can check this value before associating the article with the current record using form scripts or in another process initiated by form scripts.
lastModifiedOn	Date	Date on which the article was last modified. This value will be in the current user's time zone and format.
publicUrl	String	Support Portal URL of the article. If the Portal URL option is turned off, this will be blank. Use a custom action to include this in a link in the content of an email to send to a customer.
published	Boolean	Indicates whether the article is in published state. True if published; otherwise False . You should check whether the article is published before you send information about it to a customer.
question	String	The title of the article. If you're going to reference the article in any business process, you can refer to it by name using this value.
rating	Number	The rating of the article.
searchBlurb	String	A short snippet of article content that contains the areas where the search query was hit. Use this to give a glimpse of article to the users in the search list and help them determine if this is the article they are looking for.
serviceDeskUri	String	Link to the article. Use this link to open the article.
timesViewed	Number	The number of times an article is viewed on the portal by customers.

getShowTime (Client API reference)

Article • 12/16/2022

Get whether a date control shows the time portion of the date.

Control types supported

standard control for **datetime** columns.

Syntax

```
formContext.getControl(arg).getShowTime();
```

Return Value

Type: Boolean

Description: true if shows the time portion of the date; false otherwise.

Related topics

[setShowTime](#)

getSrc (Client API reference)

Article • 12/16/2022

Returns the current URL being displayed in an IFRAME or web resource.

Control types supported

iframe, webresource

Syntax

```
formContext.getControl(arg).getSrc();
```

Return Value

Type: String

Description: A URL representing the **src** property of the IFRAME or web resource.

Related topics

[setSrc](#)

getState (Client API reference)

Article • 12/16/2022

Returns the state of the timer control.

Control types supported

Timer

Syntax

```
formContext.getControl(arg).getState();
```

Return Value

Type: Number

Description: Returns one of the following values:

Value	State
1	Not Set
2	In progress
3	Warning
4	Violated
5	Success
6	Expired
7	Canceled
8	Paused

Related topics

[Controls](#)

getTotalResultCount (Client API reference)

Article • 12/16/2022

Gets the count of results found in the search control.

Control types supported

knowledge base search control

Syntax

JavaScript

```
var kbSearchControl = formContext.getControl("<name>");  
var searchCount = kbSearchControl.getTotalResultCount();
```

Return Value

Type: Number

Description: The count of the search result.

control.getValue (Client API reference)

Article • 12/16/2022

Gets the latest value in a control as the user types characters in a specific text or number column. This method helps you to build interactive experiences by validating data and alerting users as they type characters in a control.

The `getValue` method is different from the column `getValue` method because the control method retrieves the value from the control as the user is typing in the control as opposed to the column `getValue` method that retrieves the value after the user commits (saves) the column.

Syntax

```
formContext.getControl(arg).getValue();
```

Return Value

Type: String

Description: The latest data value for a control.

control.getVisible (Client API reference)

Article • 12/16/2022

Returns a value that indicates whether the control is currently visible.

ⓘ Note

`getVisible` only returns whether the control is configured to be visible. `getVisible` will return true when the control is within a section or tab that is hidden.

Control types supported

All

Syntax

```
formContext.getControl(arg).getVisible();
```

Return Value

Type: Boolean.

Description: true if the control is visible; false otherwise.

Related topics

[setVisible](#)

openSearchResult (Client API reference)

Article • 12/16/2022

Opens a search result in the search control by specifying the result number.

Control types supported

knowledge base search control

Syntax

JavaScript

```
var kbSearchControl = formContext.getControl("<name>");  
var openResultStatus = kbSearchControl.openSearchResult(resultNumber, mode);
```

Parameter

Name	Type	Required	Description
resultNumber	Number	Yes	Numerical value specifying the result number to be opened. Result number starts from 1.
mode	String	No	<p>Specify "Inline" or "Popout". If you do not specify a value for the argument, the default ("Inline") option is used.</p> <p>The "Inline" mode opens the result inline either in the reading pane of the control or in a reference panel tab in case of reference panel. The "Popout" mode opens the result in a pop-out window.</p>

Return Value

Type: Boolean

Description: Status of opening the specified search result. Returns 1 if successful; 0 if unsuccessful. The method will return -1 if the specified resultNumber value is not present, or if the specified mode value is invalid.

control.refresh (Client API reference)

Article • 12/16/2022

Refreshes the data displayed in a timelinewall and timer control.

Control types supported

timelinewall, timer

Syntax

```
formContext.getControl(arg).refresh();
```

Related topics

[Controls](#)

removeOnLookupTagClick (Client API reference)

Article • 12/16/2022

Removes an event handler from the [OnLookupTagClick](#) event.

Control types supported

Lookup

Syntax

```
formContext.getControl(arg).removeOnLookupTagClick(myFunction);
```

Parameter

Name	Type	Required	Description
myFunction	Function reference	Yes	The function to be removed from the OnLookupTagClick event.

Related topics

[addOnLookupTagClick](#)

[OnLookupTagClick](#) event

removeOnOutputChange (Client API reference)

Article • 08/18/2022

Removes an event handler from the [OnOutputChange](#) event.

Control types supported

Standard control

Syntax

JavaScript

```
var control = formContext.getControl("<name>");  
control.removeOnOutputChange(myFunction);
```

Parameters

Name	Type	Required	Description
myFunction	Function	Yes	The function to remove from the OnOutputChange event.

Related topics

[addOnOutputChange](#)

removeOnPostSearch (Client API reference)

Article • 12/16/2022

Removes an event handler from the [PostSearch](#) event.

Control types supported

knowledge base search control

Syntax

JavaScript

```
var kbSearchControl = formContext.getControl("<name>";  
kbSearchControl.removeOnPostSearch(myFunction);
```

Parameters

Name	Type	Required	Description
myFunction	Function	Yes	The function to remove from the PostSearch event.

Related topics

[PostSearch event](#)

[addOnPostSearch](#)

removeOnResultOpened (Client API reference)

Article • 12/16/2022

Removes an event handler from the [OnResultOpened](#) event.

Control types supported

knowledge base search control

Syntax

JavaScript

```
var kbSearchControl = formContext.getControl("<name>");  
kbSearchControl.removeOnResultOpened(myFunction);
```

Parameters

Name	Type	Required	Description
myFunction	Function	Yes	The function to remove from the OnResultOpened event.

Related topics

[OnResultOpened event](#)

[addOnResultOpened](#)

removeOnSelection (Client API reference)

Article • 12/16/2022

Removes an event handler from the [OnSelection](#) event.

Control types supported

Knowledge base search control

Syntax

JavaScript

```
var kbSearchControl = formContext.getControl("<name>");  
kbSearchControl.removeOnSelection(myFunction);
```

Parameters

Name	Type	Required	Description
myFunction	Function	Yes	The function to remove from the OnSelection event.

Related topics

[addOnSelection](#)

removeOption (Client API reference)

Article • 12/16/2022

Removes an option from a control.

Control types supported

choice, choices

Syntax

```
formContext.getControl(arg).removeOption(value);
```

Parameters

Name	Type	Required	Description
value	Number	Yes	The value of the option you want to remove.

Related topics

[addOption](#)

[clearOptions](#)

removePreSearch (Client API reference)

Article • 12/16/2022

Removes event handler functions that have previously been set for the [PreSearch](#) event.

Control types supported

Lookup

Syntax

```
formContext.getControl(arg).removePreSearch(myFunction)
```

Parameters

Name	Type	Required	Description
myFunction	Function	Yes	The function to remove. The execution context is automatically passed as the first parameter to this function.

Related topics

[PreSearch event](#)

[addPreSearch](#)

setData (Client API reference)

Article • 12/16/2022

Sets the value of the data query string parameter passed to a Silverlight web resource.

Control types supported

webresource

Syntax

```
formContext.getControl(arg).setData(string);
```

Parameter

Name	Type	Required	Description
string	String	Yes	The data value to pass to the Silverlight web resource.

Related topics

[getData](#)

setDefaultView (Client API reference)

Article • 12/16/2022

Sets the default view for the lookup control dialog box.

Control types supported

Lookup

Syntax

```
formContext.getControl(arg).setDefaultView(viewId);
```

Parameter

Name	Type	Required	Description
viewId	String	Yes	The ID of the view to be set as the default view.

Example

This `setDefaultViewSample` function will set the **account** form primary contact lookup default view to the **My Active Contacts** view.

JavaScript

```
function setDefaultViewSample(executionContext) {
    var formContext = executionContext.getFormContext();
    formContext.getControl("primarycontactid").setDefaultView("{00000000-0000-00AA-000010001003}");
}
```

Related topics

[getDefaultView](#)

control.setDisabled (Client API reference)

Article • 12/16/2022

Sets whether the control is disabled.

Control types supported

All except kbsearch control type

Syntax

```
formContext.getControl(arg).setDisabled(bool);
```

Parameter

Name	Type	Required	Description
bool	Boolean	Yes	Specify true or false to disable or enable the control.

ⓘ Note

If a control bound to a Business Required column is set to be disabled, the form will no longer require it to have a value before saving. See [Column requirement level](#) for more information.

Related topics

[getDisabled](#)

setEntityTypes (Client API reference)

Article • 12/16/2022

Sets the types of tables allowed in the lookup control.

Control types supported

Lookup control

Syntax

```
formContext.getControl(arg).setEntityTypes([entityLogicalNames]);
```

Parameter

Name	Type	Required	Description
entityLogicalNames	Array of String	Yes	Specify the logical name of the tables allowed in the lookup control.

Related topics

[getEntityTypes](#)

control.setFocus (Client API reference)

Article • 12/16/2022

Sets the focus on the control.

Control types supported

all

Syntax

```
formContext.getControl(arg).setFocus();
```

control.setLabel (Client API reference)

Article • 12/16/2022

Sets the label of the control.

Control types supported

All

Syntax

```
formContext.getControl(arg).setLabel(label);
```

Parameter

Name	Type	Required	Description
label	String	Yes	The new label of the control.

Related topics

[getLabel](#)

setNotification (Client API reference)

Article • 12/16/2022

Displays an error message for the control to indicate that data isn't valid. When this method is used, a red "X" icon appears next to the control. On Dynamics 365 mobile clients, tapping on the icon will display the message.

ⓘ Note

Always make sure that the control you are using `setNotification` with is visible on the form.

`setNotification` will not return an error if you use it on a control that is visible but is within a section or tab that isn't visible. When saving a record `setNotification` will display a message at the top of the form as a validation error. If the control is not visible because it is within a hidden parent section or tab, the user cannot fix this.

Control types supported

Standard, lookup, choices and choice control types.

Syntax

```
formContext.getControl(arg).setNotification(message,uniqueId);
```

Parameters

Name	Type	Required	Description
message	String	Yes	The message to display.
uniqueId	String	No	The ID to use to clear this message when using the <code>clearNotification</code> method.

Return Value

Type: Boolean

Description: Indicates whether the method succeeded.

Remarks

Setting an error notification on a control will block the form from saving.

Related topics

[addNotification](#)

[clearNotification](#)

setSearchQuery (Client API reference)

Article • 12/16/2022

Sets the text used as the search criteria for the knowledge base search control.

Control types supported

knowledge base search control

Syntax

JavaScript

```
var kbSearchControl = formContext.getControl("<name>");  
kbSearchControl.setSearchQuery(searchString);
```

Parameters

Name	Type	Required	Description
searchString	String	Yes	The text for the search query.

Related topics

[getSearchQuery](#)

setShowTime (Client API reference)

Article • 12/16/2022

Specify whether a date control should show the time portion of the date.

Control types supported

standard control for **datetime** attributes.

Syntax

```
formContext.getControl(arg).setShowTime(bool);
```

Parameter

Name	Type	Required	Description
bool	Boolean	Yes	Specify true to show the time portion of the date; false otherwise.

Remarks

This method will show or hide the time component of a date control where the attribute uses the **DateAndTime** format. This method will have no effect when the **DateOnly** format is used.

Related topics

[getShowTime](#)

setSrc (Client API reference)

Article • 12/16/2022

Sets the URL to be displayed in an IFRAME or web resource.

Control types supported

iframe, webresource

Syntax

```
formContext.getControl(arg).setSrc(string);
```

Parameter

Name	Type	Required	Description
string	String	Yes	The URL.

Related topics

[getSrc](#)

[Known issues: Component from an IFRAME](#)

control.setVisible (Client API reference)

Article • 12/16/2022

Sets a value that indicates whether the control is visible.

Control types supported

All

Syntax

```
formContext.getControl(arg).setVisible(bool);
```

Parameter

Name	Type	Required	Description
bool	Boolean	Yes	Specify true to show the control; false to hide the control.

ⓘ Note

If a control is set to false and is in a section that is hidden and if you set the control to true, the section will be visible.

ⓘ Note

If a control bound to a Business Required column is set to not be visible, the form will no longer require it to have a value before saving. See [Column requirement level](#) for more information.

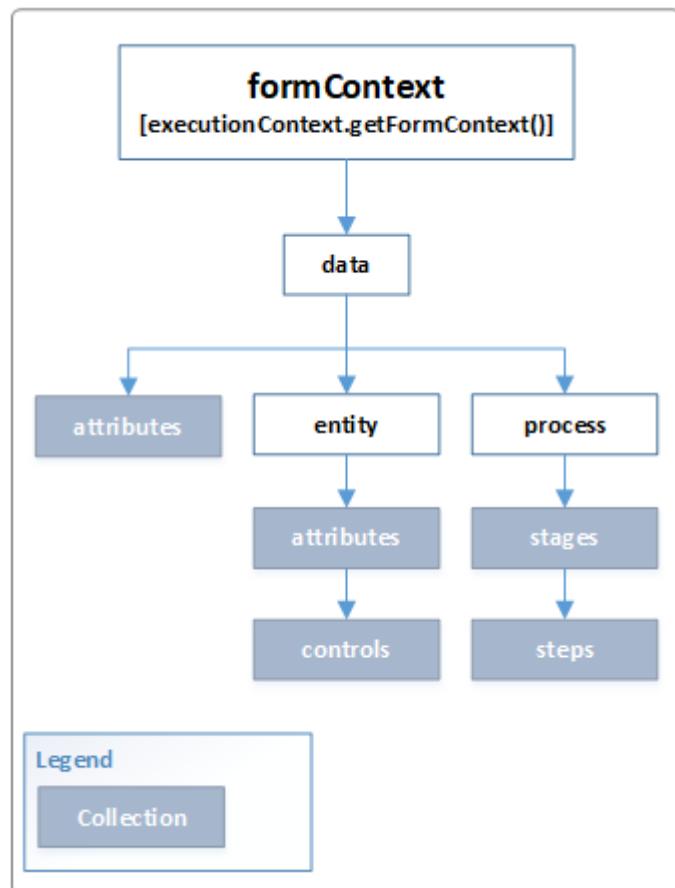
Related topics

[getVisible](#)

formContext.data (Client API reference)

Article • 11/30/2022

Provides properties and methods to work with the data on a form.



Properties

Name	Description
attributes	Collection of non-table data on the form. Items in this collection are of the same type as the columns collection, but they are not columns of the form table. More information: Collections
entity	Provides methods to retrieve information specific to the record displayed on the page, the save method, and a collection of all the columns included on the form. Column data is limited to columns represented on the form. More information: formContext.data.entity
process	Provides objects and methods to interact with the business process flow data on a form. More information: formContext.data.process

Methods

Name	Description
addOnLoad	Adds a function to be called when form data is loaded.
getIsDirty	Gets a boolean value indicating whether the form data has been modified.
isValid	Gets a boolean value indicating whether all of the form data is valid. If the form has empty required columns on it, control-level error notifications will be shown. A column may also be set as invalid using the setIsValid method <code>setIsValid</code> method.
refresh	Asynchronously refreshes and optionally saves all the data of the form without reloading the page. The form data onload event occurs after the data is refreshed.
removeOnLoad	Removes a function to be called when form data is loaded.
save	Saves the record asynchronously with the option to set callback functions to be executed after the save operation is completed.

Related topics

[formContext.data.entity](#)

[formContext.data.process](#)

data.addOnLoad (Client API reference)

Article • 11/30/2022

Adds a function to be called when form data is loaded.

Syntax

```
formContext.data.addOnLoad(myFunction)
```

Parameter

Name	Type	Required	Description
myFunction	function reference	Yes	The function to be executed when the form data loads. The function will be added to the bottom of the event handler pipeline. The execution context is automatically passed as the first parameter to the function. See Execution context for more information.

Related topics

[removeOnLoad](#)

[Form data OnLoad event](#)

data.getIsDirty (Client API reference)

Article • 11/30/2022

Gets a boolean value indicating whether the form data has been modified.

Syntax

```
formContext.data.getIsDirty();
```

Return Type

Type: Boolean

Description: true if the form data has changed; false otherwise.

Related topics

[formContext.data.entity.getIsDirty](#)

[formContext](#)

data.isValid (Client API reference)

Article • 11/30/2022

Gets a boolean value indicating whether all of the form data is valid. If the form has empty required columns on it, control-level error notifications will be shown. A column may also be set as invalid using the [setIsValid method](#) setIsValid method.

Syntax

```
formContext.data.isValid();
```

Return Type

Type: Boolean

Description: true if all of the form data is valid; false otherwise.

Related topics

[formContext](#)

data.refresh (Client API reference)

Article • 11/30/2022

Asynchronously refreshes and optionally saves all the data of the form without reloading the page. The [form data onload event](#) occurs after the data is refreshed.

Syntax

```
formContext.data.refresh(save).then(successCallback, errorCallback);
```

Parameter

Name	Type	Required	Description
save	Boolean	No	true if the data should be saved before it is refreshed, otherwise false.
successCallback	Function	No	A function to call when the operation succeeds.
errorCallback	Function	No	A function to call when the operation fails.

Related topics

[formContext](#)

data.removeOnLoad (Client API reference)

Article • 11/30/2022

Removes a function to be called when form data is loaded.

Syntax

```
formContext.data.removeOnLoad(myFunction)
```

Parameter

Name	Type	Required	Description
myFunction	function reference	Yes	The function to be removed when the form data loads.

Related topics

[addOnLoad](#)

[Form data OnLoad event](#)

formContext.data.save (Client API reference)

Article • 11/30/2022

Saves the record asynchronously with the option to set callback functions to be executed after the save operation is completed.

You can also set an object to control how appointment, recurring appointment, or service activity records are processed.

Syntax

```
formContext.data.save(saveOptions).then(successCallback, errorCallback);
```

Parameters

Name	Type	Required	Description
saveOptions	Object	No	<p>An object for specifying options for saving the record. The object has following parameters:</p> <ul style="list-style-type: none">- saveMode: (Optional) Number. Specify a value indicating how the save event was initiated. For a list of supported values, see the return value of the getSaveMode method. Note that setting the saveMode does not actually take the corresponding action; it is just to provide information to the OnSave event handlers about the reason for the save operation.- useSchedulingEngine: (Optional) Boolean. Indicate whether to use the Book or Reschedule messages rather than the Create or Update messages. This option is only applicable when used with appointment, recurring appointment, or service activity records. <p>NOTE: <code>useSchedulingEngine</code> property is not supported in Unified Interface.</p>
successCallback	Function	No	A function to call when the operation succeeds.

Name	Type	Required	Description
errorCallback	Function	No	A function to call when the operation fails. An object with the following properties will be passed: <ul style="list-style-type: none">- errorCode: Number. The error code.- message: String. A localized error message.

ⓘ Note

When working with forms, and you call the `formContext.data.save` method, make sure that you also call the `preventDefault` to ensure that any default save operation is not triggered when a user saves the form.

Related topics

[formContext.data.entity.save](#)

[formContext](#)

formContext.data.entity (Client API reference)

Article • 11/30/2022

Provides properties and methods to retrieve information specific to the record displayed on the page, the save method, and a collection of all the columns included in the form. Column data is limited to columns represented on the form.

Properties

Name	Description
attributes	Collection of columns for a record displayed on the form. More information: Collections and Columns .

Methods

Name	Description
addOnSave	Adds a function to be called when the OnSave event is triggered. The event occurs before the save occurs, giving the handler an option to cancel the save operation.
addOnPostSave	Adds an event handler to the PostSave Event event.
getDataXml	Returns a string representing the XML that will be sent to the server when the record is saved. Only data in columns that have changed or have their submit mode set to "always" are sent to the server.
getEntityName	Returns a string representing the logical name of the table for the record.
getEntityReference	Returns a lookup value that references the record.
getId	Returns a string representing the GUID value for the record.
getIsDirty	Gets a boolean value indicating whether any columns in the form have been modified.
getPrimaryAttributeValue	Gets a string for the value of the primary column of the table.
isValid	Gets a boolean value indicating whether all of the table data is valid.
removeOnPostSave	Removes an event handler from the PostSave Event event.

Name	Description
removeOnSave	Removes a function to be called when the record is saved.
save	Saves the record synchronously with the options to close the form or open a new form after the save is completed.

Related topics

[Understand Xrm object model](#)

[Controls \(Client API reference\)](#)

addOnPostSave (Client API reference)

Article • 08/04/2023

Adds an event handler to the [PostSave Event](#) event.

ⓘ Note

This API is not available for Dynamics 365 Customer Engagement on-premise deployments. More information: [Client APIs not supported in Customer Engagement \(on-premises\)](#)

Syntax

JavaScript

```
formContext.data.entity.addOnPostSave(myFunction);
```

Parameter

Name	Type	Required	Description
myFunction	function reference	Yes	The function to be added to the <code>PostSave</code> event after the record is saved with success or failure.

See also

[PostSave event](#)

[removeOnPostSave](#)

addOnSave (Client API reference)

Article • 11/30/2022

Adds a function to be called when the [OnSave](#) event is triggered. The event occurs before the save occurs, giving the handler an option to cancel the save operation.

Syntax

```
formContext.data.entity.addOnSave(myFunction)
```

Parameter

Name	Type	Required	Description
myFunction	function reference	Yes	The function to be executed when the record is saved. The function will be added to the bottom of the event handler pipeline. The execution context is automatically passed as the first parameter to the function. See Execution context for more information.

Related topics

[removeOnSave](#)

[Form OnSave event](#)

getDataXml (Client API reference)

Article • 11/30/2022

Returns a string representing the XML that will be sent to the server when the record is saved. Only data in columns that have changed or have their submit mode set to "always" are sent to the server.

Syntax

```
formContext.data.entity.getDataXml();
```

Return Value

Type: String.

Description: In this example, the following three columns for an account record were updated: name, accountnumber, telephone2.

```
"<account><name>Contoso</name><accountnumber>55555</accountnumber><telephone2>425  
555-1234</telephone2></account>"
```

Remarks

This method does not work with Microsoft Dynamics 365 for tablets.

entity.getEntityName (Client API reference)

Article • 11/30/2022

Returns a string representing the logical name of the table for the record.

Syntax

```
formContext.data.entity.getEntityName();
```

Return Value

Type: String.

Description: The name of the table.

entity.getEntityReference (Client API reference)

Article • 11/30/2022

Returns a lookup value that references the record.

Syntax

```
formContext.data.entity.getEntityReference();
```

Return Value

Type: Lookup object.

Description: The returned object has following three parameters:

- **entityType**: String. Logical name of the table record. For example, "account".
- **id**: String. GUID value of the table record.
- **name**: (Optional) String. Name of the table record.

entity.getId (Client API reference)

Article • 11/30/2022

Returns a string representing the GUID value for the record.

Syntax

```
formContext.data.entity.getId();
```

Return Value

Type: String.

Description: The GUID value for the record.

entity.getIsDirty (Client API reference)

Article • 11/30/2022

Gets a boolean value indicating whether any columns in the form have been modified.

Syntax

```
formContext.data.entity.getIsDirty();
```

Return Type

Type: Boolean

Description: true if any columns in the form have been changed; false otherwise.

Related topics

[formContext.data.getIsDirty](#)

[formContext](#)

entity.getPrimaryAttributeValue (Client API reference)

Article • 11/30/2022

Gets a string for the value of the primary column of the table.

Syntax

```
formContext.data.entity.getPrimaryAttributeValue();
```

Return Value

Type: String.

Description: The name of the table.

Remarks

Each table has one string column that is designated as the [PrimaryNameAttribute](#). The value for this column is used when links to the record are displayed.

entity.isValid (Client API reference)

Article • 11/30/2022

Gets a boolean value indicating whether all of the table data is valid.

Syntax

```
formContext.data.entity.isValid();
```

Return Type

Type: Boolean

Description: true if all of the table data is valid; false otherwise.

Related topics

[formContext](#)

removeOnPostSave (Client API reference)

Article • 08/04/2023

Removes an event handler from the [PostSave Event](#) event.

ⓘ Note

This API is not available for Dynamics 365 Customer Engagement on-premise deployments. More information: [Client APIs not supported in Customer Engagement \(on-premises\)](#)

Syntax

JavaScript

```
formContext.data.entity.removeOnPostSave(myFunction);
```

Parameter

Name	Type	Required	Description
myFunction	function reference	Yes	The function to be removed from the PostSave event.

See also

[PostSave event](#)

[addOnPostSave](#)

removeOnSave (Client API reference)

Article • 11/30/2022

Removes a function to be called when the record is saved.

Syntax

```
formContext.data.entity.removeOnSave(myFunction)
```

Parameter

Name	Type	Required	Description
myFunction	function reference	Yes	The function to be removed for the OnSave event.

Related topics

[addOnSave](#)

[Form OnSave event](#)

entity.save (Client API reference)

Article • 11/30/2022

Saves the record synchronously with the options to close the form or open a new form after the save is completed.

Syntax

```
formContext.data.entity.save(saveOption);
```

Note

This method is deprecated and we recommend to use the `formContext.data.save` method.

Parameters

Name	Type	Required	Description
saveOption	String	No	<p>Specify options for saving the record. If no parameter is included in the method, the record will simply be saved. This is the equivalent of using the Save command.</p> <p>You can specify one of the following values:</p> <ul style="list-style-type: none">- saveandclose: This is the equivalent of using the Save and Close command.- saveandnew: This is the equivalent of the using the Save and New command.

Example

To open a new form after the save is completed:

```
formContext.data.entity.save("saveandnew");
```

Related topics

[formContext.data.save](#)

formContext

formContext.data.process (Client API reference)

Article • 11/30/2022

Provides events, methods, and objects to interact with the business process flow data on a form. See [formContext.ui.process \(Client API reference\)](#) for methods to interact with the business process flow control on the form.

Process events and event handler methods

Use the following events and event handler methods to write scripts for business process flows.

Event	Event handler methods
OnPreProcessStatusChange	<code>addOnPreProcessStatusChange</code> <code>removeOnPreProcessStatusChange</code>
OnProcessStatusChange	<code>addOnProcessStatusChange</code> <code>removeOnProcessStatusChange</code>
OnPreStageChange	<code>addOnPreStageChange</code> <code>removeOnPreStageChange</code>
OnStageChange	<code>addOnStageChange</code> <code>removeOnStageChange</code>
OnStageSelected	<code>addOnStageSelected</code> <code>removeOnStageSelected</code>

Active Process methods

Use these methods to retrieve information about the active process and set a different process as the active process.

Name	Description
getActiveProcess	Returns a <code>Process</code> object representing the active process.
setActiveProcess	Sets a <code>Process</code> as the active process.

Process methods

A process contains the data for a business process flow. Use the methods to access properties of the process.

Name	Description
<code>getId</code>	Returns the unique identifier of the process.
<code>getName</code>	Returns the name of the process.
<code>getStages</code>	Returns a collection of stages in the process.
<code>isRendered</code>	Returns a boolean value indicating whether the process is rendered.

ProcessInstance methods

Use these methods to retrieve information about all the process instances for a record and to set a process instance as the active instance.

Name	Description
<code>getProcessInstances</code>	Returns all the process instances for the table record that the calling user has access to.
<code>setActiveProcessInstance</code>	Sets a process instance as the active instance.

Instance methods

A process instance contains the data for an instance of the business process flow. Use the methods to access properties of the process instance.

Name	Description
<code>getInstanceId</code>	Returns the unique identifier of the process instance.
<code>getInstanceName</code>	Returns the name of the process instance.
<code>getStatus</code>	Returns the current status of the process instance.
<code>setStatus</code>	Sets the current status of the active process instance.

Active Stage methods

Use these methods to retrieve information about the active stage and set a different stage as the active stage.

Name	Description
getActiveStage	Returns a Stage object representing the active stage.
setActiveStage	Sets a completed stage as the active stage.

Stage methods

A stage contains the data for a stage in a business process flow. Use the methods to access properties of the stage.

Name	Description
getCategory	Returns an object with a getValue method which will return the integer value of the business process flow category.
getEntityName	Returns the logical name of the table associated with the stage.
getId	Returns the unique identifier of the stage.
getName	Returns the name of the stage.
getNavigationBehavior	Returns a navigation behavior object for a stage that can be used to define whether the Create button is available for users to create other table record in a cross-table business process flow navigation scenario.
getStatus	Returns the status of the stage.
getSteps	Returns a collection of steps in the stage.

Step methods

A step contains the data for a step in a stage in a business process flow. Use the methods to access properties of the step.

Name	Description
getAttribute	Returns the logical name of the column associated to the step.
getName	Returns the name of the step.
getProgress	Returns the progress of the action step.
isRequired	Returns a boolean value indicating whether the step is required in the business process flow.
setProgress	Updates the progress of the action step.

Navigation methods

Use these methods to move to next and previous stages. Both these methods will cause the OnStageChange event to occur.

Name	Description
moveNext	Progresses to the next stage.
movePrevious	Moves to the previous stage.

Other useful methods

Use these methods to find information about the stages in the active path, enabled processes, and selected stage.

Name	Description
getActivePath	Gets a collection of stages currently in the active path with methods to interact with the stages displayed in the business process flow control.
getEnabledProcesses	Asynchronously retrieves the business process flows enabled for a table that the current user can switch to.
getSelectedStage	Gets the currently selected stage.

Related topics

[formContext.ui.process \(Client API reference\)](#)

[Understand Xrm object model](#)

[Controls \(Client API reference\)](#)

addOnPreProcessStatusChange (Client API reference)

Article • 10/16/2022

Applies to Dynamics 365 (online), version 9.x

Adds a function as an event handler for the [OnPreProcessStatusChange](#) event so that it will be called **before** the business process flow status changes.

Syntax

```
formContext.data.process.addOnPreProcessStatusChange(myFunction);
```

Parameter

Name	Type	Required	Description
myFunction	Function reference	Yes	<p>The function to be executed when the business process flow status changes. The function will be added to the start of the event handler pipeline. The execution context is automatically passed as the first parameter to the function. See Execution context for more information.</p> <p>You should use a reference to a named function rather than an anonymous function if you may later want to remove the event handler.</p>

This client API is only supported on the Unified Client. The legacy web client does not support this client API.

Related topics

[removeOnPreProcessStatusChange](#)

[formContext.data.process](#)

addOnPreStageChange (Client API reference)

Article • 10/16/2022

Adds a function as an event handler for the [OnPreStageChange](#) event so that it will be called **before** the business process flow stage changes.

Syntax

```
formContext.data.process.addOnPreStageChange(myFunction);
```

Parameter

Name	Type	Required	Description
myFunction	Function reference	Yes	<p>The function that runs before the business process flow stage changes. The function will be added to the start of the event handler pipeline. The execution context is automatically passed as the first parameter to the function. See Execution context for more information.</p> <p>You should use a reference to a named function rather than an anonymous function if you may later want to remove the event handler.</p>

This client API is only supported on the Unified Client. The legacy web client does not support this client API.

Related topics

- [addOnStageChange](#)
- [removeOnStageChange](#)
- [formContext.data.process](#)

addOnProcessStatusChange (Client API reference)

Article • 11/30/2022

Adds a function as an event handler for the [OnProcessStatusChange](#) event so that it will be called when the business process flow status changes.

Syntax

```
formContext.data.process.addOnProcessStatusChange(myFunction);
```

Parameter

Name	Type	Required	Description
myFunction	Function reference	Yes	<p>The function to be executed when the business process flow status changes. The function will be added to the bottom of the event handler pipeline. The execution context is automatically passed as the first parameter to the function. See Execution context for more information.</p> <p>You should use a reference to a named function rather than an anonymous function if you may later want to remove the event handler.</p>

Related topics

[removeOnProcessStatusChange](#)

[formContext.data.process](#)

addOnStageChange (Client API reference)

Article • 11/30/2022

Adds a function as an event handler for the [OnStageChange](#) event so that it will be called when the business process flow stage changes.

Syntax

```
formContext.data.process.addOnStageChange(myFunction);
```

Parameter

Name	Type	Required	Description
myFunction	Function reference	Yes	<p>The function to be executed when the business process flow stage changes. The function will be added to the bottom of the event handler pipeline. The execution context is automatically passed as the first parameter to the function. See Execution context for more information.</p> <p>You should use a reference to a named function rather than an anonymous function if you may later want to remove the event handler.</p>

Related topics

[removeOnStageChange](#)

[formContext.data.process](#)

addOnStageSelected (Client API reference)

Article • 11/30/2022

Adds a function as an event handler for the [OnStageSelected](#) event so that it will be called when a business process flow stage is selected.

Syntax

```
formContext.data.process.addOnStageSelected(myFunction);
```

Parameter

Name	Type	Required	Description
myFunction	Function reference	Yes	<p>The function to be executed when the business process flow stage is selected. The function will be added to the bottom of the event handler pipeline. The execution context is automatically passed as the first parameter to the function. See Execution context for more information.</p> <p>You should use a reference to a named function rather than an anonymous function if you may later want to remove the event handler.</p>

Related topics

[removeOnStageSelected](#)

[formContext.data.process](#)

removeOnPreProcessStatusChange (Client API reference)

Article • 10/16/2022

Applies to Dynamics 365 (online), version 9.x

Removes an event handler from the [OnPreProcessStatusChange](#) event.

Syntax

```
formContext.data.process.removeOnPreProcessStatusChange(myFunction);
```

Parameter

Name	Type	Required	Description
myFunction	Function reference	Yes	The function to be removed from the OnPreProcessStatusChange event.

Related topics

[addOnProcessStatusChange](#)

[formContext.data.process](#)

removeOnPreStageChange (Client API reference)

Article • 11/30/2022

Removes an event handler from the [OnPreStageChange](#) event.

Syntax

```
formContext.data.process.removeOnPreStageChange(myFunction);
```

Parameter

Name	Type	Required	Description
myFunction	Function reference	Yes	The function to be removed from the OnPreStageChange event.

Related topics

[addOnPreStageChange](#)

[formContext.data.process](#)

removeOnProcessStatusChange (Client API reference)

Article • 11/30/2022

Removes an event handler from the [OnProcessStatusChange](#) event.

Syntax

```
formContext.data.process.removeOnProcessStatusChange(myFunction);
```

Parameter

Name	Type	Required	Description
myFunction	Function reference	Yes	The function to be removed from the OnProcessStatusChange event.

Related topics

[addOnProcessStatusChange](#)

[formContext.data.process](#)

removeOnStageChange (Client API reference)

Article • 11/30/2022

Removes an event handler from the [OnStageChange](#) event.

Syntax

```
formContext.data.process.removeOnStageChange(myFunction);
```

Parameter

Name	Type	Required	Description
myFunction	Function reference	Yes	The function to be removed from the OnStageChange event.

Related topics

[addOnStageChange](#)

[formContext.data.process](#)

removeOnStageSelected (Client API reference)

Article • 11/30/2022

Removes an event handler from the [OnStageSelected](#) event.

Syntax

```
formContext.data.process.removeOnStageSelected(myFunction);
```

Parameter

Name	Type	Required	Description
myFunction	Function reference	Yes	The function to be removed from the OnStageSelected event.

Related topics

[addOnStageSelected](#)

[formContext.data.process](#)

getActiveProcess (Client API reference)

Article • 11/30/2022

Returns a **Process** object representing the active process.

Syntax

```
var activeProcess = formContext.data.process.getActiveProcess();
```

Return Value

Type: Process | null

Description: The currently active process or null if there is no active process. See [Process methods](#) for the methods to access the properties of the process returned.

Related topics

[setActiveProcess](#)

[formContext.data.process](#)

setActiveProcess (Client API reference)

Article • 11/30/2022

Sets a **Process** as the active process.

If the business process being set as active already has an instance, the instance will be marked as active and the primary record form will be reloaded to show it. If no instance exists for the process, a new instance is created, marked as active, and the primary record form will be reloaded to show it.

If multiple instances of the process exist, one of these will be chosen as per the defaulting logic to be marked as active. Typically, this is the most recently used process instance by the current user.

ⓘ Note

- The `setActiveProcess` method should be used while creating or editing a record. Use the `setActiveProcess` method to set the active process for a business process flow instead of `processId` and `processInstanceId` with `Xrm.Navigation.openForm` method .
- The `setActiveProcess` method will reload the form and any unsaved data will be lost.

Syntax

```
formContext.data.process.setActiveProcess(processId, callbackFunction);
```

Parameter

Name	Type	Required	Description
processId	String	Yes	The Id of the process to set as the active process.
callbackFunction	Function	No	A function to call when the operation is complete. This callback function is passed one of the following string values to indicate whether the operation succeeded: - success : The operation succeeded. - invalid : The processId isn't valid or the process isn't enabled.

Related topics

[getActiveProcess](#)

[setActiveProcessInstance](#)

[formContext.data.process](#)

getEnabledProcesses (Client API reference)

Article • 11/30/2022

Asynchronously retrieves the business process flows enabled for a table that the current user can switch to.

Syntax

```
formContext.data.process.getEnabledProcesses(callbackFunction(enabledProcesses));
```

Parameter

Name	Type	Required	Description
callbackFunction	Function	Yes	<p>The callback function must accept a parameter that contains an object with dictionary properties where the name of the property is the Id of the business process flow and the value of the property is the name of the business process flow.</p> <p>The enabled processes are filtered according to the user's privileges. The list of enabled processes is the same ones a user can see in the UI if they want to change the process manually.</p>

Example

The **Sdk.formOnLoad** function in the example uses the **formContext.data.process.getEnabledProcesses** method to asynchronously retrieve information about business process flows that are enabled for the table. The sample passes an anonymous function as the first parameter. This function is executed asynchronously when the data is returned and the data is passed as the parameter to the anonymous function.

The information about enabled business process flow is provided as a dictionary object where the Id of the process is the name of the property and the name of the business process flow is the value of the property. The sample code processes this information and sets the values in a global **Sdk.enabledProcesses** array to be accessed by logic that executes later. The sample also loops through the values in the **Sdk.enabledProcesses**

array, and uses the `Sdk.writeToConsole` function to write information about the retrieved business process flows to the console.

① Note

The `Sdk.formOnLoad` function in the sample JavaScript library must be set as the **OnLoad** event handler for a form, and the **Pass execution context as the first parameter** check box must be selected in the **Handler Properties** dialog.

Also, this sample just illustrates the use of some of the methods in the `formContext.data.process` API. It doesn't represent using this API to meet a business requirement; it's only intended to demonstrate how the key property values can be accessed in code.

JavaScript

```
//A namespace defined for SDK sample code
//You should define a unique namespace for your libraries
var Sdk = window.Sdk || {};
(function () {
    //A global variable to store information about enabled business
    processes after they are retrieved asynchronously
    this.enabledProcesses = [];

    // A function to log messages while debugging only
    this.writeToConsole = function (message) {
        if (typeof console != 'undefined')
        { console.log(message); }
    };

    // Code to run in the OnLoad event
    this.formOnLoad = function (executionContext) {
        // Retrieve the formContext
        var formContext = executionContext.getFormContext();

        // Retrieve Enabled processes
        formContext.data.process.getEnabledProcesses(function (processes) {
            //Move processes to the global Sdk.enabledProcesses array;
            for (var processId in processes) {
                Sdk.enabledProcesses.push({ id: processId, name:
processes[processId] })
            }
            Sdk.writeToConsole("Enabled business processes flows retrieved
and added to Sdk.enabledProcesses array.");
        });

        //Write the values of the Sdk.enabledProcesses array to the
        console
        if (Sdk.enabledProcesses.length < 0) {
            Sdk.writeToConsole("There are no enabled business process
flows for this table.");
        }
    };
});
```

```

        }
        else {
            Sdk.writeToConsole("These are the enabled business process
flows for this table:");
            for (var i = 0; i < Sdk.enabledProcesses.length; i++) {
                var enabledProcess = Sdk.enabledProcesses[i];
                Sdk.writeToConsole("id: " + enabledProcess.id + " name:
" + enabledProcess.name)
            }
        }

        //Any code that depends on the Sdk.enabledProcesses array needs
to be initiated here
    });

};

}).call(Sdk);

```

When you run this sample with the browser developer tools open, the following is an example of the output written to the console for a table with multiple business process flows enabled.

```

Enabled business processes flows retrieved and added to Sdk.enabledProcesses
array.
These are the enabled business process flows for this table:
id: 7994be68-899e-4a40-8d18-f5c3b6940188 name: Sample Lead Process
id: 919e14d1-6489-4852-abd0-a63a6ecaac5d name: Lead to Opportunity Sales
Process

```

Related topics

[setActiveProcessInstance](#)

[formContext.data.process](#)

process.getId (Client API reference)

Article • 11/30/2022

Returns the unique identifier of the process.

Syntax

```
var processId = procObj.getId();
```

Return Value

Type: String.

Description: Value represents the string representation of a GUID value.

Related topics

[formContext.data.process](#)

process.getName (Client API reference)

Article • 11/30/2022

Returns the name of the process.

Syntax

```
var processName = procObj.getName();
```

Return Value

Type: String.

Description: Name of the process.

Related topics

[formContext.data.process](#)

getStages (Client API reference)

Article • 11/30/2022

Returns a collection of stages in the process.

Syntax

```
var stageCollection = procObj.getStages();
```

Returns

Type: Collection.

Description: See [Stage methods](#) for the methods to access the properties of the stages returned.

Related topics

[formContext.data.process](#)

isRendered (Client API reference)

Article • 11/30/2022

Returns a boolean value indicating whether the process is rendered.

Syntax

```
var processRendered = procObj.isRendered();
```

Returns

Type: Boolean.

Description: true if the process is rendered; false otherwise.

Related topics

[formContext.data.process](#)

getProcessInstances (Client API reference)

Article • 11/30/2022

Returns all the process instances for the table record that the calling user has access to.

Syntax

```
formContext.data.process.getProcessInstances(callbackFunction(object));
```

Parameter

Name	Type	Required	Description
callbackFunction	Function	Yes	<p>The callback function is passed an object with the following columns and their corresponding values as the key: value pair. All returned values are of String type except for CreatedOnDate, which is of Date ↗ type.</p> <ul style="list-style-type: none">- CreatedOn (deprecated)- CreatedOnDate- ProcessDefinitionID- ProcessDefinitionName- ProcessInstanceID- ProcessInstanceName- StatusCodeName <p>The process instances are filtered according to the user's privileges.</p>

Related topics

[setActiveProcessInstance](#)

[formContext.data.process](#)

setActiveProcessInstance (Client API reference)

Article • 11/30/2022

Sets a process instance as the active instance.

Syntax

```
formContext.data.process.setActiveProcessInstance(processInstanceId,  
callbackFunction);
```

Parameter

Name	Type	Required	Description
processInstanceId	String	Yes	The Id of the process instance to set as the active instance.
callbackFunction	Function	No	A function to call when the operation is complete. This callback function is passed one of the following string values to indicate whether the operation succeeded: <ul style="list-style-type: none">- success: The operation succeeded.- invalid: The processInstanceId isn't valid or the process isn't enabled.

Related topics

[getProcessInstances](#)

[formContext.data.process](#)

getInstanceId (Client API reference)

Article • 11/30/2022

Returns the unique identifier of the process instance.

Syntax

```
formContext.data.process.getInstanceId();
```

Return Value

Type: String.

Description: Value represents the string representation of a GUID value.

Related topics

[formContext.data.process](#)

getINSTANCEName (Client API reference)

Article • 11/30/2022

Returns the name of the process instance.

Syntax

```
formContext.data.process.getINSTANCEName();
```

Return Value

Type: String.

Description: Process instance name.

Related topics

[formContext.data.process](#)

process.getStatus (Client API reference)

Article • 11/30/2022

Returns the current status of the process instance.

Syntax

```
formContext.data.process.getStatus();
```

Return Value

Type: String.

Description: Returns one of the following values: **active**, **aborted**, or **finished**.

Related topics

[formContext.data.process](#)

setStatus (Client API reference)

Article • 11/30/2022

Sets the current status of the active process instance.

Syntax

```
formContext.data.process.setStatus(status, callbackFunction);
```

Parameters

Name	Type	Required	Description
status	String	Yes	The new status. The values can be active , aborted , finished , or invalid .
callbackFunction	Function	No	A function to call when the operation is complete. This callback function is passed the new status as a string value.

Type: String.

Description: Returns one of the following values: **active**, **aborted**, or **finished**. It returns **invalid** if the setStatus API fails.

Related topics

[formContext.data.process](#)

getActiveStage (Client API reference)

Article • 11/30/2022

Returns a **Stage** object representing the active stage.

Syntax

```
formContext.data.process.getActiveStage();
```

Return Value

Type: Stage.

Description: The currently active stage. See [Stage methods](#) for the methods to access the properties of the stage returned.

Related topics

[setActiveStage](#)

[getSelectedStage \(Client API reference\)](#)

[formContext.data.process](#)

setActiveStage (Client API reference)

Article • 11/30/2022

Sets a completed stage as the active stage.

Syntax

```
formContext.data.process.setActiveStage(stageId, callbackFunction);
```

Parameters

Name	Type	Required	Description												
stageId	String	Yes	The ID of the completed stage for the table to make the active stage.												
callbackFunction	Function	No	A function to call when the operation is complete. This callback function is passed one of the following string values to indicate the status of the operation: <table border="1"><thead><tr><th>Value</th><th>Reason</th></tr></thead><tbody><tr><td>success</td><td>The operation succeeded.</td></tr><tr><td>invalid</td><td>There are three reasons why this value may be returned:<ul style="list-style-type: none">The *stageId* parameter is a non-existent stage ID value.The active stage isn't the selected stage.The record hasn't been saved yet.</td></tr><tr><td>unreachable</td><td>The stage exists on a different path.</td></tr><tr><td>dirtyForm</td><td>This value will be returned if the data in the page is not saved.</td></tr><tr><td>preventDefault</td><td>This value will be returned if an `OnPreStageChange` event handler invokes preventDefault.</td></tr></tbody></table>	Value	Reason	success	The operation succeeded.	invalid	There are three reasons why this value may be returned: <ul style="list-style-type: none">The *stageId* parameter is a non-existent stage ID value.The active stage isn't the selected stage.The record hasn't been saved yet.	unreachable	The stage exists on a different path.	dirtyForm	This value will be returned if the data in the page is not saved.	preventDefault	This value will be returned if an `OnPreStageChange` event handler invokes preventDefault.
Value	Reason														
success	The operation succeeded.														
invalid	There are three reasons why this value may be returned: <ul style="list-style-type: none">The *stageId* parameter is a non-existent stage ID value.The active stage isn't the selected stage.The record hasn't been saved yet.														
unreachable	The stage exists on a different path.														
dirtyForm	This value will be returned if the data in the page is not saved.														
preventDefault	This value will be returned if an `OnPreStageChange` event handler invokes preventDefault.														

Important

This method can only be used when the selected stage and the active stage are the same. When your code is initiated from the **OnStageChange** event, the current stage will be selected. When your code is initiated from the **OnStageSelected** event, you should use the **getActiveStage** method to verify that the selected stage is also the active stage. For any other form event, it isn't possible to determine which stage is currently selected. For best results, this method should only be used in code that is called in functions initiated by the **OnStageChange** and **OnStageSelected** events.

Related topics

[getActiveStage](#)

[formContext.data.process](#)

getActivePath (Client API reference)

Article • 11/30/2022

Gets a collection of stages currently in the active path with methods to interact with the stages displayed in the business process flow control.

The active path represents stages currently rendered in the process control based on the branching rules and current data in the record.

Syntax

```
var stageCollection = formContext.data.process.getActivePath();
```

Return Value

Type: Collection.

Description: A collection of all completed stages, the currently active stage, and the predicted set of future stages based on satisfied conditions in the branching rule. This may be a subset of the stages returned with [formContext.data.process.getActiveProcess](#) because it will only include those stages which represent a valid transition from the current stage based on branching that has occurred in the process.

Example

The Sdk.formOnLoad function uses the [formContext.data.process.getActivePath](#) method to retrieve a collection of stages. Then, the sample code uses the [forEach](#) method of the collection to loop through each stage. The code then writes key properties of the stage to the console using the [Sdk.writeToConsole](#) function defined in this library. The code then accesses a collection of steps for each stage using the [getSteps](#) method. Finally, the sample uses the [forEach](#) method of the steps collection to access each step and write key properties of the step to the console.

Note

The [Sdk.formOnLoad](#) function in the sample JavaScript library must be set as the **OnLoad** event handler for a form, and the **Pass execution context as the first parameter** check box must be selected in the **Handler Properties** dialog. Also, this sample just illustrates the use of some of the methods in the

`formContext.data.process` API. It doesn't represent using this API to meet a business requirement; it's only intended to demonstrate how the key property values can be accessed in code.

JavaScript

```
// A namespace defined for SDK sample code
// You should define a unique namespace for your libraries
var Sdk = window.Sdk || {};
(function () {

    // A function to log messages while debugging only
    this.writeToConsole = function (message) {
        if (typeof console != 'undefined')
            { console.log(message); }
    };

    // Code to run in the OnLoad event
    this.formOnLoad = function (executionContext) {
        // Retrieve the formContext
        var formContext = executionContext.getFormContext();

        // Enumerate the stages and steps in the active path
        var activePathCollection = formContext.data.process.getActivePath();
        activePathCollection.forEach(function (stage, n) {
            Sdk.writeToConsole("Stage Index: " + n);
            Sdk.writeToConsole("Table: " + stage.getEntityName());
            Sdk.writeToConsole("StageId: " + stage.getId());
            Sdk.writeToConsole("Status: " + stage.getStatus());
            var stageSteps = stage.getSteps();
            stageSteps.forEach(function (step, i) {
                Sdk.writeToConsole("      Step Name: " + step.getName());
                Sdk.writeToConsole("      Step Column: " +
step.getAttribute());
                Sdk.writeToConsole("      Step Required: " +
step.isRequired());
                Sdk.writeToConsole("-----")
            })
            Sdk.writeToConsole("-----")
        });
    };
}).call(Sdk);
```

When the sample runs in the browser, you can use the developer tools of the browser to view the text written to the console. For example, when this sample is run in the Opportunity form with the Opportunity Sales Process, the following is written to the console:

Stage Index: 0
Table: opportunity
StageId: 6b9ce798-221a-4260-90b2-2a95ed51a5bc
Status: active

Step Name: Identify Contact
Step Column: parentcontactid
Step Required: false

Step Name: Identify Account
Step Column: parentaccountid
Step Required: false

Step Name: Purchase Timeframe
Step Column: purchasetimeframe
Step Required: false

Step Name: Estimated Budget
Step Column: budgetamount
Step Required: false

Step Name: Purchase Process
Step Column: purchaseprocess
Step Required: false

Step Name: Identify Decision Maker
Step Column: decisionmaker
Step Required: false

Step Name: Capture Summary
Step Column: description
Step Required: false

Stage Index: 1
Table: opportunity
StageId: 650e06b4-789b-46c1-822b-0da76bedb1ed
Status: inactive

Step Name: Customer Need
Step Column: customerneed
Step Required: false

Step Name: Proposed Solution
Step Column: proposedsolution
Step Required: false

Step Name: Identify Stakeholders
Step Column: identifycustomercontacts
Step Required: false

Step Name: Identify Competitors
Step Column: identifycompetitors
Step Required: false

```
Stage Index: 2
Table: opportunity
StageId: d3ca8878-8d7b-47b9-852d-fcd838790cf
Status: inactive
Step Name: Identify Sales Team
Step Column: identifypursuitteam
Step Required: false
-----
Step Name: Develop Proposal
Step Column: developproposal
Step Required: false
-----
Step Name: Complete Internal Review
Step Column: completeinternalreview
Step Required: false
-----
Step Name: Present Proposal
Step Column: presentproposal
Step Required: false
-----
Stage Index: 3
Table: opportunity
StageId: bb7e830a-61bd-441b-b1fd-6bb104ffa027
Status: inactive
Step Name: Complete Final Proposal
Step Column: completestinalproposal
Step Required: false
-----
Step Name: Present Final Proposal
Step Column: presentfinalproposal
Step Required: false
-----
Step Name: Confirm Decision Date
Step Column: finaldecisiondate
Step Required: false
-----
Step Name: Send Thank You
Step Column: sendthankyounote
Step Required: false
-----
Step Name: File De-brief
Step Column: filedebrief
Step Required: false
-----
```

Related topics

[formContext.data.process](#)

getSelectedStage (Client API reference)

Article • 11/30/2022

Gets the currently selected stage.

Syntax

```
formContext.data.process.getSelectedStage();
```

Return Value

Type: Stage.

Description: The currently selected stage. See [Stage methods](#) for the methods to access the properties of the stage returned.

Related topics

[getActiveStage \(Client API reference\)](#)

[formContext.data.process](#)

getCategory (Client API reference)

Article • 11/30/2022

Returns an object with a `getValue` method which will return the integer value of the business process flow category.

Syntax

```
var stageCategoryNumber = stageObj.getCategory().getValue();
```

Return Value

Type: Number.

Description: Here is the list of possible values.

Value	Description
0	Qualify
1	Develop
2	Propose
3	Close
4	Identify
5	Research
6	Resolve

Related topics

[formContext.data.process](#)

stage.getEntityName (Client API reference)

Article • 11/30/2022

Returns the logical name of the table associated with the stage.

Syntax

```
var stageEntityName = stageObj.getEntityName();
```

Return Value

Type: String.

Description: Logical name of the table associated with the stage.

Related topics

[formContext.data.process](#)

stage.getId (Client API reference)

Article • 11/30/2022

Returns the unique identifier of the stage.

Syntax

```
var stageId = stageObj.getId();
```

Returns

Type: String.

Description: Unique identifier of the stage in the GUID format.

Related topics

[formContext.data.process](#)

stage.getName (Client API reference)

Article • 11/30/2022

Returns the name of the stage.

Syntax

```
var stageName = stageObj.getName();
```

Return Value

Type: String.

Description: Name of the stage.

Related topics

[formContext.data.process](#)

getNavigationBehavior (Client API reference)

Article • 11/30/2022

Returns a navigation behavior object for a stage that can be used to define whether the **Create** button is available for users to create other table record in a cross-table business process flow navigation scenario.

ⓘ Note

This method is available only for **Unified Interface**.

Syntax

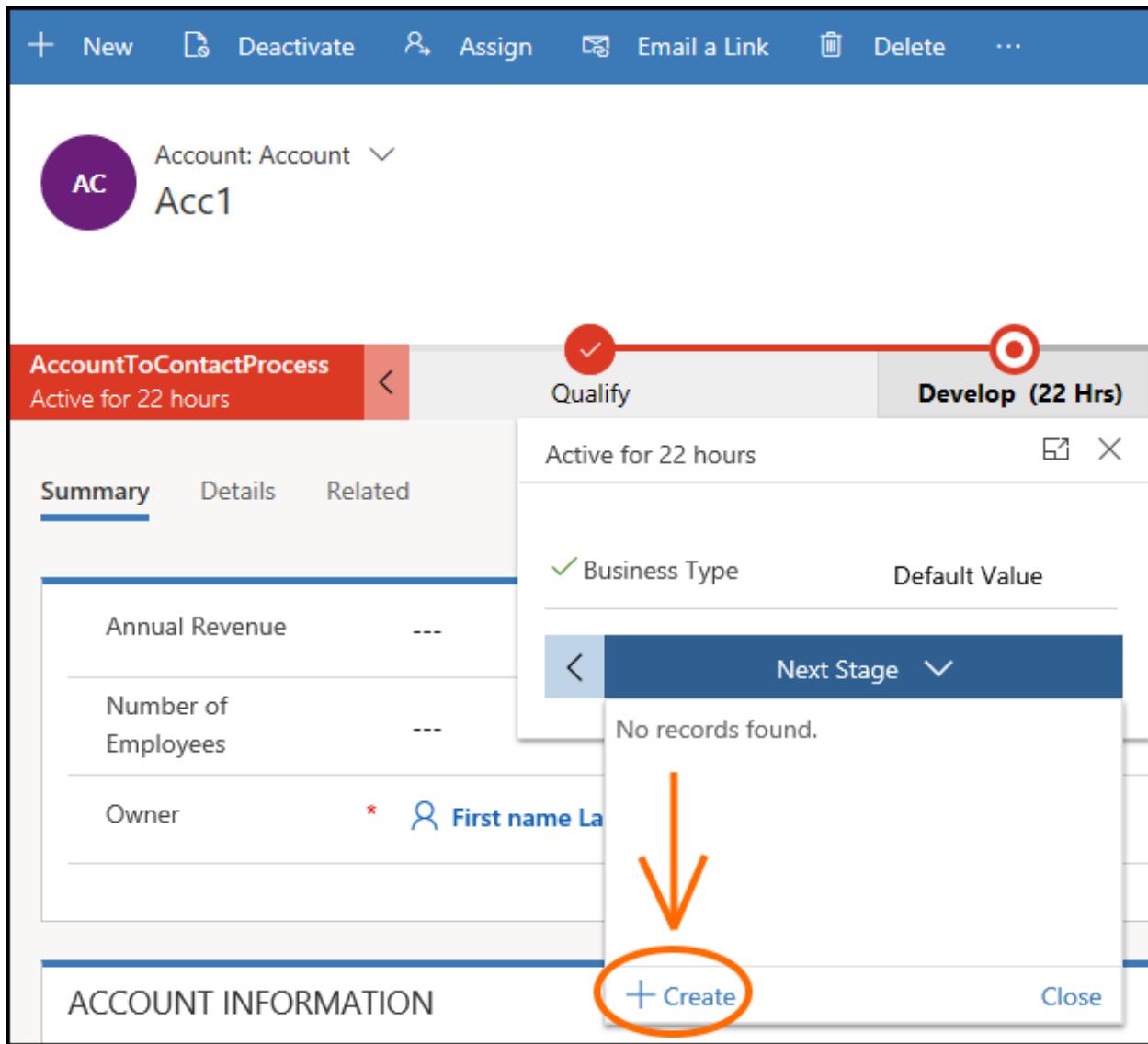
```
stageObj.getNavigationBehavior().allowCreateNew = function () {  
    return true|false;  
}
```

Returns

Type: Object

Description: An object with the `allowCreateNew` property that lets you define whether the **Create** button will be available in a stage so that user can create an instance of tableB from the tableA form in a cross-table business process flow navigation scenario.

For example, here is the **Create** button in the **Develop** stage of the **AccountToContactProcess** sample business process flow that lets you create a Contact record from the Account form.



The `allowCreateNew` property will return `undefined` for business process flow records that do not implement cross-table navigation.

Example

The following sample code shows how you can hide or display the **Create** button for an active stage of a business process flow depending on its name.

JavaScript

```
function sampleFunction(executionContext) {
    var formContext = executionContext.getFormContext();

    formContext.data.process.getActiveStage().getNavigationBehavior().allowCreateNew = function () {
        if (formContext.data.process.getName() === 'Test Process') {
            return false; // Create button is not available
        }
        else {
            return true; // Create button is available
        }
    }
}
```

```
        }  
    }  
}
```

Related topics

[formContext.data.process](#)

stage.getStatus (Client API reference)

Article • 11/30/2022

Returns the status of the stage.

Syntax

```
var stageStatus = stageObj.getStatus();
```

Returns

Type: String.

Description: This method will return either **active** or **inactive**.

Related topics

[formContext.data.process](#)

getSteps (Client API reference)

Article • 11/30/2022

Returns a collection of steps in the stage.

Syntax

```
var stepsCollection = stageObj.getSteps();
```

Return Value

Type: Array.

Description: See [Step methods](#) for methods to access the property values of the step.

Related topics

[formContext.data.process](#)

step.getAttribute (Client API reference)

Article • 11/30/2022

Returns the logical name of the column associated to the step.

Syntax

```
var stepAttributeName = stepObj.getAttribute();
```

Returns

Type: String.

Description: Some steps don't contain a column value.

Related topics

[formContext.data.process](#)

step.getName (Client API reference)

Article • 11/30/2022

Returns the name of the step.

Syntax

```
var stepName = stepObj.getName();
```

Return Value

Type: String.

Description: Name of the step.

Related topics

[formContext.data.process](#)

getProgress (Client API reference)

Article • 11/30/2022

Returns the progress of the action step.

Syntax

```
var stepProgress = stepObj.getProgress();
```

Return Value

Type: Number.

Description: Returns one of the following values:

Value	Description
0	None
1	Processing
2	Completed
3	Failure
4	Invalid

Remarks

This method is supported only for the action steps; not for the data steps. Action steps are buttons on the business process stages that users can click to trigger an on-demand workflow or action. Action step is a preview feature introduced in the v9.0 release. More information: See the **Business Process Flow automation with Action Steps** section in [Blog: New automation and visualization features for Business Process Flows \(public preview\)](#) ↗

Related topics

[setProgress](#)

[formContext.data.process](#)

isRequired (Client API reference)

Article • 11/30/2022

Returns a boolean value indicating whether the step is required in the business process flow.

Syntax

```
var stepIsRequired = stepObj.isRequired();
```

Returns

Type: Boolean.

Description: true if the step is marked as required in the Business Process Flow editor; false otherwise.

Related topics

[formContext.data.process](#)

setProgress (Client API reference)

Article • 11/30/2022

Updates the progress of the action step.

Syntax

```
stepObj.setProgress(stepProgress,message);
```

Parameters

Name	Type	Required	Description
stepProgress	Number	Yes	<p>Specify one of the following values to specify the step progress:</p> <ul style="list-style-type: none">• 0: None• 1: Processing• 2: Completed• 3: Failure• 4: Invalid
message	String	No	An optional message that is set as the Alt text on the icon for the step.

Return Value

Type: String.

Description: Returns "invalid" or "success" depending on whether the step progress was updated.

Remarks

This method is supported only for the action steps. Action steps are buttons on the business process stages that users can click to trigger an on-demand workflow or action. Action step is a preview feature introduced in the v9.0 release. More information: See the [Business Process Flow automation with Action Steps](#) section in [Blog: New automation and visualization features for Business Process Flows \(public preview\)](#).

Related topics

[getProgress](#)

[formContext.data.process](#)

moveNext (Client API reference)

Article • 11/30/2022

Progresses to the next stage.

Moving to next stage is not supported for different table.

Syntax

```
formContext.data.process.moveNext(callbackFunction);
```

Parameters

Name	Type	Required	Description														
callbackFunction	Function	No	A function to call when the operation is complete. This callback function is passed one of the following string values to indicate the status of the operation: <table border="1"><thead><tr><th>Value</th><th>Reason</th></tr></thead><tbody><tr><td>success</td><td>The operation succeeded.</td></tr><tr><td>crossEntity</td><td>The next stage is for a different table.</td></tr><tr><td>end</td><td>The active stage is the last stage of the active path.</td></tr><tr><td>invalid</td><td>The operation failed because the selected stage isn't the same as the active stage.</td></tr><tr><td>dirtyForm</td><td>This value will be returned if the data in the page is not saved.</td></tr><tr><td>stageGate</td><td>One or more required column on the current stage is empty.</td></tr></tbody></table>	Value	Reason	success	The operation succeeded.	crossEntity	The next stage is for a different table.	end	The active stage is the last stage of the active path.	invalid	The operation failed because the selected stage isn't the same as the active stage.	dirtyForm	This value will be returned if the data in the page is not saved.	stageGate	One or more required column on the current stage is empty.
Value	Reason																
success	The operation succeeded.																
crossEntity	The next stage is for a different table.																
end	The active stage is the last stage of the active path.																
invalid	The operation failed because the selected stage isn't the same as the active stage.																
dirtyForm	This value will be returned if the data in the page is not saved.																
stageGate	One or more required column on the current stage is empty.																

ⓘ Important

This method can only be used when the selected stage and the active stage are the same. When your code is initiated from the [OnStageChange](#) event, the current stage will be selected. When your code is initiated from the [OnStageSelected](#)

event, you should use the [getActiveStage](#) method to verify that the selected stage is also the active stage. For any other form event, it isn't possible to determine which stage is currently selected. For best results, this method should only be used in code that is called in functions initiated by the [OnStageChange](#) and [OnStageSelected](#) events.

Remarks

This methods will cause the [OnStageChange](#) event to occur.

Related topics

[movePrevious](#)

[formContext.data.process](#)

movePrevious (Client API reference)

Article • 11/30/2022

Moves to the previous stage.

You can also move to a previous stage in a different table.

Syntax

```
formContext.data.process.movePrevious(callbackFunction);
```

Parameters

Name	Type	Required	Description																
callbackFunction	Function	No	A function to call when the operation is complete. This callback function is passed one of the following string values to indicate the status of the operation:																
<table><thead><tr><th>Value</th><th>Reason</th></tr></thead><tbody><tr><td>success</td><td>The operation succeeded.</td></tr><tr><td>crossEntity</td><td>The previous stage is for a different table.</td></tr><tr><td>beginning</td><td>The active stage is the first stage of the active path.</td></tr><tr><td>invalid</td><td>The operation failed because the selected stage isn't the same as the active stage.</td></tr><tr><td>dirtyForm</td><td>This value will be returned if the data in the page is not saved.</td></tr><tr><td>stageGate</td><td>One or more required column on the current stage is empty.</td></tr><tr><td>preventDefault</td><td>This value will be returned if an `OnPreStageChange` event handler invokes preventDefault.</td></tr></tbody></table>				Value	Reason	success	The operation succeeded.	crossEntity	The previous stage is for a different table.	beginning	The active stage is the first stage of the active path.	invalid	The operation failed because the selected stage isn't the same as the active stage.	dirtyForm	This value will be returned if the data in the page is not saved.	stageGate	One or more required column on the current stage is empty.	preventDefault	This value will be returned if an `OnPreStageChange` event handler invokes preventDefault.
Value	Reason																		
success	The operation succeeded.																		
crossEntity	The previous stage is for a different table.																		
beginning	The active stage is the first stage of the active path.																		
invalid	The operation failed because the selected stage isn't the same as the active stage.																		
dirtyForm	This value will be returned if the data in the page is not saved.																		
stageGate	One or more required column on the current stage is empty.																		
preventDefault	This value will be returned if an `OnPreStageChange` event handler invokes preventDefault.																		

 **Important**

This method can only be used when the selected stage and the active stage are the same. When your code is initiated from the [OnStageChange](#) event, the current stage will be selected. When your code is initiated from the [OnStageSelected](#) event, you should use the [getActiveStage](#) method to verify that the selected stage is also the active stage. For any other form event, it isn't possible to determine which stage is currently selected. For best results, this method should only be used in code that is called in functions initiated by the [OnStageChange](#) and [OnStageSelected](#) events.

Remarks

This methods will cause the [OnStageChange](#) event to occur.

Related topics

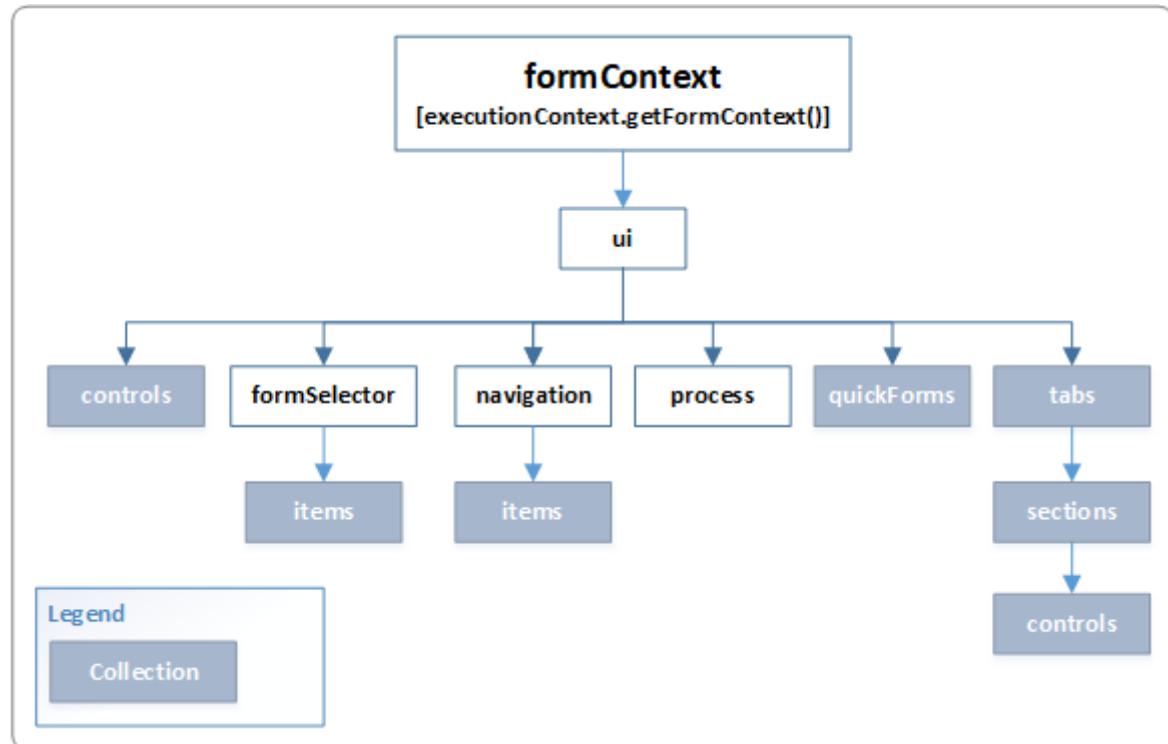
[moveNext](#)

[formContext.data.process](#)

formContext.ui (Client API reference)

Article • 11/30/2022

Provides properties and methods to retrieve information about the user interface (UI) as well as collections for several sub components of the form.



Properties

Name	Description
controls	Collection of all the controls on the page. See Collections for information about the collections and Controls for information about the control objects in the collection.
formSelector	Use the <code>formSelector.getCurrentItem</code> method to retrieve information about the form currently in use. Use the <code>formSelector.items</code> collection to return information about all the forms available for the user. <code>formSelector</code> is not available for Microsoft Dynamics 365 for tablets.
navigation	An object containing a property <code>items</code> , which is a collection of all the navigation items on the page. See Collections for information about the collection methods and formContext.ui.navigation item for information about the items in the collection. <code>navigation</code> is not available for Microsoft Dynamics 365 for tablets.
process	Provides objects and methods to interact with the business process flow control on a form. More information: formContext.ui.process

Name	Description
quickForms	A collection of all the quick view controls on a form using the new form rendering engine (also called "turbo forms"). More information: formContext.ui.quickForms
tabs	A collection of all the tabs on the page. See Collections for information about the collection methods and formContext.ui.tab for information about the items in the collection.

Methods

Name	Description
addOnLoad	Adds a function to be called on the form OnLoad event.
clearFormNotification	Removes form level notifications.
close	Closes the form.
getFormType	Gets the form type for the record.
getViewPortHeight	Gets the height of the viewport in pixels.
getViewPortWidth	Get the width of the viewport in pixels.
refreshRibbon	Causes the ribbon to re-evaluate data that controls what is displayed in it.
removeOnLoad	Removes a function from the form OnLoad event.
setFormEntityName	Sets the name of the table to be displayed on the form.
setFormNotification	Displays form level notifications.

formContext.ui.footerSection removed

The `formContext.ui.footerSection` was removed in October of 2021. More information: [Form footers in model-driven apps won't be supported with the 2021 release wave 2](#)

Related topics

[formContext](#)

ui.addOnLoad (Client API reference)

Article • 11/30/2022

Adds a function to be called on the form [OnLoad](#) event.

Syntax

```
formContext.ui.addOnLoad(myFunction)
```

Parameter

Name	Type	Required	Description
myFunction	function reference	Yes	The function to be executed on the form OnLoad event. The function will be added to the bottom of the event handler pipeline. The execution context is automatically passed as the first parameter to the function. See Execution context for more information.

Related topics

[removeOnLoad](#)

[Form OnLoad event](#)

[formContext.ui](#)

[formContext](#)

ui.clearFormNotification (Client API reference)

Article • 11/30/2022

Removes form level notifications.

Syntax

```
formContext.ui.clearFormNotification(uniqueId)
```

Parameter

Name	Type	Required	Description
uniqueId	String	Yes	A unique identifier for the message to be cleared that was set using the setFormNotification method.

Return Value

Type: Boolean

Description: true if the method succeeded, false otherwise.

Related topics

[setFormNotification](#)

[formContext.ui](#)

[formContext](#)

close (Client API reference)

Article • 11/30/2022

Closes the form.

Syntax

```
formContext.ui.close();
```

Remarks

The HTML `Window.close` method is suppressed. To close a form window, you must use this method. If there are any unsaved changes in the form, the user will be prompted whether they want to save their changes before the window closes.

For Microsoft Dynamics 365 for tablets, this method mimics the behavior of the back navigation button.

Related topics

[formContext.ui](#)

[formContext](#)

getFormType (Client API reference)

Article • 11/30/2022

Gets the form type for the record.

Syntax

```
formContext.ui.getFormType();
```

Return Value

Type: Number

Description: Form type. Returns one of the following values

Value	Form type
0	Undefined
1	Create
2	Update
3	Read Only
4	Disabled
6	Bulk Edit

ⓘ Note

Quick Create forms return 1.

Related topics

[formContext.ui](#)

[formContext](#)

getViewPortHeight (Client API reference)

Article • 11/30/2022

Gets the height of the viewport in pixels.

The viewport is the area of the page containing form data. It corresponds to the body of the form and does not include the navigation, header, footer or form assistant areas of the page.

Syntax

```
formContext.ui.getViewPortHeight();
```

Return Value

Type: Number

Description: The viewport height in pixels.

Related topics

[getViewPortWidth](#)

[formContext.ui](#)

[formContext](#)

getViewPortWidth (Client API reference)

Article • 11/30/2022

Get the width of the viewport in pixels.

The viewport is the area of the page containing form data. It corresponds to the body of the form and does not include the navigation, header, footer or form assistant areas of the page.

Syntax

```
formContext.ui.getViewPortWidth();
```

Return Value

Type: Number

Description: The viewport width in pixels.

Related topics

[getViewPortHeight](#)

[formContext.ui](#)

[formContext](#)

ui.refreshRibbon (Client API reference)

Article • 11/30/2022

Causes the ribbon to re-evaluate data that controls what is displayed in it.

Syntax

```
formContext.ui.refreshRibbon(refreshAll);
```

Parameter

Name	Type	Required	Description
refreshAll	Boolean	No	Indicates whether all the ribbon command bars on the current page are refreshed. If you specify false , only the page-level ribbon command bar is refreshed. If you do not specify this parameter, by default false is passed.

Remarks

This function is used when a ribbon action `JavaScriptFunction` (`RibbonDiffXml`) changes the data in the form. For example, changing of state of the record via a ribbon action. After your code changes the data that is used by a rule, use this method to force the ribbon to reevaluate the data in the form so that the rule can be reapplied.

Guidance

For optimal performance of your form loads, you should not use this function in `EnableRule` (`RibbonDiffXml`) or `onLoad` (`FormXml`). The form load itself triggers rules evaluation of all the ribbon actions. In case, if you want to control the visibility of a ribbon action, use promises and asynchronous pattern in `EnableRule`.

Related topics

[formContext.ui](#)

[formContext](#)

ui.removeOnLoad (Client API reference)

Article • 11/30/2022

Removes a function from the form [OnLoad](#) event.

Syntax

```
formContext.ui.removeOnLoad(myFunction)
```

Parameter

Name	Type	Required	Description
myFunction	function reference	Yes	The function to be removed from the form OnLoad event.

Related topics

[addOnLoad](#)

[Form data OnLoad event](#)

[formContext.ui](#)

[formContext](#)

setFormEntityName (Client API reference)

Article • 11/30/2022

Sets the name of the table to be displayed on the form.

Syntax

```
formContext.ui.setFormEntityName(arg);
```

Parameter

Name	Type	Required	Description
arg	String	Yes	Name of the table to be displayed on the form.

Related topics

[formContext.ui](#)

[formContext](#)

setFormNotification (Client API reference)

Article • 11/30/2022

Displays form level notifications.

You can display any number of notifications and they will be displayed until they are removed using [clearFormNotification](#). The height of the notification area is limited so each new message will be added to the top. Users can scroll down to view older messages that have not yet been removed.

Syntax

```
formContext.ui.setFormNotification(message, level, uniqueId);
```

Parameter

Name	Type	Required	Description
message	String	Yes	The text of the message.
level	String	Yes	The level of the message, which defines how the message will be displayed. Specify one of the following values: <code>ERROR</code> : Notification will use the system error icon. <code>WARNING</code> : Notification will use the system warning icon. <code>INFO</code> : Notification will use the system info icon.
uniqueId	String	Yes	A unique identifier for the message that can be used later with clearFormNotification to remove the notification.

Return Value

Type: Boolean

Description: true if the method succeeded; false otherwise.

Related topics

[clearFormNotification](#)

[formContext.ui](#)

formContext

formContext.ui.formSelector (Client API reference)

Article • 11/30/2022

The `formContext.ui.formSelector` property lets you work with form items where a form item represents a form that is available to a user because it is associated with a security role that the user is also associated to. Often there will be only one form. When more than one form is available, methods for a form item can be used to change the form the user is viewing.

ⓘ Note

The `formContext.ui.formSelector` is not supported for quick create forms.

ⓘ Note

The form selector is not visible if the user only has access to one main form

Form Items are available through any of the following:

- **formselector.items** collection: A collection of all the form items accessible to the current user. Only those forms that share an association with one of the user's security roles are available in this collection. Example:

```
formItem = formContext.ui.formSelector.items.get(arg);
```

See [Collections](#) for information about the collection methods.

ⓘ Note

This collection isn't available for **Dynamics 365 mobile clients (phones and tablets)**.

- **formselector.getCurrentItem** method: Returns a reference to the form currently being shown. When only one form is available this method will return **null**. Example:

```
formItem = formContext.ui.formSelector.getCurrentItem();
```

Form Item methods

After retrieving a form item using one of the above ways, use the following methods to work with the form item.

Name	Description
getId	Returns the ID of the form.
getLabel	Returns the label of the form.
getVisible	Returns a value that indicates whether the form is currently visible.
navigate	Opens the specified form.
setVisible	Sets a value that indicates whether the form is visible.

formItem.getId (Client API reference)

Article • 11/30/2022

Returns the ID of the form.

Syntax

```
formItem.getId();
```

Return Value

Type: String.

Description: ID of the form.

Related topics

[formContext.ui.formSelector](#)

formItem.getLabel (Client API reference)

Article • 11/30/2022

Returns the label of the form.

Syntax

```
formItem.getLabel();
```

Return Value

Type: String.

Description: Label of the form.

Related topics

[formContext.ui.formSelector](#)

formItem.getVisible (Client API reference)

Article • 11/30/2022

Returns a value that indicates whether the form is currently visible.

Syntax

```
formItem.getVisible();
```

Return Value

Type: Boolean.

Description: true if the form is visible; false otherwise.

Related topics

[setVisible](#)

navigate (Client API reference)

Article • 11/30/2022

Opens the specified form.

When you use the `navigate` method while unsaved changes exist, the user is prompted to save changes before the new form can be displayed. The **Onload** event occurs when the new form loads.

Syntax

```
formItem.navigate();
```

Related topics

[formContext.ui.formSelector](#)

formItem.setVisible (Client API reference)

Article • 11/30/2022

Sets a value that indicates whether the form is visible.

Syntax

```
formItem.setVisible(bool);
```

Parameter

Name	Type	Required	Description
bool	Boolean	Yes	Specify true to show the form; false to hide the form.

Related topics

[getVisible](#)

formContext.ui.headerSection item (Client API reference)

Article • 08/04/2023

Provides information on how to set the visibility of header section.

ⓘ Note

This API is not available for Dynamics 365 Customer Engagement on-premise deployments. More information: [Client APIs not supported in Customer Engagement \(on-premises\)](#)

Methods

Name	Description
getBodyVisible	Returns the header's body visibility.
getCommandBarVisible	Returns the command bar visibility.
getTabNavigatorVisible	Returns the tab navigator visibility.
setBodyVisible	Sets the header's body visibility.
setCommandBarVisible	Sets the command bar visibility.
setTabNavigatorVisible	Sets the tab navigator visibility.

Related topics

[Client API Xrm object](#)

getBodyVisible (Client API reference)

Article • 11/30/2022

Returns the header's body visibility.

ⓘ Note

This method is supported only on Unified Interface.

Syntax

```
formContext.ui.headerSection.getBodyVisible();
```

Return Value

Type: Boolean

Description: **true** if visible; **false** otherwise.

Related topics

[setBodyVisible](#)

getCommandBarVisible (Client API reference)

Article • 11/30/2022

Returns the command bar visibility.

ⓘ Note

This method is supported only on Unified Interface.

Syntax

```
formContext.ui.headerSection.getCommandBarVisible();
```

Return Value

Type: Boolean

Description: **true** if the command bar is visible; **false** otherwise.

Related topics

[setCommandBarVisible](#)

getTabNavigatorVisible (Client API reference)

Article • 11/30/2022

Returns the tab navigator visibility.

ⓘ Note

This method is supported only on Unified Interface.

Syntax

```
formContext.ui.headerSection.getTabNavigatorVisible();
```

Return Value

Type: Boolean

Description: **true** if the tab navigation is visible; **false** otherwise.

Related topics

[setTabNavigatorVisible](#)

setBodyVisible (Client API reference)

Article • 11/30/2022

Sets the header's body visibility.

ⓘ Note

This method is supported only on Unified Interface.

Syntax

```
formContext.ui.headerSection.setBodyVisible(bool);
```

Parameter

Name	Type	Required	Description
bool	Boolean	Yes	Specify true to show the body; false to hide the body.

Related topics

[formContext.ui.headerSection](#)

setCommandBarVisible (Client API reference)

Article • 11/30/2022

Sets the command bar visibility.

ⓘ Note

This method is supported only on Unified Interface.

Syntax

```
formContext.ui.headerSection.setCommandBarVisible(bool);
```

Parameter

Name	Type	Required	Description
bool	Boolean	Yes	Specify true to show the command bar; false to hide the command bar.

Related topics

[formContext.ui.headerSection](#)

setTabNavigatorVisible (Client API reference)

Article • 11/30/2022

Sets the tab navigator visibility.

ⓘ Note

This method is supported only on Unified Interface.

Syntax

```
formContext.ui.headerSection.setTabNavigatorVisible(bool);
```

Parameter

Name	Type	Required	Description
bool	Boolean	Yes	Specify true to show the tab navigator; false to hide the tab navigator.

Related topics

[formContext.ui.headerSection](#)

formContext.ui.navigation item (Client API reference)

Article • 11/30/2022

Each item represents one of the available navigation options available in the navigation bar for tables that have been updated to the new user experience or on the left side of the form for tables that have not been updated.

ⓘ Note

These methods do not work with Microsoft Dynamics 365 for tablets in versions earlier than 9.0.

Navigation item methods

Name	Description
getId	Returns the name of the item.
getLabel	Returns the label for the item.
getVisible	Returns a value that indicates whether the item is currently visible.
setFocus	Sets the focus on the item.
setLabel	Sets the label for the item.
setVisible	Sets a value that indicates whether the item is visible.

navigationItem.getId (Client API reference)

Article • 11/30/2022

Returns the name of the item.

Syntax

```
navigationItem.getId();
```

Return Value

Type: String.

Description: Name of the item.

Related topics

[formContext.ui.navigation](#)

navigationItem.getLabel (Client API reference)

Article • 11/30/2022

Returns the label for the item.

Syntax

```
navigationItem.getLabel();
```

Return Value

Type: String.

Description: Label of the item.

Related topics

[setLabel](#)

[formContext.ui.navigation](#)

navigationItem.getVisible (Client API reference)

Article • 11/30/2022

Returns a value that indicates whether the item is currently visible.

Syntax

```
navigationItem.getVisible();
```

Return Value

Type: Boolean.

Description: true if the item is visible; false otherwise..

Related topics

[setVisible](#)

[formContext.ui.navigation](#)

navigationItem.setFocus (Client API reference)

Article • 11/30/2022

Sets the focus on the item.

Syntax

```
navigationItem.setFocus();
```

Related topics

[formContext.ui.navigation](#)

navigationItem.setLabel (Client API reference)

Article • 11/30/2022

Sets the label for the item.

Syntax

```
navigationItem.setLabel(label);
```

Parameter

Name	Type	Required	Description
label	String	Yes	The new label for the item.

Related topics

[getLabel](#)

[formContext.ui.navigation](#)

navigationItem.setVisible (Client API reference)

Article • 11/30/2022

Sets a value that indicates whether the item is visible.

Syntax

```
navigationItem.setVisible(bool);
```

Parameter

Name	Type	Required	Description
bool	Boolean	Yes	Specify true or false to indicate whether the item is visible or not.

Related topics

[getVisible](#)

[formContext.ui.navigation](#)

formContext.ui.process (Client API reference)

Article • 11/30/2022

Provides methods to interact with the business process flow control on a form.

Name	Description
getDisplayState	Retrieves the display state for the business process control.
getVisible	Returns a value indicating whether the business process control is visible.
reflow	Reflows the UI of the business process control.
setDisplayState	Sets the display state of the business process control.
setVisible	Shows or hides the business process control.

process.getDisplayState (Client API reference)

Article • 11/30/2022

Retrieves the display state for the business process control.

Syntax

```
formContext.ui.process.getDisplayState();
```

Return Value

Type: String.

Description: Returns "expanded" or "collapsed" on the legacy web client; returns "expanded", "collapsed", or "floating" on Unified Interface.

Related topics

[setDisplayState](#)

[formContext.ui.process](#)

process.getVisible (Client API reference)

Article • 11/30/2022

Returns a value indicating whether the business process control is visible.

Syntax

```
formContext.ui.process.getVisible();
```

Return Value

Type: Boolean.

Description: true if the control is visible; false otherwise.

Related topics

[setVisible](#)

[formContext.ui.process](#)

reflow (Client API reference)

Article • 11/30/2022

Reflows the UI of the business process control.

Syntax

```
formContext.ui.process.reflow(updateUI, parentStage, nextStage);
```

Parameter

Name	Type	Required	Description
updateUI	Boolean	No	Specify true to update the UI of the process control; false otherwise.
parentStage	String	No	Specify the ID of the parent stage in the GUID format.
nextStage	String	No	Specify the ID of the next stage in the GUID format.

Related topics

[formContext.ui.process](#)

process.setDisplayState (Client API reference)

Article • 11/30/2022

Sets the display state of the business process control.

Syntax

```
formContext.ui.process.setDisplayState(state);
```

Parameter

Name	Type	Required	Description
state	String	Yes	Specify "expanded", "collapsed", or "floating". The value "floating" is not supported on the web client.

Related topics

[getDisplayState](#)

[formContext.ui.process](#)

process.setVisible (Client API reference)

Article • 11/30/2022

Shows or hides the business process control.

Syntax

```
formContext.ui.process.setVisible(bool);
```

Parameter

Name	Type	Required	Description
bool	Boolean	Yes	Specify true to show the control; false to hide the control.

Related topics

[getVisible](#)

[formContext.ui.process](#)

formContext.ui.quickForms (Client API reference)

Article • 11/30/2022

Provides methods to access all the quick view controls and its constituent controls on the model-driven apps forms when using the new form rendering engine (also called "turbo forms"). A quick view control is a quick view form added to a main form in model-driven apps that enables you to view information about a related table record within the main form. Data in constituent controls in a quick view control cannot be edited.

The **quickForms** collection provides access to all the quick view controls on a form, and supports all the standard methods of the collections. See [Collections](#)) for information about the collection methods.

You can retrieve a quick view control in the **quickForms** collection by using the [get](#) method by specifying either the index value (integer) or name (string) of the quick view control as the argument:

```
quickViewControl = formContext.ui.quickForms.get(arg)
```

Quick form control Methods

Name	Description
getControl	Gets the control on a form.
getControlType	Returns a string value that categorizes quick view controls.
getDisabled	Gets a boolean value indicating whether the control is disabled.
getLabel	Returns the label for the quick view control.
getName	Returns the name assigned to the quick view control.
getParent	Returns a reference to the section object that contains the control.
getVisible	Returns a value that indicates whether the quick view control is currently visible.
isLoaded	Returns whether the data binding for the constituent controls in a quick view control is complete.
refresh	Refreshes the data displayed in a quick view control.

Name	Description
setDisabled	Sets the state of the control to either enabled or disabled.
setFocus	Sets focus on the control.
setLabel	Sets the label for the quick view control.
setVisible	Displays or hides a control.

Related topics

[formContext.ui](#)

quickViewControl.getControl (Client API reference)

Article • 11/30/2022

Gets the control on a form.

Syntax

```
quickViewControl.getControl(arg);
```

Parameter

arg: Optional. You can access a single control in the constituent controls collection by passing an argument as either the name or the index value of the constituent control in a quick view control. For example: `quickViewControl.getControl("firstname")` or `quickViewControl.getControl(0)`

Return Value

Type: Object or Object collection.

Description: Object if you use the method with parameter; object collection if you use the method without any parameters.

Remarks

After you have retrieved a constituent control in a quick view control, you can use any of the methods supported for a control in model-driven apps on the constituent control that does not alter the constituent control data. This is because constituent controls in a quick view control are read only. For example, you can use:

```
quickViewControl.getControl(0).getAttribute()
```

For more information about methods supported for a control, see [Controls](#).

 **Important**

The **getAttribute** or any data related methods on a constituent control might not work on the main form **OnLoad** event because the quick view form that its bound to might not have loaded completely when the main form has loaded. You must use the **isLoaded** method for the quick view control instance to help you determine if the bounded quick view form has loaded completely.

Also, the way you retrieve constituent controls in a quick view control on forms using the new form rendering engine is different from the legacy forms. So, if you are using legacy forms and have code targeting constituent controls in a quick view control, you must update your code when you decide to use the new form rendering engine.

Related topics

[formContext.ui.quickForms](#)

quickViewControl.getControlType (Client API reference)

Article • 11/30/2022

Returns a string value that categorizes quick view controls.

Syntax

```
quickViewControl.getControlType();
```

Return Value

Type: String.

Description: For a quick view control, the method returns "quickform".

For a constituent control in a quick view control, the method returns the actual category of the control. For more information about possible return values, see [getControlType..](#)

Related topics

[formContext.ui.quickForms](#)

quickViewControl.getDisabled (Client API reference)

Article • 11/30/2022

Gets a boolean value indicating whether the control is disabled.

Syntax

```
quickViewControl.getDisabled();
```

Return Value

Type: Boolean.

Description: true if disabled; false otherwise.

Related topics

[formContext.ui.quickForms](#)

quickViewControl.getLabel (Client API reference)

Article • 11/30/2022

Returns the label for the quick view control.

Syntax

```
quickViewControl.getLabel();
```

Return Value

Type: String.

Description: Label of the quick view control.

Related topics

[setLabel](#)

[formContext.ui.quickForms](#)

quickViewControl.getName (Client API reference)

Article • 11/30/2022

Returns the name assigned to the quick view control.

Syntax

```
quickViewControl.getName();
```

Return Value

Type: String.

Description: The name of the quick view control.

Related topics

[formContext.ui.quickForms](#)

quickViewControl.getParent (Client API reference)

Article • 11/30/2022

Returns a reference to the section object that contains the control.

Syntax

```
quickViewControl.getParent();
```

Return Value

Type: [formContext.ui.section](#)

Related topics

[formContext.ui.quickForms](#)

quickViewControl.getVisible (Client API reference)

Article • 11/30/2022

Returns a value that indicates whether the quick view control is currently visible.

ⓘ Note

If the containing **section** or **tab** for this control isn't visible, this method can still return **true**. To make certain that the control is actually visible; you need to also check the visibility of the containing elements.

Syntax

```
quickViewControl.getVisible();
```

Return Value

Type: Boolean.

Description: true if the control is visible; false otherwise.

Related topics

[setVisible](#)

[formContext.ui.quickForms](#)

isLoading (Client API reference)

Article • 11/30/2022

Returns whether the data binding for the constituent controls in a quick view control is complete.

Syntax

```
quickViewControl.isLoading();
```

Return Value

Type: Boolean.

Description: true is the data binding for a constituent control is complete; false otherwise.

Remarks

The data binding for the constituent controls in a quick view control may not be complete during the main form **OnLoad** event because the quick view form that the control is bound to may not have loaded completely. As a result, using the [getAttribute](#) or any data-related methods on a constituent control might not work. The **isLoading** method for the quick view control helps determine the data binding status for constituent controls in a quick view control.

Example

The following sample code demonstrates how you can use the **isLoading** method to check the binding status, and then retrieve the value of the column that a constituent control in a quick view control is bound to.

JavaScript

```
function getAttributeValue(executionContext) {
    var formContext = executionContext.getFormContext();
    var quickViewControl = formContext.ui.quickForms.get(""
<QuickViewControlName>");
    if (quickViewControl != undefined) {
        if (quickViewControl.isLoading()) {
            // Access the value of the column bound to the constituent
```

```
control
    var myValue =
quickViewControl.getControl(0).getAttribute().getValue();
    console.log(myValue);

    // Search by a specific column present in the control
    var myValue2 = quickViewControl.getControl().find(control =>
control.getName() == "<AttributeSchemaName>").getAttribute().getValue();
    console.log(myValue2);

    return;
}
else {
    // Wait for some time and check again
    setTimeout(getAttributeValue, 10, executionContext);
}
else {
    console.log("No data to display in the quick view control.");
    return;
}
}
```

Related topics

[formContext.ui.quickForms](#)

quickViewControl.refresh (Client API reference)

Article • 11/30/2022

Refreshes the data displayed in a quick view control.

Syntax

```
quickViewControl.refresh();
```

Related topics

[formContext.ui.quickForms](#)

quickViewControl.setDisabled (Client API reference)

Article • 11/30/2022

Sets the state of the control to either enabled or disabled.

Syntax

```
quickViewControl.setDisabled(bool);
```

Parameter

Name	Type	Required	Description
bool	Boolean	Yes	Specify true or false to disable or enable the control.

Related topics

[getDisabled](#)

[formContext.ui.quickForms](#)

quickViewControl.setFocus (Client API reference)

Article • 11/30/2022

Sets focus on the control.

Syntax

```
quickViewControl.setFocus();
```

Related topics

[formContext.ui.quickForms](#)

quickViewControl.setLabel (Client API reference)

Article • 11/30/2022

Sets the label for the quick view control.

Syntax

```
quickViewControl.setLabel(label);
```

Parameter

Name	Type	Required	Description
label	String	Yes	The new label of the quick view control.

Related topics

[getLabel](#)

[formContext.ui.quickForms](#)

quickViewControl.setVisible (Client API reference)

Article • 11/30/2022

Displays or hides a control.

Syntax

```
quickViewControl.setVisible(bool);
```

Parameter

Name	Type	Required	Description
bool	Boolean	Yes	Specify true or false to display or hide the control.

Related topics

[getVisible](#)

[formContext.ui.quickForms](#)

formContext.ui.tabs (Client API reference)

Article • 12/09/2022

A tab is a group of sections on a page. It contains properties and methods to manipulate tabs as well as access to sections within the tab through the sections collection.

You can retrieve a tab object (say `tabObj`) by using the following example code:

JavaScript

```
var tabObj = formContext.ui.tabs.get(arg);
```

Properties

- **sections**: The sections collection provides access to sections within the tab. See [Collections \(Client API reference\)](#) for information about methods to access the sections in the collection. See [formContext.ui.tabs section](#) for information about the properties and methods of the section objects in the collection.

Methods

Name	Description
addTabStateChange	Adds a function to be called when the TabStateChange event occurs.
getContentType	Returns the content type.
getDisplayState	Gets display state of the tab.
getLabel	Returns the label for the tab.
getName	Returns the name of the tab.
getParent	Returns the formContext.ui object containing the tab.
getVisible	Returns a value that indicates whether the tab is currently visible.
removeTabStateChange	Removes a function to be called when the TabStateChange event occurs.
setContent-Type	Sets the content type.

Name	Description
setDisplayState	Sets display state of the tab.
setFocus	Sets the focus on the tab.
setLabel	Sets the label of the tab.
setVisible	Sets a value that indicates whether the tab is visible.

Related topics

[formcontext.ui.tabs](#)

[formContext](#)

addTabStateChange (Client API reference)

Article • 11/30/2022

Adds a function to be called when the [TabStateChange](#) event occurs..

Syntax

```
tabObj.addTabStateChange(myFunction);
```

Parameter

Name	Type	Required	Description
myFunction	function reference	Yes	The function to be executed on the TabStateChange event. The function will be added to the bottom of the event handler pipeline. The execution context is automatically passed as the first parameter to the function. See Execution context for more information.

Related topics

[formContext.ui](#)

[formContext](#)

getContentType (Client API reference)

Article • 11/30/2022

Returns the content type.

ⓘ Note

This method is supported only on Unified Interface.

Syntax

```
tabObj.getContentType();
```

Return Value

Type: Content type.

Description: Returns "cardSections" or "singleComponent".

Related topics

[formContext.ui.tabs](#)

tab.getDisplayState (Client API reference)

Article • 11/30/2022

Gets display state of the tab.

Syntax

```
tabObj.getDisplayState();
```

Return Value

Type: String.

Description: Returns "expanded" or "collapsed".

Related topics

[setDisplayState](#)

tab.getLabel (Client API reference)

Article • 11/30/2022

Returns the label for the tab.

Syntax

```
tabObj.getLabel();
```

Return Value

Type: String

Description: The label of the tab.

Related topics

[setLabel](#)

tab.getName (Client API reference)

Article • 11/30/2022

Returns the name of the tab.

Syntax

```
tabObj.getName();
```

Return Value

Type: String

Description: Name of the tab.

tab.getParent (Client API reference)

Article • 11/30/2022

Returns the [formContext.ui](#) object containing the tab.

Syntax

```
tabObj.getParent();
```

Return Value

Type: [formContext.ui](#) object

tab.getVisible (Client API reference)

Article • 11/30/2022

Returns a value that indicates whether the tab is currently visible.

Syntax

```
tabObj.getVisible();
```

Return Value

Type: Boolean.

Description: true if the tab is visible; false otherwise.

Related topics

[setVisible](#)

removeTabStateChange (Client API reference)

Article • 11/30/2022

Removes a function to be called when the [TabStateChange](#) event occurs..

Syntax

```
tabObj.removeTabStateChange(myFunction);
```

Parameter

Name	Type	Required	Description
myFunction	function reference	Yes	The function to be removed from the TabStateChange event.

Related topics

[formContext.ui](#)

[formContext](#)

setContentType (Client API reference)

Article • 11/30/2022

Sets the content type.

ⓘ Note

This method is supported only on Unified Interface.

Syntax

```
tabObj.setContentType(contentType);
```

Parameter

Name	Type	Required	Description
contentType	String	Yes	Defines the content type. It has the following parameters: - cardSections : The default tab behavior. - singleComponent : Maximizes the content of the first component in the tab.

Related topics

[getContentType](#)

tab.setDisplayState (Client API reference)

Article • 11/30/2022

ⓘ Important

The `setDisplayState` method will be deprecated with the 2021 release wave 1 (April 2021). Use the [setFocus](#) method in Unified Interface to ensure the correct tab is opened on a form.

Sets display state of the tab.

Syntax

```
tabObj.setDisplayState(state);
```

Parameter

Name	Type	Required	Description
state	String	Yes	Specify "expanded" or "collapsed".

Related topics

[getDisplayState](#)

tab.setFocus (Client API reference)

Article • 11/30/2022

Sets the focus on the tab.

Syntax

```
tabObj.setFocus();
```

Related topics

[formcontext.ui.tabs](#)

tab.setLabel (Client API reference)

Article • 11/30/2022

Sets the label of the tab.

Syntax

```
tabObj.setLabel(label);
```

Parameter

Name	Type	Required	Description
label	String	Yes	The new label of the tab.

Related topics

[getLabel](#)

tab.setVisible (Client API reference)

Article • 11/30/2022

Sets a value that indicates whether the tab is visible.

Syntax

```
tabObj.setVisible(bool);
```

Parameter

Name	Type	Required	Description
bool	Boolean	Yes	Specify true to show the tab; false to hide the tab.

Remarks

Another way to hide a tab is to hide all the sections within it. If all the sections within a tab are not visible, the tab will not be visible.

Note

If you set the value to false, the first visible tab is shown by default. If there are no visible tabs, the body of the form will show as a blank page.

Related topics

[getVisible](#)

formContext.ui.tabs section (Client API reference)

Article • 12/09/2022

A section contains methods to manage how it appears as well as accessing the tab that contains the section.

You can retrieve a section object (say `sectionObj`) by using the following example code:

JavaScript

```
var tabObj = formContext.ui.tabs.get(arg);
var sectionObj = tabObj.sections.get(arg);
```

Properties

- **controls**: The section controls collection provides access to the controls within a section. See [Collections \(Client API reference\)](#) for information about the methods exposed by collections. See [Controls \(Client API reference\)](#) for information about the properties and methods exposed by the objects in this collection.

Methods

Name	Description
getLabel	Returns the label for the section.
getName	Returns the name of the section.
getParent	Returns the tab containing the section.
getVisible	Returns a value that indicates whether the section is currently visible.
setLabel	Sets the label of the section.
setVisible	Sets a value that indicates whether the section is visible.

Related topics

[formcontext.ui.tabs](#)

[formContext](#)

section.getLabel (Client API reference)

Article • 11/30/2022

Returns the label for the section.

Syntax

```
sectionObj.getLabel();
```

Return Value

Type: String

Description: The label of the section.

Related topics

[setLabel](#)

section.getName (Client API reference)

Article • 11/30/2022

Returns the name of the section.

Syntax

```
sectionObj.getName();
```

Return Value

Type: String

Description: Name of the section.

Related topics

[Controls](#)

section.getParent (Client API reference)

Article • 11/30/2022

Returns the tab containing the section.

Syntax

```
sectionObj.getParent();
```

Return Value

Type: [formContext.ui tab](#) object

section.getVisible (Client API reference)

Article • 11/30/2022

Returns a value that indicates whether the section is currently visible.

Syntax

```
sectionObj.getVisible();
```

Return Value

Type: Boolean.

Description: true if the section is visible; false otherwise.

Related topics

[setVisible](#)

section.setLabel (Client API reference)

Article • 11/30/2022

Sets the label of the section.

Syntax

```
sectionObj.setLabel(label);
```

Parameter

Name	Type	Required	Description
label	String	Yes	The new label of the section.

Related topics

[getLabel](#)

section.setVisible (Client API reference)

Article • 11/30/2022

Sets a value that indicates whether the section is visible.

Syntax

```
sectionObj.setVisible(bool);
```

Parameter

Name	Type	Required	Description
bool	Boolean	Yes	Specify true to show the section; false to hide the section.

Related topics

[getVisible](#)

Grids and subgrids in model-driven apps (Client API reference)

Article • 12/16/2022

Grids present data in a tabular format in model-driven apps. Grids can span the entire form or can be one of the items on a form; the latter are called *subgrids*.

Types of grids

There are two types of grids in model-driven apps:

- **Read-only grids:** Display data in a tabular format. To edit the data displayed in a read-only grid, you have to select the record in the grid to open the form, edit the data, and then save.
- **Editable grids:** In addition to displaying data in a tabular format, provides rich inline editing capabilities on web and mobile clients including the ability to group, sort, and filter data within the same grid so that you do not have to switch records or views. The editable grid is a custom control, and is supported in the main grid and subgrids on a form in the web client and in dashboards and on form grids on the mobile clients. Although the editable grid control provides editing capability, it honors the read-only grid metadata and field-level security settings.

Getting the grid context

Grid context is the grid or subgrid instance on a form against which you want to run your code. For more information about getting the grid context to execute your JavaScript code, see [Client API grid context](#)

Events

Name	Description	Applicable for
Subgrid OnLoad Event	Occurs every time the subgrid refreshes. This includes when users sort values in subgrid by clicking the column headings.	Read-only grid
Grid OnChange	Occurs when a value is changed in a cell in the editable grid and the cell loses focus	Editable grid

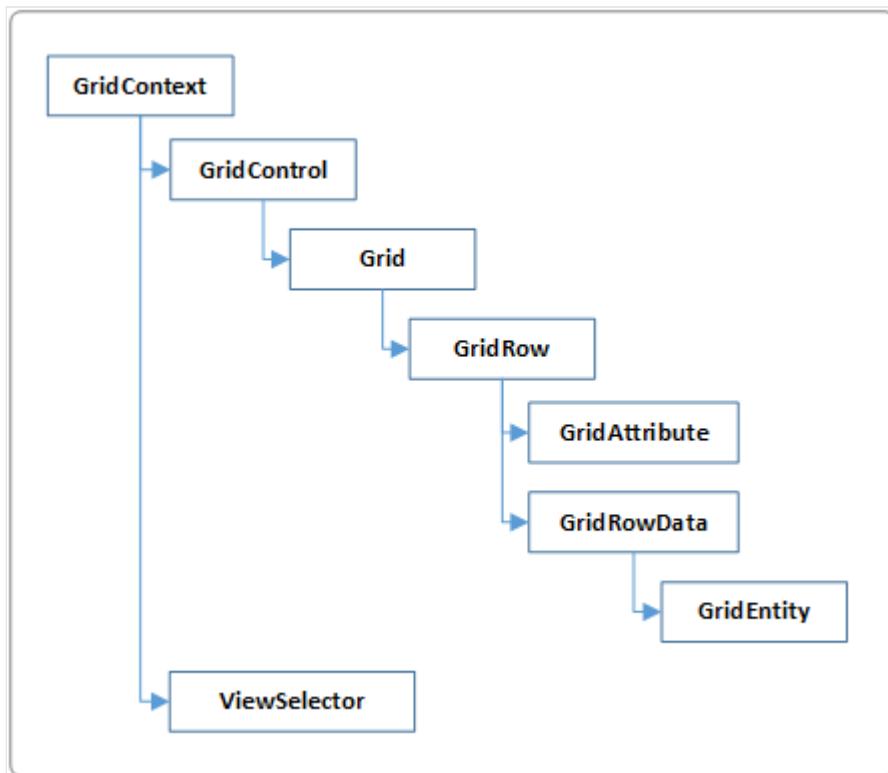
Name	Description	Applicable for
Grid OnRecordSelect	Occurs when a single row (record) is selected in the editable grid	Editable grid
Grid OnSave	Occurs before sending the updated information to the server, and when any of the following occurs: there is a change in the record selection, the user explicitly triggers a save operation using the editable grid's save button, or the user applies a sort, filter, group, pagination, or navigation operation from the editable grid while there are pending changes.	Editable grid

ⓘ Note

You can register for the **OnChange**, **OnRecordSelect**, and **OnSave** events using the **Events** tab of the model-driven apps page that is used to enable editable grids for a table or a read-only grid.

Objects

Use the following objects to interact with grids. These objects form a hierarchy as shown in the following diagram:



Name	Description	Available for
------	-------------	---------------

Name	Description	Available for
GridControl	Provides methods to work with the grid or subgrid control.	Read-only and editable grids
Grid	Provides methods to access information about data in the grid.	Read-only and editable grids
GridRow	Provides methods to work with rows or selected rows in the grid.	Read-only and editable grids
GridRowData	Provides methods to work with rows or selected rows in the grid.	Read-only and editable grids
GridEntity	Provides methods to access data about the specific records in the rows.	Read-only and editable grids
GridAttribute	Provides methods to access the data in the cell of an editable grid.	Editable grid
GridCell	Provides methods to access the data related to control on a form that is tied to a column in an editable grid.	Editable grid
ViewSelector	Provides methods to get or set information about the view selector of the subgrid control.	Read-only grid

Related topics

[Client API grid context](#)

[Use editable grids](#)

[Client API Reference for model-driven apps](#)

[Model-driven apps Developer Overview](#)

GridControl (Client API reference)

Article • 12/16/2022

GridControl or the gridContext is the instance of grid or subgrid on a form against which you want to execute your script. Use the form context to get GridControl ([gridContext](#)) on a form.

Methods for grids

Name	Description	Available for
addOnLoad	Adds event handlers to the Subgrid OnLoad event event.	Read-only grid
getEntityName	Gets the logical name of the table data displayed in the grid.	Read-only and editable grids
getFetchXml	Gets the FetchXML query that represents the current data, including filtered and sorted data, in the grid control.	Read-only and editable grids
getGrid	Get access to the Grid available in the GridControl (gridContext).	Read-only and editable grids
getGridType	Gets the grid type (grid or subgrid).	Read-only and editable grids
getRelationship	Gets information about the relationship used to filter the subgrid.	Read-only and editable grids
getUrl	Gets the URL of the current grid control.	Read-only and editable grids
getViewSelector	Use this method to access the ViewSelector methods available for the grid control.	Read-only grid
openRelatedGrid	Displays the associated grid for the grid.	Read-only and editable grids
refresh	Refreshes the grid.	Read-only and editable grids
refreshRibbon	Refreshes the ribbon rules for the grid control.	Read-only and editable grids
removeOnLoad	Removes event handlers from the Subgrid OnLoad event event.	Read-only grid

Additional methods for subgrids

Along with the methods mentioned above, subgrid also have the following methods:

Name	Description	Available for
getControlType	Returns a value that categorizes controls.	Read-only and editable grids
getDisabled	Returns whether the control is disabled.	Read-only and editable grids
getName	Returns the name assigned to the control.	Read-only and editable grids
getParent	Returns a reference to the section object that contains the control.	Read-only and editable grids
getVisible	Returns a value that indicates whether the control is currently visible.	Read-only and editable grids
setDisabled	Sets whether the control is disabled.	Read-only and editable grids
setFocus	Sets the focus on the control.	Read-only and editable grids
setVisible	Sets a value that indicates whether the control is visible.	Read-only and editable grids

Related topics

[Grid](#)

[Grids and subgrids in model-driven apps](#)

gridContext.addOnLoad (Client API reference)

Article • 12/16/2022

Adds event handlers to the [Subgrid OnLoad event](#) event.

Grid types supported

Read-only and editable grids

Syntax

```
gridContext.addOnLoad(myFunction);
```

Parameter

Name	Type	Required	Description
myFunction	function reference	Yes	The function to be executed when the subgrid loads. The function will be added to the bottom of the event handler pipeline. The execution context is automatically passed as the first parameter to the function. See execution context for more information.

Remarks

To get the `gridContext`, see [Getting the grid context](#).

Example

Add the `myContactsGridOnloadFunction` function to the Contacts subgrid [OnLoad](#) event.

JavaScript

```
function myFunction(executionContext) {
    let formContext = executionContext.getFormContext(); // get the form context
    let gridContext = formContext.getControl("Contacts");// get the grid
```

```
context
  let myContactsGridOnloadFunction = function () { console.log("Contacts
Subgrid OnLoad event occurred") };
  gridContext.addOnLoad(myContactsGridOnloadFunction);
}
```

Related topics

[removeOnLoad](#)

gridContext.getEntityName (Client API reference)

Article • 12/16/2022

Gets the logical name of the table data displayed in the grid.

Grid types supported

Read-only and editable grids

Syntax

```
gridContext.getEntityName();
```

Return Value

Type: String

Description: The logical name of the table data displayed in the grid.

Remarks

To get the `gridContext`, see [Getting the grid context](#).

getFetchXml (Client API reference)

Article • 12/16/2022

Gets the FetchXML query that represents the current data, including filtered and sorted data, in the grid control.

Grid types supported

Read-only and editable grids

Syntax

```
var result = gridContext.getFetchXml();
```

Return Value

Type: String

Description: The FetchXML query.

Remarks

To get the `gridContext`, see [Getting the grid context](#)

Example

The following example displays the retrieved Fetch XML of the Contacts subgrid in the console:

JavaScript

```
function myFunction(executionContext) {
    var formContext = executionContext.getFormContext(); // get the form
    context
    var gridContext = formContext.getControl("Contacts"); // get the grid
    context
    var retrieveFetchXML = function () {
        var result = gridContext.getFetchXml();
        console.log(result)
    };
}
```

```
    gridContext.addOnLoad(retrieveFetchXML);  
}
```

getGrid (Client API reference)

Article • 12/16/2022

Get access to the [Grid](#) available in the GridControl (`gridContext`).

Grid types supported

Read-only and editable grids

Syntax

```
let grid = gridContext.getGrid();
```

Return Value

Type: [Grid](#)

Description: The [Grid](#) object.

Remarks

To get the `gridContext`, see [Getting the grid context](#).

getGridType (Client API reference)

Article • 12/16/2022

Gets the grid type (grid or subgrid).

Grid types supported

Read-only and editable grids

Syntax

```
let gridType = gridContext.getGridType();
```

Return Value

Type: Number

Description: Returns one of the following values:

Value	Description
1	HomePageGrid
2	Subgrid

Remarks

To get the `gridContext`, see [Getting the grid context](#).

getRelationship (Client API reference)

Article • 12/16/2022

Gets information about the relationship used to filter the subgrid.

Grid types supported

Read-only and editable grids

Syntax

```
gridContext.getRelationship();
```

Return Value

Type: Object.

Description: A relationship object with the following:

- **attributeName**: String. Name of the column.
- **name**: String. Name of the relationship.
- **navigationPropertyName**: String. Name of the navigation property for this relationship.
- **relationshipType**: Number. Returns one of the following values to indicate the relationship type:
 - 0: OneToMany
 - 1: ManyToMany
- **roleType**: Number. Returns one of the following values to indicate the role type of relationship:
 - 1: Referencing
 - 2: AssociationEntity

Remarks

To get the `gridContext`, see [Getting the grid context](#).

Related topics

[openRelatedGrid](#)

getUrl (Client API reference)

Article • 12/16/2022

Gets the URL of the current grid control.

Grid types supported

Read-only and editable grids

Syntax

```
gridContext.getUrl(client);
```

Parameter

Name	Type	Required	Description
client	Number	No	Indicates the client type. You can specify one of the following values: 0: Browser 1: MobileApplication

Return Value

Type: String

Description: The Url of the current grid control.

Remarks

To get the `gridContext`, see [Getting the grid context](#).

getViewSelector (Client API reference)

Article • 12/16/2022

Use this method to access the [ViewSelector methods](#) available for the grid control.

Grid types supported

Read-only grid

Syntax

```
gridContext.getViewSelector();
```

Return Value

Type: [ViewSelector](#)

Description: The [ViewSelector](#) object.

Remarks

To get the `gridContext`, see [Getting the grid context](#).

openRelatedGrid (Client API reference)

Article • 12/16/2022

Displays the associated grid for the grid.

This method does nothing if the grid is not filtered based on a relationship.

Grid types supported

Read-only and editable grids

Syntax

```
gridContext.openRelatedGrid();
```

Remarks

To get the `gridContext`, see [Getting the grid context](#).

Related topics

[getRelationship](#)

gridContext.refresh (Client API reference)

Article • 12/16/2022

Refreshes the grid.

Grid types supported

Read-only and editable grids

Syntax

```
gridContext.refresh();
```

Remarks

To get the `gridContext`, see [Getting the grid context](#).

gridContext.refreshRibbon (Client API reference)

Article • 12/16/2022

Refreshes the ribbon rules for the grid control.

Grid types supported

Read-only and editable grids

Syntax

```
gridContext.refreshRibbon();
```

Remarks

To get the `gridContext`, see [Getting the grid context](#).

gridContext.removeOnLoad (Client API reference)

Article • 12/16/2022

Removes event handlers from the [Subgrid OnLoad event](#) event.

Grid types supported

Read-only grids

Syntax

```
gridContext.removeOnLoad(myFunction);
```

Parameter

Name	Type	Required	Description
myFunction	function reference	Yes	The function to be removed from the OnLoad event.

Remarks

To get the `gridContext`, see [Getting the grid context](#).

Related topics

[addOnLoad](#)

Grid (Client API reference)

Article • 12/16/2022

Grid is returned by the [getGrid](#) method. Use Grid methods to access information about data in the grid.

```
let myGrid = gridContext.getGrid();
```

Methods

Name	Description	Available for
getRows	Returns a collection of every GridRow in the Grid.	Read-only and editable grids
getSelectedRows	Returns a collection of every selected GridRow in the Grid.	Read-only and editable grids
getTotalRecordCount	Returns the total number of records that match the filter criteria of the view, not limited by the number visible in a single page.	Read-only and editable grids

Related topics

[GridRow](#)

[Grids and subgrids in model-driven apps](#)

getRows (Client API reference)

Article • 12/16/2022

Returns a collection of every [GridRow](#) in the Grid.

Grid types supported

Read-only and editable grids

Syntax

```
let allRows = gridContext.getGrid().getRows();
```

Return Value

Type: [Collection](#)

Description: A collection of rows in the grid.

Remarks

To get the `gridContext`, see [Getting the grid context](#).

See [Collections \(Client API reference\)](#) for information on the methods available to access data in a collection.

getSelectedRows (Client API reference)

Article • 12/16/2022

Returns a collection of every selected [GridRow](#) in the Grid.

Grid types supported

Read-only and editable grids

Syntax

```
let allSelectedRows = gridContext.getGrid().getSelectedRows();
```

Return Value

Type: [Collection](#)

Description: A collection of selected rows in the grid.

Remarks

To get the `gridContext`, see [Getting the grid context](#).

See [Collections \(Client API reference\)](#) for information on the methods available to access data in a collection.

getTotalRecordCount (Client API reference)

Article • 12/16/2022

Returns the total number of records that match the filter criteria of the view, not limited by the number visible in a single page.

- When the Dynamics 365 for Outlook client isn't connected to the server, this number is limited to those records that the user has selected to take offline.
- For Dynamics 365 mobile clients, this method will return the number of records in the subgrid.

Grid types supported

Read-only and editable grids

Syntax

```
let filteredRecordCount = gridContext.getGrid().getTotalRecordCount();
```

Return Value

Type: Number

Description: Total number of records that match the filter criteria of the view.

Remarks

To get the `gridContext`, see [Getting the grid context](#).

See [Collections \(Client API reference\)](#) for information on the methods available to access data in a collection.

GridRow (Client API reference)

Article • 12/16/2022

A collection of GridRow is returned by [Grid.getRows](#) and [Grid.getSelectedRows](#) methods.

JavaScript

```
var myRows = gridContext.getGrid().getRows();
var gridRow = myRows.get(arg);
```

Properties

Name	Description	Available for
data	An object containing the GridRowData for the GridRow.	Read-only and editable grids

Methods

Name	Description	Available for
getData	Deprecated. Returns the GridRowData for the GridRow.	Read-only and editable grids

Related topics

[GridRowData](#)

[Grids and subgrids in model-driven apps](#)

gridRow.getData (Client API reference)

Article • 12/16/2022

Deprecated. Returns the [GridRowData](#) for the GridRow.

As this is deprecated, you should use [GridRow.data](#).

Grid types supported

Read-only and editable grids

Syntax

```
gridRow.getData();
```

Return Value

Type: [GridRowData](#)

Remarks

To get the `gridRow` object, see [GridRow](#).

GridRowData (Client API reference)

Article • 12/16/2022

GridRowData is returned by the [GridRow.getData](#) method.

GridRowData also provides methods for retrieving information specific to a record displayed in an editable grid row, including a collection of all the columns included in the row. Column data is limited to the columns presented by the editable grid. See [Collections \(Client API reference\)](#) for information on the methods available to access data in a collection.

JavaScript

```
var myRows = gridContext.getGrid().getRows();
var myRow = myRows.get(arg);
var gridRowData = myRow.data;
```

Properties

Name	Description	Available for
entity	Returns the GridEntity for the GridRowData.	Read-only and editable grids

Methods

Name	Description	Available for
getEntity	Deprecated. Returns the GridEntity for the GridRowData.	Read-only and editable grids

getEntity (Client API reference)

Article • 12/16/2022

Deprecated. Returns the [GridEntity](#) for the GridRowData.

As this is deprecated, you should use [GridRowData.entity](#).

Grid types supported

Read-only and editable grids

Syntax

```
gridRowData.getEntity();
```

Return Value

Type: [GridEntity](#)

Remarks

To get the `gridRowData` object, see [GridRowData](#).

GridEntity (Client API reference)

Article • 12/16/2022

GridEntity is returned by the [GridRowData.getEntity](#) method or by directly accessing the [GridRowData.entity](#) object. Use the GridEntity methods to access data about the specific records in the rows.

JavaScript

```
var myRows = gridContext.getGrid().getRows();
var myRow = myRows.get(arg);
var gridEntity = myRow.data.entity;
```

GridEntity also supports the **columns** collection that provides methods of working with a collection of columns for a table in the editable grid. Each column ([GridAttribute](#)) represents the data in the cell of an editable grid, and contains a reference to all the cells associated with the column. See [Collections \(Client API reference\)](#) for information on the methods available to access data in a collection.

Methods

Name	Description	Available for
getEntityName	Returns the logical name for the record in the row.	Read-only and editable grids
getEntityReference	Returns a Lookup value that references the record in the row.	Read-only and editable grids
getId	Returns the Id for the record in the row.	Read-only and editable grids
getPrimaryAttributeValue	Returns the primary column value for the record in the row.	Read-only grid

Related topics

[GridAttribute](#)

[Grids and subgrids in model-driven apps](#)

[Columns](#)

gridEntity.getEntityName (Client API reference)

Article • 12/16/2022

Returns the logical name for the record in the row.

Grid types supported

Read-only and editable grids

Syntax

```
gridEntity.getEntityName();
```

Return Value

Type: String

Description: The logical name for the record in the row.

Remarks

To get the `gridEntity` object, see [GridEntity](#).

gridEntity.getEntityReference (Client API reference)

Article • 12/16/2022

Returns a Lookup value that references the record in the row.

Grid types supported

Read-only and editable grids

Syntax

```
gridEntity.getEntityReference();
```

Return Value

Type: Lookup

Description: Lookup object that references the record in the row. The object has the following values:

- **entityType**: String. The logical name for the record in the row. The same data returned by the [GridEntity.getEntityName](#) method.
- **id**: String. The Id for the record in the row. The same data returned by the [GridEntity.getId](#) method.
- **name**: String. The primary column value for the record in the row. The same data returned by the [GridEntity.getPrimaryAttributeValue](#) method.

Remarks

To get the `gridEntity` object, see [GridEntity](#).

gridEntity.getId (Client API reference)

Article • 12/16/2022

Returns the Id for the record in the row.

Grid types supported

Read-only and editable grids

Syntax

```
gridEntity.getId();
```

Return Value

Type: String

Description: The Id for the record in the row.

Remarks

To get the `gridEntity` object, see [GridEntity](#).

gridEntity.getPrimaryAttributeValue (Client API reference)

Article • 12/16/2022

Returns the primary column value for the record in the row.

Grid types supported

Read-only grid

Syntax

```
gridEntity.getPrimaryAttributeValue();
```

Return Value

Type: String

Description: The primary column value for the record in the row.

Remarks

To get the `gridEntity` object, see [GridEntity](#).

GridAttribute (Client API reference)

Article • 12/16/2022

GridAttribute is supported for both read-only and editable grids.

GridAttribute represents the data in the cell of an editable grid, and contains a reference to all the cells associated with the column. See [Collections \(Client API reference\)](#) for information on the methods available to access data in a collection.

GridAttribute also supports the **controls** collection for columns of a selected grid row, which provides methods to work with a collection of cells associated with the column. Each cell ([GridCell](#)) of a selected grid row is analogous to a control on a form that is tied to a column in an editable grid. See [Collections \(Client API reference\)](#) for information on the methods available to access data in a collection.

Tip

For performance reasons, a row (record) in an editable grid is not editable until the record is selected. Users must select a single record in a grid to edit it. Once a record is selected in an editable grid, Dynamics 365 internally evaluates a number of things including user access to the record, whether the record is active, and column validations to ensure that data security and validity are honored when you edit data. Consider using the **OnRecordSelect** event with the **getFormContext** method to access records in the grid that are in the editable state.

Methods

GridAttribute supports the following methods for columns of a selected grid row.

Name	Description
getName	Returns the logical name of the column of a selected grid row.
getRequiredLevel	Returns a string value indicating whether a value for the column is required or recommended.
setRequiredLevel	Sets whether data is required or recommended for the column of a selected grid row before the record can be saved.
getValue	Retrieves the data value for a column.
setValue	Sets the data value for a column.

 **Note**

To select a row in an editable grid, use the `Grid.getSelectedRows`

Related topics

[GridCell](#)

[Grids and subgrids in model-driven apps](#)

[Controls collection](#)

GridCell (Client API reference)

Article • 12/16/2022

GridCell is only supported for editable grids.

GridCell of a selected grid row is analogous to a control on a form that is tied to a column in an editable grid. See [Collections \(Client API reference\)](#) for information on the methods available to access data in a collection.

Methods

GridCell supports the following methods.

Name	Description
clearNotification	Clears notification for a cell.
getDisabled	Returns whether the cell is disabled (read-only).
setDisabled	Sets whether the cell is disabled. NOTE: Enabling a read-only cell for editing can cause an error when the record is saved. If the column is considered read-only by the server, an error may occur if the value is modified. This may happen in scenarios where the user doesn't have write privileges to the record, the record is disabled, or the user doesn't have the necessary field-level security privileges.
setNotification	Displays an error message for a cell to indicate that data isn't valid.
getLabel	Returns the label of the column that contains the cell.

Related topics

[Grids and subgrids in model-driven apps](#)

[Controls](#)

ViewSelector methods (Client API reference)

Article • 12/16/2022

Provides methods to get or set information about the view selector of the subgrid control. If the subgrid control is not configured to display the view selector, calling the **ViewSelector** methods will throw an error.

ViewSelector is returned by the [gridContext.getViewSelector](#) method.

JavaScript

```
var viewSelector = gridContext.getViewSelector();
```

Methods

Name	Description	Available for
getCurrentView	Gets a reference to the current view.	Read-only grid
isVisible	Returns a boolean value to indicate whether the view selector is visible.	Read-only grid
setCurrentView	Sets the current view.	Read-only grid

Related topics

[gridContext](#)

[Grids and subgrids in model-driven apps](#)

getCurrentView (Client API reference)

Article • 12/16/2022

Gets a reference to the current view.

Grid types supported

Read-only grid

Syntax

```
viewSelector.getCurrentView();
```

Return Value

Type: Lookup object

Description: The Lookup object has the following values:

- **entityType**: Number. The object type code for the SavedQuery (1039) or UserQuery (4230) that represents the view the user can select.
- **id**: String. The Id for the view the user can select.
- **name**: String. The name of the view the user can select.

Remarks

If the subgrid control is not configured to display the view selector, calling this method on the `viewSelector` object will throw an error.

To get the `viewSelector` object, see [ViewSelector](#).

isVisible (Client API reference)

Article • 12/16/2022

Returns a boolean value to indicate whether the view selector is visible.

Grid types supported

Read-only grid

Syntax

```
viewSelector.isVisible();
```

Return Value

Type: Boolean

Description: true if visible; false otherwise.

Remarks

If the subgrid control is not configured to display the view selector, calling this method on the [ViewSelector](#) returned by the `GridControl.getViewSelector` method will throw an error.

To get the `viewSelector` object, see [ViewSelector](#).

setCurrentView (Client API reference)

Article • 12/16/2022

Sets the current view.

Grid types supported

Read-only grid

Syntax

```
viewSelector.setCurrentView(object);
```

Parameter

Name	Type	Required	Description
object	Lookup object	Yes	Specify the Lookup object that has the following values: - entityType : Number. The object type code for the SavedQuery (1039) or UserQuery (4230) that represents the view the user can select. - id : String. The Id for the view the user can select. - name : String. The name of the view the user can select.

Remarks

If the subgrid control is not configured to display the view selector, calling this method on the `viewSelector` object will throw an error.

To get the `viewSelector` object, see [ViewSelector](#).

Example

JavaScript

```
function setView(executionContext) {
    var ContactsIFollow = {
        entityType: 1039, // SavedQuery
        id: "3A282DA1-5D90-E011-95AE-00155D9CFA02",
        name: "Contacts I Follow"
```

```
}

// Get the gridContext
var formContext = executionContext.getFormContext();
var gridContext = formContext.getControl("Contacts");

// Set the view using ContactsIFollow
gridContext.getViewSelector().setCurrentView(ContactsWithFollow);
}
```

Related topics

[ViewSelector](#)

Xrm.App (Client API reference)

Article • 11/30/2022

Provides app-related methods.

Method	Description
addGlobalNotification	Displays an error, information, warning, or success notification for an app, and lets you specify actions to execute based on the notification.
clearGlobalNotification	Clears a notification in the app.

Related topics

[Client API Xrm object](#)

addGlobalNotification (Client API reference)

Article • 11/30/2022

Displays an error, information, warning, or success notification for an app, and lets you specify actions to execute based on the notification.

Syntax

```
Xrm.App.addGlobalNotification(notification).then(successCallback, errorCallback);
```

Parameters

Name	Type	Required	Description
notification	Object	Yes	<p>The notification to add. The object contains the following values:</p> <ul style="list-style-type: none">• action: (Optional) Object. Contains the following values:<ul style="list-style-type: none">◦ actionLabel: (Optional) String. The label for the action in the message.◦ eventHandler: (Optional) Function reference. The function to execute when the action label is clicked.• level: Number. Defines the level of notification. Valid values are:<ul style="list-style-type: none">◦ 1: Success◦ 2: Error◦ 3: Warning◦ 4: Information• message: String. The message to display in the notification.• showCloseButton: (Optional) Boolean. Indicates whether or not the user can close or dismiss the notification. If you don't specify this parameter, users can't close or dismiss the notification by default.• type: Number. Defines the type of notification. Currently, only a value of 2 is supported, which displays a message bar at the top of the app.
successCallback	Function	No	A function to call when notification is displayed. A GUID value is passed to uniquely identify the notification. You

can use the GUID value to close or dismiss the notification using the [clearGlobalNotification](#) method.

errorCallback	Function	No	A function to call when the operation fails.
---------------	----------	----	--

Return Value

On success, returns a promise object containing a GUID value to uniquely identify the notification as described earlier in the description of the **successCallback** parameter.

Examples

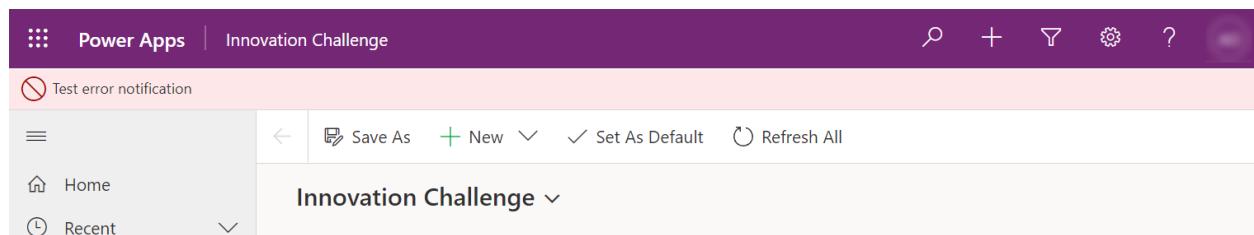
Display an error notification that can't be closed or dismissed by user

JavaScript

```
// define notification object
var notification =
{
    type: 2,
    level: 2, //error
    message: "Test error notification"
}

Xrm.App.addGlobalNotification(notification).then(
    function success(result) {
        console.log("Notification created with ID: " + result);
        // perform other operations as required on notification display
    },
    function (error) {
        console.log(error.message);
        // handle error conditions
    }
);
```

This is how the error notification will appear in the app:



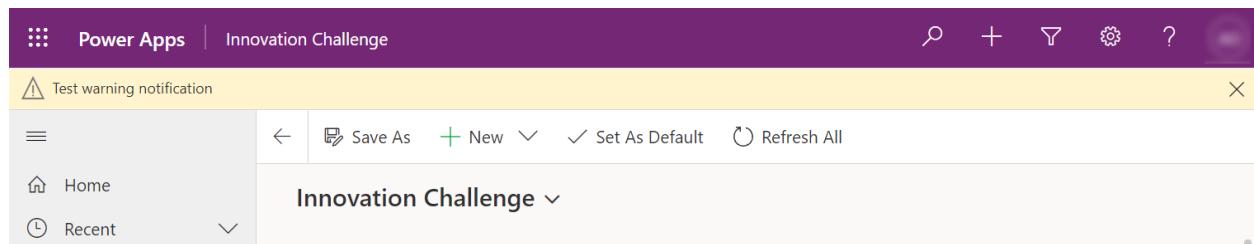
Display a warning notification that can be closed or dismissed by user

JavaScript

```
// define notification object
var notification =
{
    type: 2,
    level: 3, //warning
    message: "Test warning notification",
    showCloseButton: true
}

Xrm.App.addGlobalNotification(notification).then(
    function success(result) {
        console.log("Notification created with ID: " + result);
        // perform other operations as required on notification display
    },
    function (error) {
        console.log(error.message);
        // handle error conditions
    }
);
```

This is how the warning notification will appear in the app:



Display an information notification with a "Learn more" link that can be clicked by users

JavaScript

```
// define action object
var myAction =
{
    actionLabel: "Learn more",
    eventHandler: function () {
        Xrm.Navigation.openUrl("https://learn.microsoft.com/powerapps/");
        // perform other operations as required on clicking
    }
}
```

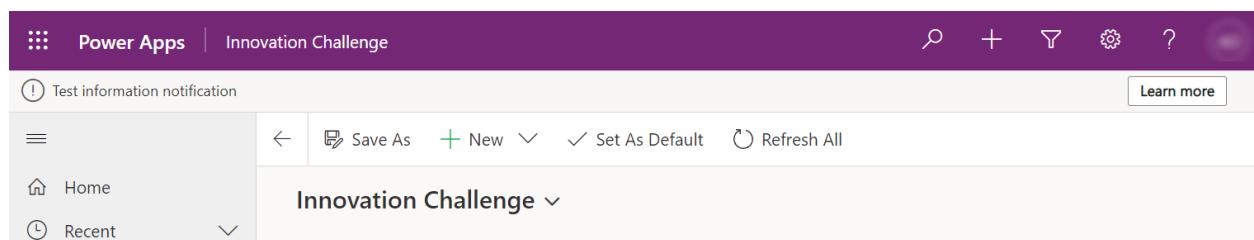
```

// define notification object
var notification =
{
    type: 2,
    level: 4, // information
    message: "Test information notification",
    action: myAction
}

Xrm.App.addGlobalNotification(notification).then(
    function success(result) {
        console.log("Notification created with ID: " + result);
        // perform other operations as required on notification display
    },
    function (error) {
        console.log(error.message);
        // handle error conditions
    }
);

```

This is how the information notification will appear in the app:



See also

[clearGlobalNotification](#)

clearGlobalNotification (Client API reference)

Article • 11/30/2022

Clears a notification in the app.

Syntax

```
Xrm.App.clearGlobalNotification(uniqueId).then(successCallback, errorCallback);
```

Parameters

Name	Type	Required	Description
uniqueId	String	Yes	The ID to use to clear a specific notification that was set using addGlobalNotification .
successCallback	Function	No	A function to call when the notification is cleared.
errorCallback	Function	No	A function to call when the operation fails.

Return Value

On success, returns a promise object.

Examples

The following example shows how to add a notification and then close it automatically after 5 seconds.

JavaScript

```
// define notification object
var notification =
{
  type: 2,
  level: 3, //warning
  message: "Test warning notification"
}

Xrm.App.addGlobalNotification(notification).then(
  function success(result) {
```

```
    console.log("Notification created with ID: " + result);

    // Wait for 5 seconds and then clear the notification
    window.setTimeout(function () {
        Xrm.App.clearGlobalNotification(result); }, 5000);
    },
    function (error) {
        console.log(error.message);
        // handle error conditions
    }
);
```

See also

[addGlobalNotification](#)

AppSidePane (Client API reference)

Article • 08/11/2023

Provides methods for managing a single side pane.

Methods

Methods	Description
close	Closes the side pane and removes it from the side bar.
select	Specify whether the pane should be selected or expanded.
navigate	Opens a page within the selected pane. This method accepts the same parameters as the navigateTo method.

Properties

Property Name	Type	Required	Description
title	string	No	The title of the pane. Used in pane header and for tooltip.
panelId	string	No	The ID of the new pane. If the value is not passed, the ID value is auto-generated.
canClose	Boolean	No	Whether the pane header will show a close button or not.
imageSrc	string	No	The path of the icon to show in the panel switcher control.
width	Number	No	The width of the pane in pixels.
hidden	Boolean	No	Hides the pane and tab.
alwaysRender	Boolean	No	Prevents the pane from unmounting when it is hidden.
keepBadgeOnSelect	Boolean	No	Prevents the badge from getting cleared when the pane becomes selected.

Related topics

[Creating side panes using client API](#)

sidePanes (Client API reference)

Article • 08/16/2023

Provides methods for managing side panes.

ⓘ Note

This API is not available for Dynamics 365 Customer Engagement on-premise deployments. More information: [Client APIs not supported in Customer Engagement \(on-premises\)](#)

Methods

Methods	Description
createPane	Add empty pane to sidePanes collection. Need to call <code>pane.navigate()</code> to load the page.
getAllPanes	Returns a collection containing all active panes.
getSelectedPane	Returns the current selected pane.
getPane	Returns the pane corresponding to the input ID. If pane doesn't exist, undefined is returned.

Parameter

Parameter Name	Description
<code>state</code>	Returns whether the selected pane is collapsed or expanded.

Related topics

[Creating side panes using client API](#)

createPane (Client API reference)

Article • 11/30/2022

Provides all the information to create side panes.

Syntax

```
Xrm.App.sidePanes.createPane(paneOptions);
```

paneOptions object

The `paneOptions` object has the following values:

Parameter Name	Type	Required	Description
title	string	No	The title of the pane. Used in pane header and for tooltip.
panelId	string	No	The ID of the new pane. If the value is not passed, the ID value is auto-generated.
canClose	Boolean	No	Whether the pane header will show a close button or not.
imageSrc	string	No	The path of the icon to show in the panel switcher control.
hideHeader	Boolean	No	Hides the header pane, including the title and close button. Default value is false.
isSelected	Boolean	No	When set to false, the created pane is not selected and leaves the existing pane selected. It also does not expand the pane if collapsed.
width	Number	No	The width of the pane in pixels.
hidden	Boolean	No	Hides the pane and tab.
alwaysRender	Boolean	No	Prevents the pane from unmounting when it is hidden.
keepBadgeOnSelect	Boolean	No	Prevents the badge from getting cleared when the pane becomes selected.

Related topics

sidePanes

[Creating side panes using client API](#)

getAllPanes (Client API reference)

Article • 11/30/2022

Returns a collection containing all active panes.

Syntax

```
Xrm.App.sidePanes.getAllPanes();
```

Related topics

[sidePanes](#)

[Creating side panes using client API](#)

getPane (Client API reference)

Article • 08/21/2023

Returns the side pane corresponding to the input ID. Returns null if the side pane does not exist.

Syntax

```
Xrm.App.sidePanes.getPane(panelId);
```

Return value

Returns the [AppSidePane](#) object.

Related topics

[sidePanes](#)

[Creating side panes using client API](#)

getSelectedPane (Client API reference)

Article • 11/30/2022

Returns the currently selected pane.

Syntax

```
Xrm.App.sidePanes.getSelectedPane();
```

Return value

Returns the [AppSidePane](#) object.

Related topics

[sidePanes](#)

[Creating side panes using client API](#)

Xrm.Device (Client API reference)

Article • 11/30/2022

Provides methods to use native device capabilities.

Method	Description
captureAudio	Invokes the device microphone to record audio.
captureImage	Invokes the device camera to capture an image.
captureVideo	Invokes the device camera to record video.
getBarcodeValue	Invokes the device camera to scan the barcode information, such as a product number.
getCurrentPosition	Returns the current location using the device geolocation capability.
pickFile	Opens a dialog box to select files for the upload.

Related articles

[Client API Xrm object](#)

captureAudio (Client API reference)

Article • 11/30/2022

Invokes the device microphone to record audio.

Available for

This method is supported only for the mobile clients.

Syntax

```
Xrm.Device.captureAudio().then(successCallback, errorCallback)
```

Parameters

Parameter	Type	Required	Description
Name			
successCallback	Function	Yes	A function to call when audio is returned. A base64 encoded audio object with the following values is passed to the function: - fileContent : Contents of the audio file. String. - fileName : Name of the audio file. String. - fileSize : Size of the audio file in KB. Number. - mimeType : Audio file MIME type. String.
errorCallback	Function	Yes	A function to call when the operation fails.

Return Value

On success, returns a base64 encoded audio object with the values specified earlier.

Exceptions

See [Web service error codes](#)

Related topics

[Xrm.Device](#)

captureImage (Client API reference)

Article • 11/30/2022

Invokes the device camera to capture an image.

Available for

This method is supported only for the mobile clients.

Syntax

```
Xrm.Device.captureImage(imageOptions).then(successCallback, errorCallback)
```

Parameters

Parameter	Type	Required	Description
Name			
imageOptions	Object	No	An object with the following values: - allowEdit : Indicates whether to edit the image before saving. Boolean. - quality : Quality of the image file in percentage. Number. - height : Height of the image to capture. Number. - width : Width of the image to capture. Number. Note: Both the height and width dimensions must be specified if used.
successCallback	Function	Yes	A function to call when image is returned. A base64 encoded image object with the following values is passed to the function: - fileContent : Contents of the image file. String - fileName : Name of the image file. String - fileSize : Size of the image file in KB. Number - mimeType : Image file MIME type. String
errorCallback	Function	Yes	A function to call when the operation fails.

Return Value

On success, returns a base64 encoded image object with the values specified earlier.

Exceptions

See [Web service error codes](#)

Related topics

[Xrm.Device](#)

captureVideo (Client API reference)

Article • 11/30/2022

Invokes the device camera to record video.

Available for

This method is supported only for the mobile clients.

Syntax

```
Xrm.Device.captureVideo().then(successCallback, errorCallback)
```

Parameters

Parameter	Type	Required	Description
Name			
successCallback	Function	Yes	A function to call when Video is returned. A base64 encoded Video object with the following values is passed to the function: - fileContent : Contents of the Video file. String. - fileName : Name of the Video file. String. - fileSize : Size of the Video file in KB. Number. - mimeType : Video file MIME type. String.
errorCallback	Function	Yes	A function to call when the operation fails.

Return Value

On success, returns a base64 encoded Video object with the values specified earlier.

Exceptions

See [Web service error codes](#)

Related topics

[Xrm.Device](#)

getBarcodeValue (Client API reference)

Article • 11/30/2022

Invokes the device camera to scan the barcode information, such as a product number.

Available for

This method is supported only for the mobile clients.

Syntax

```
Xrm.Device.getBarcodeValue().then(successCallback, errorCallback)
```

Parameters

Parameter	Type	Required	Description
Name			
successCallback	Function	Yes	A function to call when the barcode value is returned as a String.
errorCallback	Function	Yes	A function to call when the operation fails. An error object with the message property (String) will be passed that describes the error details.

Return Value

On success, returns a string containing the scanned barcode value.

Exceptions

See [Web service error codes](#)

Example

JavaScript

```
Xrm.Device.getBarcodeValue().then(  
    function success(result) {
```

```
        Xrm.Navigation.openAlertDialog({ text: "Barcode value: " + result
    });
},
function (error) {
    Xrm.Navigation.openAlertDialog( {text: error.message} );
}
);
```

Related topics

[Xrm.Device](#)

getCurrentPosition (Client API reference)

Article • 11/30/2022

Returns the current location using the device geolocation capability.

Available for

This method is supported only for the mobile clients.

Syntax

```
Xrm.Device.getCurrentPosition().then(successCallback, errorCallback)
```

Parameters

Parameter Name	Type	Required	Description
successCallback	Function	Yes	A function to call when the current geolocation information is returned. A geolocation object with the following values is passed to the function.: - coords : Contains a set of geographic coordinates along with associated accuracy as well as a set of other optional values such as altitude and speed. - timestamp : Represents the time when the object was acquired and is represented as DOMTimeStamp.
errorCallback	Function	Yes	A function to call when the operation fails. An object with the following properties will be passed: - code : The error code. Number. - message : RLocalized message describing the error details. String. If the user location setting is not enabled on your mobile device, the error message indicates the same. If you are using an earlier version of the model-driven apps mobile client or if geolocation capability is not available on your mobile device, null is passed to the error callback.

Return Value

On success, returns a geolocation object with the values specified earlier in the `successCallback` function.

Exceptions

See [Web service error codes](#)

Remarks

For the `getCurrentPosition` method to work, the geolocation capability must be enabled on your mobile device, and the model-driven apps mobile clients must have permissions to access the device location, which isn't enabled by default.

Example

JavaScript

```
Xrm.Device.getCurrentPosition().then(
    function success(location) {
        Xrm.Navigation.openAlertDialog({
            text: "Latitude: " + location.coords.latitude +
                ", Longitude: " + location.coords.longitude
        });
    },
    function (error) {
        Xrm.Navigation.openAlertDialog({ text: error.message });
    }
);
```

Related topics

[Xrm.Device](#)

pickFile (Client API reference)

Article • 11/30/2022

Opens a dialog box to select files for the upload.

Available for

This method is supported for both web and mobile clients.

Syntax

```
Xrm.Device.pickFile(pickFileOptions).then(successCallback, errorCallback)
```

Parameters

Parameter	Type	Required	Description
Name			
pickFileOptions	Object	No	An object with the following values: - accept : Image file types to select. Valid values are "audio", "video", or "image". String. - allowMultipleFiles : Indicates whether to allow selecting multiple files. Boolean. - maximumAllowedFileSize : Maximum size of the files(s) to be selected. Number.
successCallback	Function	Yes	A function to call when selected files are returned. An array of objects with <i>each</i> object having the following values is passed to the function: - fileContent : Contents of the file. String - fileName : Name of the file. String. - fileSize : Size of the file in KB. Number. - mimeType : File MIME type. String.
errorCallback	Function	Yes	A function to call when the operation fails.

Return Value

On success, returns a promise with array of objects as specified earlier for the `successCallback` function.

Exceptions

See [Web service error codes](#)

Related topics

[Xrm.Device](#)

Xrm.Encoding (Client API reference)

Article • 11/30/2022

Provides methods to encode and decode strings.

Method	Description
htmlAttributeEncode	Encodes the specified string so that it can be used in an HTML.
htmlDecode	Converts a string that has been HTML-encoded into a decoded string.
htmlEncode	Converts a string to an HTML-encoded string.
xmlAttributeEncode	Encodes the specified string so that it can be used in an XML.
xmlEncode	Converts a string to an XML-encoded string.

Related topics

[Client API Xrm object](#)

htmlAttributeEncode (Client API reference)

Article • 11/30/2022

Encodes the specified string so that it can be used in an HTML.

Syntax

```
Xrm.Encoding.htmlAttributeEncode(arg)
```

Parameters

Parameter Name	Type	Required	Description
arg	String	Required	String to be encoded.

Return Value

Type: String

Description: Encoded string.

Related topics

[htmEncode](#)

htmlDecode (Client API reference)

Article • 11/30/2022

Converts a string that has been HTML-encoded into a decoded string.

Syntax

```
Xrm.Encoding.htmlDecode(arg)
```

Parameters

Parameter Name	Type	Required	Description
arg	String	Required	HTML-encoded string to be decoded.

Return Value

Type: String

Description: Decoded string.

Related topics

[htmlEncode](#)

[htmlAttributeEncode](#)

htmlEncode (Client API reference)

Article • 11/30/2022

Converts a string to an HTML-encoded string.

Syntax

```
Xrm.Encoding.htmlEncode(arg)
```

Parameters

Parameter Name	Type	Required	Description
arg	String	Required	String to be encoded.

Return Value

Type: String

Description: Encoded string.

Related topics

[htmlAttributeEncode](#)

[htmlDecode](#)

XmlAttributeEncode (Client API reference)

Article • 11/30/2022

Encodes the specified string so that it can be used in an XML.

Syntax

```
Xrm.Encoding.xmlAttributeEncode(arg)
```

Parameters

Parameter Name	Type	Required	Description
arg	String	Required	String to be encoded.

Return Value

Type: String

Description: Encoded string.

Related topics

[xmlEncode](#)

xmlEncode (Client API reference)

Article • 11/30/2022

Converts a string to an XML-encoded string.

Syntax

```
Xrm.Encoding.xmlEncode(arg)
```

Parameters

Parameter Name	Type	Required	Description
arg	String	Required	String to be encoded.

Return Value

Type: String

Description: Encoded string.

Related topics

[xmlAttributeEncode](#)

Xrm.Navigation (Client API reference)

Article • 11/30/2022

Provides navigation-related methods.

Method	Description
navigateTo	Navigates to the specified table list, table record, HTML web resource, or custom page.
openAlertDialog	Displays an alert dialog containing a message and a button.
openConfirmDialog	Displays a confirmation dialog box containing a message and two buttons.
openErrorDialog	Displays an error dialog.
openFile	Opens a file.
openForm	Opens an entity form or a quick create form.
openUrl	Opens a URL, including file URLs.
openWebResource	Opens an HTML web resource in a new window.

Related topics

[Client API Xrm object](#)

navigateTo (Client API reference)

Article • 11/30/2022

Navigates to the specified table list, table record, HTML web resource, or custom page.

ⓘ Note

This method is supported only on Unified Interface.

Syntax

```
Xrm.Navigation.navigateTo(pageInput, navigationOptions).then(successCallback,errorCallback);
```

Parameters

Name	Type	Required	Description
pageInput	Object	Yes	Input about the page to navigate to. The object definition changes depending on the type of page to navigate to: entity list , entity record , dashboard , HTML web resource , or custom page .
navigationOptions	Object	No	Options for navigating to a page: whether to open inline or in a dialog. If you don't specify this parameter, page is opened inline by default.
successCallback	function	No	A function to execute on successful navigation to the page when navigating inline and on closing the dialog when navigating to a dialog.
errorCallback	Function	No	A function to execute when the operation fails.

pageInput parameter

Entity list

The entity list object contains the following values.

Name	Type	Description
pageType	String	Specify "entitylist".
entityName	String	The logical name of the table to load in the list control.
viewId	String	(Optional) The ID of the view to load. If you don't specify it, navigates to the default main view for the table.
viewType	String	(Optional) Type of view to load. Specify "savedquery" or "userquery".

Entity record

The entity record object contains the following values.

Name	Type	Description
pageType	String	Specify "entityrecord".
entityName	String	Logical name of the table to display the form for.
entityId	String	(Optional) ID of the table record to display the form for. If you don't specify this value, the form will be opened in create mode.
createFromEntity	Lookup	(Optional) Designates a record that will provide default values based on mapped column values. The lookup object has the following String properties: entityType, id, and name (optional).
data	Object	(Optional) A dictionary object that passes extra parameters to the form. The parameters can be table columns with default values that are set on new forms (see Set column values using parameters passed to a form), or custom parameters that are accessed on the form using <code>formContext.data.attributes</code> (see Configure a form to accept custom querystring parameters , and formContext.data). Invalid parameters will cause an error.
formId	String	(Optional) ID of the form instance to be displayed.
isCrossEntityNavigate	Boolean	(Optional) Indicates whether the form is navigated to from a different table using cross-table business process flow.
isOfflineSyncError	Boolean	(Optional) Indicates whether there are any offline sync errors.
processId	String	(Optional) ID of the business process to be displayed on the form.
processInstanceId	String	(Optional) ID of the business process instance to be displayed on the form.

Name	Type	Description
relationship	Object	(Optional) Define a relationship object to display the related records on the form.
selectedStageId	String	(Optional) ID of the selected stage in business process instance.
tabName	String	(Optional) Sets the focus on the tab of the form.

Relationship object

The relationship object, used in the [Entity record](#), contains the following values.

Name	Type	Description
attributeName	String	Name of the column used for relationship.
name	String	Name of the relationship.
navigationPropertyName	String	Name of the navigation property for this relationship.
relationshipType	Number	Relationship type. Specify one of the following values: 0:OneToMany, 1:ManyToMany.
roleType	Number	Role type in relationship. Specify one of the following values: 1:Referencing, 2:AssociationEntity.

Dashboard

The dashboard object contains the following values.

Name	Type	Description
pageType	String	Specify "dashboard".
dashboardId	String	The ID of the dashboard to load. If you don't specify the ID, navigates to the default dashboard.

HTML web resource

The HTML web resource object contains the following values.

Name	Type	Description
pageType	String	Specify "webresource".

Name	Type	Description
webresourceName	String	The name of the web resource to load.
data	String	(Optional) The data to pass to the web resource.

Custom page

The Custom page object contains the following values.

Name	Type	Description
pageType	String	Specify "custom".
name	String	The logical name of the custom page to open.
entityName	String	(Optional) The logical name of the table to be made available in the custom page via Param("entityName").
recordId	String	(Optional) ID of the table record to be made available in the custom page via Param("recordId").

navigationOptions parameter

The navigationOptions object contains the following values.

Name	Type	Description
target	Number	Specify 1 to open the page inline; 2 to open the page in a dialog. Also, rest of the values (width, height, and position) are valid only if you have specified 2 in this value (open page in a dialog). Note: Entity lists can only be opened inline; entity records and web resources can be opened either inline or in a dialog.
width	Number or Object	(Optional) The width of dialog. To specify the width in pixels, just type a numeric value. To specify the width in percentage, specify an object of type SizeValue with the following properties: <ul style="list-style-type: none"> • value: The numerical value of type Number. • unit: The unit of measurement of type String. Specify "%" or "px". Default value is "px".
height	Number or Object	(Optional) The height of dialog. To specify the height in pixels, just type a numeric value. To specify the width in percentage, specify an object of type SizeValue with the following properties: <ul style="list-style-type: none"> • value: The numerical value of type Number. • unit: The unit of measurement of type String. Specify "%" or "px". Default value is "px".

Name	Type	Description
position	Number	(Optional) Specify 1 to open the dialog in center; 2 to open the dialog on the far side. Default is 1 (center).
title	String	(Optional) The dialog title on top of the center or side dialog.

Return Value

Returns a promise. The value passed when the promise resolves is dependent on the target:

- *inline*: Promise resolves right away, and does not return any value.
- *dialog*: Promise resolves when the dialog is closed. An object is passed only if the **pageType = entityRecord** and you opened the form in create mode. The object has a **savedEntityReference** array with the following properties to identify the table record created:
 - **entityType**: The logical name of the table.
 - **id**: A string representation of a GUID value for the record.
 - **name**: The primary column value of the record displayed or created.

Example

Example 1: Open account list

JavaScript

```
var pageInput = {
  pageType: "entitylist",
  entityName: "account"
};
Xrm.Navigation.navigateTo(pageInput).then(
  function success() {
    // Run code on success
  },
  function error() {
    // Handle errors
  }
);
```

Example 2: Open an existing account record within a dialog

JavaScript

```
var pageInput = {
    pageType: "entityrecord",
    entityName: "account",
    entityId: "5a57f2c3-5672-ea11-a812-000d3a339706" //replace with actual
ID
};
var navigationOptions = {
    target: 2,
    height: {value: 80, unit:"%"},
    width: {value: 70, unit:"%"},
    position: 1
};
Xrm.Navigation.navigateTo(pageInput, navigationOptions).then(
    function success() {
        // Run code on success
    },
    function error() {
        // Handle errors
    }
);
```

Example 3: Open an account form in the create mode within a dialog

JavaScript

```
var pageInput = {
    pageType: "entityrecord",
    entityName: "account"
};
var navigationOptions = {
    target: 2,
    height: {value: 80, unit:"%"},
    width: {value: 70, unit:"%"},
    position: 1
};
Xrm.Navigation.navigateTo(pageInput, navigationOptions).then(
    function success(result) {
        console.log("Record created with ID: " +
result.savedEntityReference[0].id +
            " Name: " + result.savedEntityReference[0].name)
        // Handle dialog closed
    },
    function error() {
        // Handle errors
    }
);
```

Example 4: Open an HTML web resource in a dialog

JavaScript

```
var pageInput = {
    pageType: "webresource",
    webresourceName: "new_sample_webresource.htm"
};
var navigationOptions = {
    target: 2,
    width: 500, // value specified in pixel
    height: 400, // value specified in pixel
    position: 1
};
Xrm.Navigation.navigateTo(pageInput, navigationOptions).then(
    function success() {
        // Run code on success
    },
    function error() {
        // Handle errors
    }
);
```

Related topics

[Xrm.Navigation](#)

[Navigating to and from a custom page \(preview\)](#)

openAlertDialog (Client API reference)

Article • 12/16/2022

Displays an alert dialog containing a message and a button.

Syntax

```
Xrm.Navigation.openAlertDialog(alertStrings,alertOptions).then(closeCallback,errorC  
allback);
```

Parameters

Name	Type	Required	Description
alertStrings	Object	Yes	The strings to be used in the alert dialog. The object contains the following values: <ul style="list-style-type: none">- confirmButtonLabel: (Optional) String. The confirm button label. If you do not specify the button label, OK is used as the button label.- text: String. The message to be displayed in the alert dialog.- title: (Optional) String. The title of the alert dialog.
alertOptions	Object	No	The height and width options for alert dialog. The object contains the following values: <ul style="list-style-type: none">- height: (Optional) Number. Height of the alert dialog in pixels.- width: (Optional) Number. Width of the alert dialog pixels.
successCallback	function	No	A function to execute when the alert dialog is closed by either clicking the confirm button or canceled by pressing ESC.
errorCallback	function	No	A function to execute when the operation fails.

Example

The following sample code displays an alert dialog. Clicking Yes button in the alert dialog or canceling the alert dialog by pressing ESC calls the `close` function::

JavaScript

```
var alertStrings = { confirmButtonLabel: "Yes", text: "This is an alert.",  
title: "Sample title" };  
var alertOptions = { height: 120, width: 260 };  
Xrm.Navigation.openAlertDialog(alertStrings, alertOptions).then(  
    function (success) {  
        console.log("Alert dialog closed");  
    },  
    function (error) {  
        console.log(error.message);  
    }  
);
```

Related topics

[Xrm.navigation](#)

openConfirmDialog (Client API reference)

Article • 12/16/2022

Displays a confirmation dialog box containing a message and two buttons.

Syntax

```
Xrm.Navigation.openConfirmDialog(confirmStrings,confirmOptions).then(successCallback,  
errorCallback);
```

Parameters

Name	Type	Required	Description
confirmStrings	Object	Yes	<p>The strings to be used in the confirmation dialog. The object contains the following values:</p> <ul style="list-style-type: none">- cancelButtonLabel: (Optional) String. The cancel button label. If you do not specify the cancel button label, Cancel is used as the button label.- confirmButtonLabel: (Optional) String. The confirm button label. If you do not specify the confirm button label, OK is used as the button label.- subtitle: (Optional) String. The subtitle to be displayed in the confirmation dialog.- text: String. The message to be displayed in the confirmation dialog.- title: (Optional) String. The title to be displayed in the confirmation dialog.
confirmOptions	Object	No	<p>The height and width options for confirmation dialog. The object contains the following values:</p> <ul style="list-style-type: none">- height: (Optional) Number. Height of the confirmation dialog in pixels.- width: (Optional) Number. Width of the confirmation dialog in pixels.
successCallback	function	No	A function to execute when the confirmation dialog is closed by clicking the confirm, cancel, or X in the top-right corner of the dialog. An object with the confirmed (Boolean) attribute is passed that indicates whether the confirm button was clicked to close the dialog.

Name	Type	Required	Description
errorCallback	function	No	A function to execute when the operation fails.

Example

The following code sample displays a confirmation dialog box. Appropriate message is logged in the console depending on whether confirm or cancel/X was clicked to close the dialog.

JavaScript

```
var confirmStrings = { text:"This is a confirmation.", title:"Confirmation Dialog" };
var confirmOptions = { height: 200, width: 450 };
Xrm.Navigation.openConfirmDialog(confirmStrings, confirmOptions).then(
    function (success) {
        if (success.confirmed)
            console.log("Dialog closed using OK button.");
        else
            console.log("Dialog closed using Cancel button or X.");
});
```

Related topics

[Xrm.Navigation](#)

openErrorDialog (Client API reference)

Article • 12/16/2022

Displays an error dialog.

Syntax

```
Xrm.Navigation.openErrorDialog(errorOptions).then(successCallback,errorCallback);
```

Parameters

Name	Type	Required	Description
errorOptions	Object	Yes	An object to specify the options for error dialog. The object contains the following values: - details : (Optional) String. Details about the error. When you specify this, the Download Log File button is available in the error message, and clicking it will let users download a text file with the content specified in this value. - errorCode : (Optional) Number. The error code. If you just set errorCode , the message for the error code is automatically retrieved from the server and displayed in the error dialog. If you specify an invalid errorCode value, an error dialog with a default error message is displayed. - message : (Optional) String. The message to be displayed in the error dialog. You must set either the errorCode or message value.
successCallback	function	No	A function to execute when the error dialog is closed.
errorCallback	function	No	A function to execute when the operation fails.

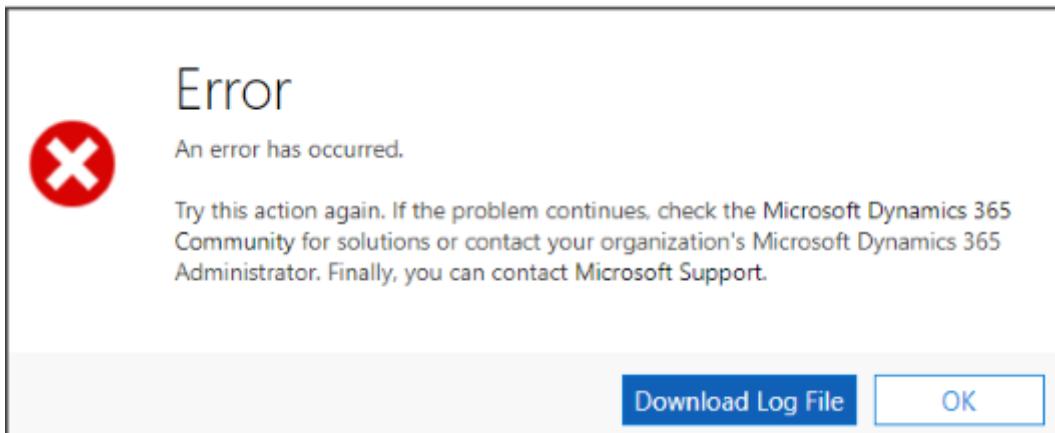
Example

The following code sample passes an incorrect errorCode (1234) to display an error dialog with default message:

```
JavaScript
```

```
Xrm.Navigation.openErrorDialog({ errorCode:1234 }).then(  
    function (success) {  
        console.log(success);  
    },  
    function (error) {  
        console.log(error);  
    });
```

This displays an error dialog with the default message:



Related topics

[Xrm.Navigation](#)

[List of error codes](#)

openFile (Client API reference)

Article • 11/30/2022

Opens a file.

Syntax

```
Xrm.Navigation.openFile(file,openFileOptions)
```

Parameters

Parameter	Type	Required	Description
Name			
file	Object	Yes	An object describing the file to open. The object has the following values: - fileContent : String. Contents of the file. - fileName : String. Name of the file. - fileSize : Number. Size of the file in KB. - mimeType : String. MIME type of the file.
openFileOptions	Object	No	An object describing whether to open or save the file. The object has the following values: - openMode : Specify 1 to open; 2 to save. If you do not specify this parameter, by default 1 (open) is passed. This parameter is only supported on Unified Interface.

Related topics

[Xrm.Navigation](#)

openForm (Client API reference)

Article • 01/03/2023

Opens an entity form or a quick create form.

① Note

To open a main form as a dialog, use the `navigateTo` method instead. More information:
[Open main form in a dialog using client API](#)

Syntax

```
Xrm.Navigation.openForm(entityFormOptions, formParameters).then(successCallback,  
errorCallback);
```

Parameters

Name	Type	Required	Description
entityFormOptions	Object	Yes	<p>Form options for opening the form. The object contains the following values:</p> <ul style="list-style-type: none">• entityName: String. Logical name of the table to display the form for.• entityId: (Optional) String. ID of the table record to display the form for.• formId: (Optional) String. ID of the form instance to be displayed.• cmdbar: (Optional) Boolean. Indicates whether to display the command bar. If you do not specify this parameter, the command bar is displayed by default. Requires passing <code>openInNewWindow</code> parameter as true.• createFromEntity: (Optional) Lookup. Designates a record that will provide default values based on mapped column values. The lookup object has the following String properties: <code>entityType</code>, <code>id</code>, and <code>name</code> (optional).• openInNewWindow: (Optional) Boolean. Indicates whether to display form in a new window or a new tab. If you specify <code>true</code> and do not specify values for height or width, the form will display in a new tab. Opening a form in a new window or a new tab makes the rendering of the form slow compared to opening the form on the same tab; consider opening a form in the main form dialog instead. This property is currently not supported for Quick Create forms, as they can not be opened in a new window or tab.• height: (Optional) Number. Height of the form window to be displayed in pixels. Requires passing <code>openInNewWindow</code> parameter

as true.

- **width:** (Optional) Number. Width of the form window to be displayed in pixels. Requires passing `openInNewWindow` parameter as true.
- **navbar:** (Optional) String. Controls whether the navigation bar is displayed and whether application navigation is available using the areas and subareas defined in the sitemap. Valid values are: `on`, `off`, or `entity`. Requires passing `openInNewWindow` parameter as true.
 - `on`: The navigation bar is displayed. This is the default behavior if the **navbar** parameter is not used.
 - `off`: The navigation bar is not displayed. People can navigate using other user interface elements or the back and forward buttons.
 - `entity`: On a form, only the navigation options for related tables are available. After navigating to a related table, a back button is displayed in the navigation bar to allow returning to the original record.
- **relationship:** (Optional) Object. Define a relationship object to display the related records on the form. The object has the following values.

Name	Type	Description
attributeName	String	Name of the column used for relationship.
name	String	Name of the relationship.
navigationPropertyName	String	Name of the navigation property for this relationship.
relationshipType	Number	Relationship type. Specify one of the following values: <ul style="list-style-type: none">◦ <code>0:OneToMany</code>◦ <code>1:ManyToOne</code>
roleType	Number	Role type in relationship. Specify one of the following values: <ul style="list-style-type: none">◦ <code>1:Referencing</code>◦ <code>2:AssociationEntity</code>

- **selectedStageId:** (Optional) String. ID of the selected stage in business process instance.
- **useQuickCreateForm:** (Optional) Boolean. Indicates whether to open a quick create form. The table must have the **Allow Quick Create** option enabled for the quick create form to be displayed and you must also add the table and the quick create form to your app. If you do not specify the value of `useQuickCreateForm`, the default will be set to `false`.

formParameters	Object	No	A dictionary object that passes extra parameters to the form. Invalid parameters will cause an error. For information about passing parameters to a form, see Set column values using parameters passed to a form and Configure a form to accept custom querystring parameters
successCallback	Function	No	A function to execute when the record is saved in the quick create form. This function is passed an object as a parameter. The object has a savedEntityReference array with the following properties to identify the record(s) displayed or created: <ul style="list-style-type: none"> • entityType: The logical name of the table. • id: A string representation of a GUID value for the record. • name: The primary column value of the record displayed or created. <p>NOTE:</p> <ul style="list-style-type: none"> • The successCallback function is not executed when you open a form for an existing or new record. • The successCallback function is executed only when you save a record in a quick create form that was opened using the openForm method.
errorCallback	Function	No	A function to execute when the operation fails.

Remarks

You must use this method to open table or quick create forms instead of the deprecated [Xrm.Utility.openEntityForm](#) and [Xrm.Utility.openQuickCreate](#) methods.

Use [setActiveProcess](#) to display a particular business process and [setActiveProcessInstance](#) to display a particular business process instance on the form.

Examples

Example 1: Open a form for existing record

The following sample code opens a contact form to display an existing contact record:

JavaScript

```
var entityFormOptions = {};
entityFormOptions["entityName"] = "contact";
entityFormOptions["entityId"] = "8DA6E5B9-88DF-E311-B8E5-6C3BE5A8B200";

// Open the form.
Xrm.Navigation.openForm(entityFormOptions).then(
    function (success) {
```

```
        console.log(success);
    },
    function (error) {
        console.log(error);
});
});
```

Example 2: Open a form for new record

The following sample code opens a contact form with some pre-populated values to create a new record:

JavaScript

```
var entityFormOptions = {};
entityFormOptions["entityName"] = "contact";

// Set default values for the Contact form
var formParameters = {};
formParameters["firstname"] = "Sample";
formParameters["lastname"] = "Contact";
formParameters["fullname"] = "Sample Contact";
formParameters["emailaddress1"] = "contact@adventure-works.com";
formParameters["jobtitle"] = "Sr. Marketing Manager";
formParameters["donotemail"] = "1";
formParameters["description"] = "Default values for this record were set
programmatically.";

// Set lookup column
formParameters["preferreddsystemuserid"] = "3493e403-fc0c-eb11-a813-002248e258e0"; // 
ID of the user.
formParameters["preferreddsystemuseridname"] = "Admin user"; // Name of the user.
// End of set lookup column

// Open the form.
Xrm.Navigation.openForm(entityFormOptions, formParameters).then(
    function (success) {
        console.log(success);
    },
    function (error) {
        console.log(error);
});
});
```

Example 3: Open a form for new record (complex lookup)

The following sample code opens a activity form with some pre-populated values (including a complex lookup) to create a new record:

JavaScript

```
var entityFormOptions = {};
entityFormOptions["entityName"] = "email";

// Set default values for the Contact form
var formParameters = {};
```

```

formParameters["subject"] = "Sample";
formParameters["description"] = "Default values for this record were set
programmatically.";

// Set lookup column
formParameters["regardingobjectid"] = "3493e403-fc0c-eb11-a813-002248e258e0"; // ID
of the user.
formParameters["regardingobjectidname"] = "Admin user"; // Name of the user.
formParameters["regardingobjectidtype"] = "systemuser"; // Table name.
// End of set lookup column

// Open the form.
Xrm.Navigation.openForm(entityFormOptions, formParameters).then(
    function (success) {
        console.log(success);
    },
    function (error) {
        console.log(error);
});

```

Example 4: Open a quick create form

The following sample code opens a quick create contact form with some pre-populated values:

JavaScript

```

var entityFormOptions = {};
entityFormOptions["entityName"] = "contact";
entityFormOptions["useQuickCreateForm"] = true;

// Set default values for the Contact form
var formParameters = {};
formParameters["firstname"] = "Sample";
formParameters["lastname"] = "Contact";
formParameters["fullname"] = "Sample Contact";
formParameters["emailaddress1"] = "contact@adventure-works.com";
formParameters["jobtitle"] = "Sr. Marketing Manager";
formParameters["donotemail"] = "1";
formParameters["description"] = "Default values for this record were set
programmatically.";

// Set lookup column
formParameters["preferreddsystemuserid"] = "3493e403-fc0c-eb11-a813-002248e258e0"; // ID
of the user.
formParameters["preferreddsystemuseridname"] = "Admin user"; // Name of the user.
formParameters["preferreddsystemuseridtype"] = "systemuser"; // Table name.
// End of set lookup column

// Open the form.
Xrm.Navigation.openForm(entityFormOptions, formParameters).then(
    function (success) {
        console.log(success);
    },
    function (error) {
        console.log(error);
});

```

Related topics

[Xrm.Navigation](#)

openUrl (Client API reference)

Article • 11/30/2022

Opens a URL, including file URLs.

Syntax

```
Xrm.Navigation.openUrl(url,openUrlOptions)
```

Parameters

Name	Type	Required	Description
url	String	Yes	URL to open.
openUrlOptions	Object	No	Options to open the URL. The object contains the following values: <ul style="list-style-type: none">- height: (Optional) Number. Height of the window to display the resultant page in pixels.- width: (Optional) Number. Width of the window to display the resultant page in pixels.

Remarks

This method is especially helpful for mobile clients to open a URL in a browser outside of shim.

Related topics

[Xrm.Navigation](#)

openWebResource (Client API reference)

Article • 12/16/2022

Opens an HTML web resource in a new window.

Syntax

```
Xrm.Navigation.openWebResource(web resourceName, windowOptions, data)
```

Parameters

Name	Type	Required	Description
web resourceName	String	Yes	Name of the HTML web resource to open.
windowOptions	Object	No	Window options for opening the web resource. The object contains the following values: - height : (Optional) Number. Height of the window to open in pixels. - width : (Optional) Number. Width of the window to open in pixels.
data	String	No	Data to be passed into the data parameter.

Remarks

You must use this method to display web resources instead of the deprecated [Xrm.Utility.openWebResource](#) method.

An HTML web resource can accept the parameter values described in [Pass parameters to HTML web resources](#). This function only provides for passing in the optional data parameter. To pass values for the other valid parameters, you must append them to the `web resourceName` parameter.

Note

The `Xrm` object isn't available in HTML web resources. Therefore, scripts containing `xrm.*` methods aren't supported in HTML web resources. `parent.Xrm.*` will work if the HTML web resource is loaded in a form container. However, for other places,

such as loading an HTML web resource as part of the SiteMap, `parent.Xrm.*` also won't work. More information: [GetGlobalContext function and ClientGlobalContext.js.aspx](#)

Examples

- Open an HTML web resource named "new_webResource.htm":

```
Xrm.Navigation.openWebResource("new_webResource.htm");
```

- Open an HTML web resource, setting the windowOptions:

JavaScript

```
var windowOptions = { height: 400, width: 400 };
Xrm.Navigation.openWebResource("new_webResource.htm",windowOptions);
```

- Open an HTML web resource including a single item of data for the `data` parameter

```
Xrm.Navigation.openWebResource("new_webResource.htm",null,"dataItemValue");
```

Related topics

[Xrm.Navigation](#)

Xrm.Panel

Article • 11/30/2022

Provides a method to display a web page in the side pane of model-driven apps form.

Method	Description
loadPanel	Displays the web page represented by a URL in the static area in the side pane, which appears on all pages in the model-driven apps web client.

ⓘ Note

The **Xrm.Panel** namespace was introduced in the December 2016 update for Dynamics 365 (online and on-premises), and the method under this namespace is a preview feature. A preview feature is a feature that is not complete, but is made available before it's officially in a release so customers can get early access and provide feedback. Preview features aren't meant for production use and may have limited or restricted functionality. We expect changes to this feature, so you shouldn't use it in production. Use it only in test and development environments. Microsoft doesn't provide support for this preview feature. Microsoft Dynamics 365 Technical Support won't be able to help you with issues or questions. Preview features aren't meant for production use and are subject to a separate [supplemental terms of use](#).

Related topics

[Client API Xrm object](#)

loadPanel (Client-side reference)

Article • 11/30/2022

Displays the web page represented by a URL in the static area in the side pane, which appears on all pages in the model-driven apps web client.

Syntax

```
Xrm.Panel.loadPanel(url, title)
```

Parameters

Parameter Name	Type	Required	Description
url	String	Yes	URL of the page to be loaded in the side pane static area.
title	String	Yes	Title of the side pane static area.

Remarks

This API is being replaced with Xrm.App.sidePanes.createPane. See [Use with Xrm.App.panels.loadPanel](#) for more details on the interaction of `loadPanel` and `createPane`.

Xrm.Utility (Client API reference)

Article • 11/30/2022

Provides a container for useful methods.

Methods

Method	Description
closeProgressIndicator	Closes a progress dialog box.
getAllowedStatusTransitions	Returns the valid state transitions for the specified table type and status code.
getEntityMetadata	Returns table definitions for the specified table.
getEntityMainFormDescriptor	Returns the default main form descriptor for the specified table.
getGlobalContext	Gets the global context.
getLearningPathAttributeName	Returns the name of the DOM attribute expected by the Learning Path (guided help) Content Designer for identifying UI controls in the model-driven apps forms.
getPageContext	Gets the page context as an object representing the page.
getResourceString	Returns the localized string for a given key associated with the specified web resource.
invokeProcessAction	Invokes an action based on the specified parameters.
lookupObjects	Opens a lookup control to select one or more items.
refreshParentGrid	Refreshes the parent grid containing the specified record.
showProgressIndicator	Displays a progress dialog with the specified message.

Deprecated methods

The following table lists the new methods you should use instead of the deprecated methods in the `Xrm.Utility` namespace. These methods were deprecated in v9.0.

Deprecated Method	New method to be used
alertDialog	<code>Xrm.Navigation.openAlertDialog</code>

Deprecated Method	New method to be used
confirmDialog	Xrm.Navigation.openConfirmDialog
getBarcodeValue	Xrm.Device.getBarcodeValue
getCurrentPosition	Xrm.Device.getCurrentPosition
openEntityForm	Xrm.Navigation.openForm
openQuickCreate	Xrm.Navigation.openForm
openWebResource	Xrm.Navigation.openWebResource

Related topics

[Client API execution context](#)

[Client API Xrm object](#)

[Client API reference](#)

[Deprecated client APIs](#)

closeProgressIndicator (Client API reference)

Article • 11/30/2022

Closes a progress dialog box.

If no progress dialog is displayed currently, this method will do nothing. You can display a progress dialog using the [showProgressIndicator](#) method.

Syntax

```
Xrm.Utility.closeProgressIndicator()
```

Related topics

[showProgressIndicator](#)

[Xrm.Utility](#)

getAllowedStatusTransitions (Client API reference)

Article • 11/30/2022

Returns the valid state transitions for the specified table type and status code.

Syntax

```
Xrm.Utility.getAllowedStatusTransitions(entityName, statusCode).then(successCallback  
, errorCallback)
```

Parameters

Name	Type	Required	Description
entityName	String	Yes	The logical name of the table.
statusCode	Number	Yes	The status code to find out the allowed status transition values.
successCallback	Function	No	The function to execute when the operation succeeds.
errorCallback	Function	No	The function to execute when the operation fails.

Return value

Returns an object with `.then()` function. The parameter to the delegate is an array of numbers representing the valid status transitions.

Related topics

[Xrm.Utility](#)

getEntityMetadata (Client API)

Article • 11/30/2022

Returns table definitions for the specified table.

Syntax

```
Xrm.Utility.getEntityMetadata(entityName,attributes).then(successCallback,  
errorCallback)
```

Parameters

Name	Type	Required	Description
entityName	String	Yes	The logical name of the table.
attributes	array of strings	No	The columns to get definitions for.
successCallback	function	No	A function to call when the table definitions are returned.
errorCallback	function	No	A function to call when the operation fails.

Returns

Type: Object

Description: An object containing the table definitions information with the following values.

Name	Type	Description
ActivityTypeMask	Number	Whether a custom activity should appear in the activity menus in the Web application. 0 indicates that the custom activity doesn't appear; 1 indicates that it does appear.
AutoRouteToOwnerQueue	Boolean	Indicates whether to automatically move records to the owner's default queue when a record of this type is created or assigned.
CanEnableSyncToExternalSearchIndex	Boolean	For internal use only.

CanTriggerWorkflow	Boolean	Indicates whether the table can trigger a workflow process.
Description	String	Description for the table.
DisplayCollectionName	String	Plural display name for the table.
DisplayName	String	Display name for the table.
EnforceStateTransitions	Boolean	Indicates whether the table will enforce custom state transitions.
EntityColor	String	The hexadecimal code to represent the color to be used for this table in the application.
EntitySetName	String	The name of the Web API table set for this table.
HasActivities	Boolean	Indicates whether activities are associated with this table.
IsActivity	Boolean	Indicates whether the table is an activity.
IsActivityParty	Boolean	Indicates whether the email messages can be sent to an email address stored in a record of this type.
IsBusinessProcessEnabled	Boolean	Indicates whether the table is enabled for business process flows.
IsBPFEntity	Boolean	Indicates whether the table is a business process flow table.
IsChildEntity	Boolean	Indicates whether the table is a child table.
IsConnectionsEnabled	Boolean	Indicates whether connections are enabled for this table.
IsCustomEntity	Boolean	Indicates whether the table is a custom table.
IsCustomizable	Boolean	Indicates whether the table is customizable.
IsDocumentManagementEnabled	Boolean	Indicates whether document management is enabled.
IsDocumentRecommendationsEnabled	Boolean	Indicates whether the document recommendations is enabled.
IsDuplicateDetectionEnabled	Boolean	Indicates whether duplicate detection is enabled.

IsEnabledForCharts	Boolean	Indicates whether charts are enabled.
IsImportable	Boolean	Indicates whether the table can be imported using the Import Wizard.
IsInteractionCentricEnabled	Boolean	Indicates the table is enabled for interactive experience.
IsKnowledgeManagementEnabled	Boolean	Indicates whether knowledge management is enabled for the table.
IsMailMergeEnabled	Boolean	Indicates whether mail merge is enabled for this table.
IsManaged	Boolean	Indicates whether the table is part of a managed solution.
IsOneNoteIntegrationEnabled	Boolean	Indicates whether OneNote integration is enabled for the table.
IsOptimisticConcurrencyEnabled	Boolean	Indicates whether optimistic concurrency is enabled for the table.
IsQuickCreateEnabled	Boolean	Indicates whether the table is enabled for quick create forms.
IsStateModelAware	Boolean	Indicates whether the table supports setting custom state transitions.
IsValidForAdvancedFind	Boolean	Indicates whether the table will be shown in Advanced Find.
IsVisibleInMobileClient	Boolean	Indicates whether Microsoft Dynamics 365 for tablets users can see data for this table.
IsEnabledInUnifiedInterface	Boolean	Indicates whether the table is enabled for Unified Interface.
LogicalCollectionName	String	The logical collection name.
LogicalName	String	The logical name for the table.
ObjectTypeCode	Number	The table type code.
OwnershipType	String	The ownership type for the table: "UserOwned" or "OrganizationOwned".
PrimaryIdAttribute	String	The name of the column that is the primary id for the table.
PrimaryImageAttribute	String	The name of the primary image column for a table.

PrimaryNameAttribute	String	The name of the primary column for a table.
Privileges	Array of objects	<p>The privilege definitions for the table where *each* object contains the following values to define the security privilege for access to a table:</p> <ul style="list-style-type: none"> • CanBeBasic: Boolean. Whether the privilege can be basic access level. • CanBeDeep: Boolean. Whether the privilege can be deep access level. • CanBeEntityReference: Boolean. Whether the privilege for an external party can be basic access level. • CanBeGlobal: Boolean. Whether the privilege can be global access level. • CanBeLocal: Boolean. Whether the privilege can be local access level. • CanBeParentEntityReference: Boolean. Whether the privilege for an external party can be parent access level. • Name: String. The name of the privilege. • PrivilegeId: String. The ID of the privilege. • PrivilegeType: Number. The type of privilege, which is one of the following: <ul style="list-style-type: none"> ◦ 0: None ◦ 1: Create ◦ 2: Read ◦ 3: Write ◦ 4: Delete ◦ 5: Assign ◦ 6: Share ◦ 7: Append ◦ 8: AppendTo
Attributes	Collection	<p>A collection of column definitions objects. The object returned depends on the type of column definitions.</p> <p>Column definitions for the <i>base</i> type</p> <p>An object returned with the following properties:</p> <ul style="list-style-type: none"> • AttributeType: Number. Type of a column. For a list of column type values, see AttributeTypeCode

- **DisplayName**: String. Display name for the column.
- **EntityLogicalName**: String. Logical name of the table that contains the column.
- **LogicalName**: String. Logical name for the column.

Column definitions for the *boolean* type

An object returned with the following properties in addition to the *base* column definitions type properties:

- **DefaultFormValue**: Boolean. Default value for a Yes/No column.
- **OptionSet**: Object. Options for the boolean column where each option is a key:value pair.

Column definitions for the *enum* type

An object returned with the following properties in addition to the *base* column definitions type properties:

- **OptionSet**: Object. Options for the column where each option is a key:value pair.

Column definitions for the *choices* type

An object returned with the following properties in addition to the *base* column definitions type properties:

- **DefaultFormValue**: Number. Default form value for the column.
- **OptionSet**: Object. Options for the column where each option is a key:value pair.

Column definitions for the *state* type

An object returned with the following properties in addition to the *base* column definitions type properties:

- **OptionSet**: Object. Options for the column where each option is a key:value pair.

The object also contains the following methods:

- **getDefaultValue(arg)**: Returns the default status (number) based on the passed in state value for a table. For default state and status values for a table, see table definitions information of the table in [table/entity reference](#).
- **getStatusValuesForState(arg)**: Returns possible status values (array of numbers) for a specified state value. For state and status values for a table, see table definitions information of the table in [table/entity reference](#).

Column definitions for the *status* type

An object returned with the following properties in addition to the *base* column definitions type properties:

- **OptionSet**: Object. Options for the column where each option is a key:value pair.

The object also contains the following method:

- **getState(arg)**: Returns the state value (number) for the specified status value (number). For default state and status values for a table, see table definitions information of the table in [table/entity reference](#).

Related topics

[Xrm.Utility](#)

getEntityMainFormDescriptor (Client API reference)

Article • 12/16/2022

Returns the default main form descriptor for the specified table.

Syntax

```
Xrm.Utility.getEntityMainFormDescriptor(entityName, formId);
```

Parameters

Name	Type	Required	Description
entityName	String	Yes	The logical name of the table.
formId	String	No	The form ID of the table.

Returns

Type: Promise

Description: Returns a promise containing the default main form descriptor with the following values.

Parameter Name	Type	Description
Attributes	Array of strings	List of all the columns on the main form.
EntityLogicalName	String	The logical name of the specified table.
Id	string	The form ID of the specified table.
Label	String	The label of the specified table.
Name	String	The display name of the specified table.
Sections	String	The sections name of the specified table.
ShowLabel	Boolean	Indicates whether to show the label of the specified table or not.

Parameter Name	Type	Description
Visible	Boolean	Indicates whether the form is visible or not.

Example

The following sample code shows how to get the main form descriptor for a specified table.

JavaScript

```
// Define the table and form ID
var entityName = "account";
var formId = "8448b78f-8f42-454e-8e2a-f8196b0419af";

// Get the main form descriptor
Xrm.Utility.getEntityMainFormDescriptor(entityName, formId);
```

Related topics

[Xrm.Utility](#)

getGlobalContext (Client API reference)

Article • 11/30/2022

Gets the global context.

The method provides access to the global context without going through the form context. It contains an equivalent of all the methods available for the `Xrm.Page.context` object (now deprecated) to retrieve information specific to the client, organization or user.

ⓘ Important

To access the global context information in a standalone HTML Web resource, you should include a reference to `ClientGlobalContext.js.aspx` in the web resource, and then use the `GetGlobalContext` function. More information: [GetGlobalContext function and ClientGlobalContext.js.aspx](#)

Properties of Global Context (getGlobalContext)

Use the following properties of global context to return information about the client, organization settings, or user settings:

Property	Description
<code>client</code>	Returns information about the client.
<code>organizationSettings</code>	Returns information about the current organization settings.
<code>userSettings</code>	Returns information about the current user settings.

Methods of Global Context (getGlobalContext)

Method	Description
<code>getAdvancedConfigSetting</code>	Returns information about the advanced configuration settings for the organization.
<code>getClientUrl</code>	Returns the base URL that was used to access the application.
<code>getCurrentAppName</code>	Returns the name of the current business app in model-driven apps.

Method	Description
getCurrentAppProperties	Returns the properties of the current business app in model-driven apps.
getCurrentAppUrl	Returns the URL of the current business app in model-driven apps.
getVersion	Returns the version number of the model-driven apps instance.
getWebResourceUrl	Returns the relative URL with the caching token for the specified web resource.
isOnPremises	Returns a boolean value indicating if the model-driven apps instance is hosted on-premises or online.
prependOrgName	Prefixes the current organization's unique name to a string, typically a URL path.

getGlobalContext.client (Client API reference)

Article • 08/04/2023

Provides access to the methods to determine which client is being used, whether the client is connected to the server, and what kind of device is being used.

```
var clientContext = Xrm.Utility.getGlobalContext().client
```

The following methods are available for the client context.

getClient

Returns a value to indicate which client the script is executing in.

Syntax

```
clientContext.getClient()
```

Return Value

Type: String

Description: The values returned are:

Value	Client
Web	Web application
Web	Unified Interface
Outlook	Dynamics 365 for Outlook client (COM add-in)
Mobile	Mobile app

getClientState

Returns a value to indicate the state of the client. A client in offline-first mode (in preview) always indicates it's offline.

Syntax

```
clientContext.getClientState()
```

Return Value

Type: String

Description: The values returned are:

Value	Client
Online	Web application, Dynamics 365 for Outlook client (COM add-in), Mobile app, Unified Interface
Offline	Outlook, Mobile app

getFormFactor

Returns information about the kind of device the user is using.

Syntax

```
clientContext.getFormFactor()
```

Return Value

Type: Number

Description: The values returned are:

Value	Form Factor
0	Unknown
1	Desktop
2	Tablet
3	Phone

isOffline

Returns information whether the client state is online or offline. A client in offline-first mode always reports it's offline.

Syntax

```
clientContext.isOffline()
```

Return Value

Type: Boolean

Description: **true** if the server is offline; **false** otherwise.

isNetworkAvailable

Returns information whether the network is available or not, regardless of client mode.

ⓘ Note

This API is not available for Dynamics 365 Customer Engagement on-premise deployments. More information: [Client APIs not supported in Customer Engagement \(on-premises\)](#)

Syntax

```
clientContext.isNetworkAvailable()
```

Return Value

Type: Boolean

Description: **true** if the network is available; **false** otherwise.

Related articles

[Organization Settings](#)

[User Settings](#)

[Xrm.Utility.getContext](#)

getGlobalContext.organizationSettings (Client API reference)

Article • 08/04/2023

Returns information about the current organization settings.

```
var organizationSettings = Xrm.Utility.getGlobalContext().organizationSettings
```

The `organizationSettings` object has the following properties.

attributes

Returns columns and their values as `key:value` pairs that are available for the organization table. More values are available as columns if they're specified as column dependencies in the web resource dependency list. The `key` is the column logical name.

Syntax

```
organizationSettings.attributes
```

Return Value

Type: Object

Description: An object with columns and their values.

baseCurrencyId

Returns the ID of the base currency for the current organization.

Deprecated; use [organizationSettings.baseCurrency](#) instead to access the display name along with the ID of the base currency.

Syntax

```
organizationSettings.baseCurrencyId
```

Return Value

Type: String

Description: ID of the base currency.

baseCurrency

Returns a lookup object containing the ID, name, and table type of the base currency for the current organization. This method is supported only on Unified Interface.

Syntax

```
organizationSettings.baseCurrency
```

Return Value

Type: Lookup Object

Description: Object containing the `id`, `name`, and `entityType` of the base currency. For example:

```
{id: "e7dd9bc6-d239-ea11-a813-000d3a35b14a", entityType: "transactioncurrency",  
name: "US Dollar"}
```

defaultCountryCode

Returns the default country/region code for phone numbers for the current organization.

Syntax

```
organizationSettings.defaultCountryCode
```

Return Value

Type: String

Description: Default country/region code for phone numbers.

isAutoSaveEnabled

Indicates whether the auto-save option is enabled for the current organization.

Syntax

```
organizationSettings.isAutoSaveEnabled
```

Return Value

Type: Boolean

Description: **true** if enabled; **false** otherwise.

languageId

Returns the preferred language ID for the current organization.

Syntax

```
organizationSettings.languageId
```

Return Value

Type: Number

Description: Preferred Language ID. For example:

```
1033
```

organizationId

Returns the ID of the current organization.

Syntax

```
organizationSettings.organizationId
```

Return Value

Type: String

Description: ID of the current organization.

isTrialOrganization

Returns a boolean indicating whether the organization is a trial organization.

ⓘ Note

This API is not available for Dynamics 365 Customer Engagement on-premise deployments. More information: [Client APIs not supported in Customer Engagement \(on-premises\)](#)

Syntax

```
organizationSettings.isTrialOrganization
```

Return Value

Type: Boolean

Description: **true** if the organization is a trial organization; **false** otherwise.

organizationExpiryDate

Returns the expiry date of the current organization if it's a trial organization.

ⓘ Note

This API is not available for Dynamics 365 Customer Engagement on-premise deployments. More information: [Client APIs not supported in Customer Engagement \(on-premises\)](#)

Syntax

```
organizationSettings.organizationExpiryDate
```

Return Value

Type: Date

Description: Returns a `Date` object with the organization's expiry date if it's a trial organization. Returns NULL if the organization isn't a trial organization.

uniqueName

Returns the unique name of the current organization.

Syntax

```
organizationSettings.uniqueName
```

Return Value

Type: String

Description: Unique name of the current organization.

useSkypeProtocol

Indicates whether the Skype protocol is used for the current organization.

Syntax

```
organizationSettings.useSkypeProtocol
```

Return Value

Type: Boolean

Description: `true` if Skype protocol is used; `false` otherwise.

fullNameConventionCode

Returns the FullNameConventionCode setting of the current organization.

Syntax

```
organizationSettings.fullNameConventionCode
```

Return Value

Type: Number

Description: Returns a number denoting the full name format selected in the system settings. The following are the possible values and the corresponding format:

- 0: LastName, FirstName
- 1: FirstName LastName
- 2: LastName, FirstName MiddleInitial
- 3: FirstName MiddleInitial LastName
- 4: LastName, FirstName MiddleName
- 5: FirstName MiddleName LastName
- 6: LastName FirstName
- 7: LastNameFirstName

Related articles

[Client context](#)

[User settings](#)

[Xrm.Utility.getContext](#)

getGlobalContext.userSettings (Client API reference)

Article • 08/04/2023

Returns information about the current user settings.

```
var userSettings = Xrm.Utility.getGlobalContext().userSettings
```

The `userSettings` object provides following properties and a method.

dateFormattingInfo

Returns the date formatting information for the current user.

Syntax

```
userSettings.dateFormattingInfo
```

Return Value

Type: Object

Description: An object with information about date formatting such as `FirstDayOfWeek`, `LongDatePattern`, `MonthDayPattern`, `TimeSeparator`, and so on.

defaultDashboardId

Returns the ID of the default dashboard for the current user.

Syntax

```
userSettings.defaultDashboardId
```

Return Value

Type: String

Description: ID of the default dashboard.

isGuidedHelpEnabled

Indicates whether guided help is enabled for the current user.

Syntax

```
userSettings.isGuidedHelpEnabled
```

Return Value

Type: Boolean

Description: true if enabled; false otherwise.

isHighContrastEnabled

Indicates whether high contrast is enabled for the current user.

Syntax

```
userSettings.isHighContrastEnabled
```

Return Value

Type: Boolean

Description: true if enabled; false otherwise.

isRTL

Indicates whether the language for the current user is a right-to-left (RTL) language.

Syntax

```
userSettings.isRTL
```

Return Value

Type: Boolean

Description: true if it is RTL; false otherwise.

languageId

Returns the language ID for the current user.

Syntax

```
userSettings.languageId
```

Return Value

Type: Number

Description: Language ID.

roles

Returns a collection of lookup objects containing the GUID and display name of each of the security role assigned to the user and any security roles assigned to the team that the user is associated with. This method is supported only on Unified Interface.

Syntax

```
userSettings.roles
```

Return Value

Type: Collection

Description: Object containing `id` and `name` of each of the security role or teams that the user is associated with.

securityRolePrivileges

Returns an array of strings that represent the GUID values of each of the security role privilege that the user is associated with or any teams that the user is associated with.

ⓘ Note

This API is not available for Dynamics 365 Customer Engagement on-premise deployments. More information: [Client APIs not supported in Customer Engagement \(on-premises\)](#)

Syntax

```
userSettings.securityRolePrivileges
```

Return Value

Type: Array

Description: GUID values of each of the security role privilege.

getSecurityRolePrivilegesInfo()

Returns a promise which resolves with an object whose keys are the security role privilege GUIDs and values are objects containing the `businessUnitId`, `depth`, and `privilegeName` of the security role privilege.

Syntax

```
userSettings.getSecurityRolePrivilegesInfo().then(successCallback, errorCallback);
```

Parameters

Name	Type	Required	Description
successCallback	Function	No	A function to call when the security role privileges information is retrieved. A dictionary will be passed to the success callback where the security role privilege GUIDs will be the keys and the values will be objects containing the following properties: <ul style="list-style-type: none">• id: String. The security role privilege GUID.• businessUnitId: String. The GUID of the business unit of the security role privilege.• privilegeName: String. The security role privilege name.• depth: String. The security role privilege depth.

errorCallback	Function	No	A function to call when the operation fails. An object with the following properties will be passed: <ul style="list-style-type: none"> • errorCode: Number. The error code. • message: String. An error message describing the issue.
---------------	----------	----	--

Return Value

Type: `Promise<{[key: string]: {id: string, businessUnitId: string, privilegeName: string, depth: number}}>`

On success, returns a promise object containing the values specified in the description of the **successCallback** parameter above.

Description: GUID and additional details like Business Unit and Privilege Name of each of the security role privileges.

Example

JavaScript

```
userSettings
  .getSecurityRolePrivilegesInfo()
  .then(function success(rolePrivileges) {
    var privilegeGuids = Object.keys(rolePrivileges);
    console.log("Privileges Count: " + privilegeGuids.length);

    // Print information about the first role privilege in the dictionary
    var guid = privilegeGuids[0];
    console.log("Privilege Id: " + rolePrivileges[guid].id);
    console.log("Privilege Name: " + rolePrivileges[guid].privilegeName);
    console.log("Privilege Business Unit Id: " +
    rolePrivileges[guid].businessUnitId);
    console.log("Privilege depth: " + rolePrivileges[guid].depth);
  });
}
```

securityRoles

Returns an array of strings that represent the GUID values of each of the security role or teams that the user is associated with.

Deprecated; use [userSettings.roles](#) instead to view the display names of security roles or teams along with the ID.

Syntax

```
userSettings.securityRoles
```

Return Value

Type: Array

Description: GUID values of each of the security role. For example:

```
["0d3dd20a-17a6-e711-a94e-000d3a1a7a9b", "ff42d20a-17a6-e711-a94e-000d3a1a7a9b"]
```

transactionCurrency

Returns a lookup object containing the ID, display name, and table type of the transaction currency for the current user. This method is supported only on Unified Interface.

Syntax

```
userSettings.transactionCurrency
```

Return Value

Type: Lookup object

Description: Object containing the `id`, `name`, and `entityType` of the transaction currency. For example:

```
{id: "e7dd9bc6-d239-ea11-a813-000d3a35b14a", entityType: "transactioncurrency",
name: "US Dollar"}
```

transactionCurrencyId

Returns the transaction currency ID for the current user.

Deprecated; use [userSettings.transactionCurrency](#) instead to access the display name along with the ID.

Syntax

```
userSettings.transactionCurrencyId
```

Return Value

Type: String

Description: Transaction currency ID.

userId

Returns the GUID of the **SystemUser.Id** value for the current user.

Syntax

```
userSettings.userId
```

Return Value

Type: String

Description: The ID of the user. For example:

```
"{75B5BA27-FD41-4D45-8E3A-C8446C95F0CC}"
```

userName

Returns the name of the current user.

Syntax

```
userSettings.userName
```

Return Value

Type: String

Description: Name of the current user.

getTimeZoneOffsetMinutes method

Returns the difference in minutes between the local time and Coordinated Universal Time (UTC).

Syntax

```
userSettings.getTimeZoneOffsetMinutes()
```

Return Value

Type: number

Description: Time zone offset in minutes.

Related topics

[Client context](#)

[Organization settings](#)

[Xrm.Utility.getContext](#)

getAdvancedConfigSetting (Client API reference)

Article • 11/30/2022

Returns information about the advanced configuration settings for the organization.

Syntax

JavaScript

```
var globalContext = Xrm.Utility.getGlobalContext();
globalContext.getAdvancedConfigSetting(setting);
```

Parameters

Name	Type	Required	Description
setting	String	Yes	Name of the configuration setting. Only the following two configuration settings are supported: "MaxChildIncidentNumber" and "MaxIncidentMergeNumber"

Return Value

Returns the advanced configuration setting value.

Related topics

[Xrm.Utility.getGlobalContext](#)

getClientUrl (Client API reference)

Article • 11/30/2022

Returns the base URL that was used to access the application.

Syntax

JavaScript

```
var globalContext = Xrm.Utility.getGlobalContext();
globalContext.getClientUrl();
```

Return Value

Type: String

Description: The values returned will resemble those listed in the following table.

Value	Client
https://[org].crm.dynamics.com	model-driven apps (online)
http(s)://[server]/[org]	model-driven apps (on-premises)
https://localhost:2525	model-driven apps for Outlook with Offline Access when offline

Related topics

[Xrm.Utility.getGlobalContext](#)

[RetrieveCurrentOrganization function](#)

[RetrieveCurrentOrganizationRequest](#)

getCurrentAppName (Client API reference)

Article • 11/30/2022

Returns the name of the current business app in model-driven apps.

Syntax

JavaScript

```
var globalContext = Xrm.Utility.getGlobalContext();
globalContext.getCurrentAppName().then(successCallback, errorCallback);
```

Parameters

Name	Type	Required	Description
successCallback	Function	Yes	A function to call when the business app name is returned.
errorCallback	Function	Yes	A function to call when the operation fails.

Return Value

If this method is called in the context of a business app, returns the name of the business app. Otherwise, it fails with an error.

Related topics

[Create, manage, and publish model-driven apps using code](#)

[Xrm.Utility.getGlobalContext](#)

getCurrentAppProperties (Client API reference)

Article • 11/30/2022

Returns the properties of the current business app in model-driven apps.

Syntax

JavaScript

```
var globalContext = Xrm.Utility.getGlobalContext();
globalContext.getCurrentAppProperties().then(successCallback,
errorCallback);
```

Parameters

Name	Type	Required	Description
successCallback	Function	Yes	A function to call when the business app property information is returned. An object with the following attributes (app properties) is passed to the function : - appId - displayName - uniqueName - url - webResourceId - webResourceName - welcomePageId - welcomePageName
errorCallback	Function	Yes	A function to call when the operation fails.

Return Value

If this method is called in the context of a business app, returns the properties of the business app. Otherwise, it fails with an error.

Related topics

[Create, manage, and publish model-driven apps using code](#)

Xrm.Utility.getGlobalContext

getCurrentAppUrl (Client API reference)

Article • 11/30/2022

Returns the URL of the current business app in model-driven apps.

ⓘ Note

In mobile client, this method returns null value.

Syntax

JavaScript

```
var globalContext = Xrm.Utility.getGlobalContext();
globalContext.getCurrentAppUrl();
```

Return Value

Type: String

Description: URL of the current business app. Possible return values:

Value	Client
https://[org].crm.dynamics.com/main.aspx?appid=[GUID]	model-driven apps (online)
https://[server]/[org]/main.aspx?appid=[GUID]	model-driven apps (on-premises)

Related topics

[Create, manage, and publish model-driven apps using code](#)

[Xrm.Utility.getGlobalContext](#)

getVersion (Client API reference)

Article • 11/30/2022

Returns the version number of the model-driven apps instance.

Syntax

JavaScript

```
var globalContext = Xrm.Utility.getGlobalContext();
globalContext.getVersion();
```

Return Value

Type: String

Description: Version of the model-driven apps instance. For example:

"9.0.0.1103"

Related topics

[Xrm.Utility.getGlobalContext](#)

getWebResourceUrl (Client API reference)

Article • 12/16/2022

Returns the relative URL with the caching token for the specified web resource.

Syntax

JavaScript

```
var globalContext = Xrm.Utility.getGlobalContext();
globalContext.getWebResourceUrl(webResourceName);
```

Parameters

Name	Type	Required	Description
webResourceName	String	Yes	Name of the web resource.

Return Value

Type: String

Description: The relative URL, including the caching token, for the specified web resource.

Note

If you:

- Use this method every time, you will have the latest version of the web resource and it will be cached for up to one year.
- Use this method once and save the URL, you will get the version, which was current at the time the URL was built, for next one year.
- Don't use this method and construct the URL yourself, the item returned won't be cached.

Example

JavaScript

```
var globalContext = Xrm.Utility.getGlobalContext();
globalContext.getWebResourceUrl("sample_webResource1.js");
```

This will return the web resource URL with the caching token:

```
/%7b637199221580014143%7d/webresources/sample_webResource1.js
```

Related topics

[Xrm.Utility.getGlobalContext](#)

isOnPremises (Client API reference)

Article • 11/30/2022

Returns a boolean value indicating if the model-driven apps instance is hosted on-premises or online.

Syntax

JavaScript

```
var globalContext = Xrm.Utility.getGlobalContext();
globalContext.isOnPremises();
```

Return Value

Type: Boolean

Description: **true** if the model-driven apps instance is on-premises; **false** otherwise.

Related topics

[Xrm.Utility.getGlobalContext](#)

prependOrgName (Client API reference)

Article • 11/30/2022

Prefixes the current organization's unique name to a string, typically a URL path.

Syntax

JavaScript

```
var globalContext = Xrm.Utility.getGlobalContext();
globalContext.prependOrgName(sPath);
```

Parameters

Name	Type	Required	Description
sPath	String	Yes	A local path to a resource.

Return Value

Type: String

Description: A path string with the organization name prefixed in the following format:

```
"/"+ orgName + sPath
```

Related topics

[Xrm.Utility.getGlobalContext](#)

getLearningPathAttributeName (Client API reference)

Article • 11/30/2022

Returns the name of the DOM attribute expected by the Learning Path (guided help) Content Designer for identifying UI controls in the model-driven apps forms.

A column by this name must be added to the UI element that needs to be exposed to Learning Path (guided help).

Syntax

```
Xrm.Utility.getLearningPathAttributeName()
```

Returns

Type: String

Description: DOM column expected by the Learning Path (guided help) Content Designer.

Related topics

[Xrm.Utility](#)

getPageContext (Client API reference)

Article • 11/30/2022

Gets the page context as an object representing the page.

ⓘ Note

This method is supported only on Unified Interface.

Syntax

```
Xrm.Utility.getPageContext();
```

Returns

The method returns an object with the `input` property. The `input` property is an object with the following values depending on whether you are currently on the *entity form* or *entity list*:

Entity form

Name	Type	Description
pageType	String	The current page type. The value returned is "entityrecord".
entityName	String	Logical name of the table currently displayed.
entityId	String	ID of the table record currently displayed in the form.
createFromEntity	Lookup	The parent record that provides default values based on mapped column values. The lookup object has the following String properties: <code>entityType</code> , <code>id</code> , and <code>name</code> .
formId	String	ID of the currently displayed form.

Entity list

Name	Type	Description
pageType	String	The current page type. The value returned is "entitylist".

Name	Type	Description
entityName	String	Logical name of the table currently displayed.
viewId	String	ID of the view currently displayed.
viewType	String	Type of the view currently displayed. Possible values are "savedquery" or "userquery".

 **Note**

Only the `pageType` and `entityName` parameters will return values every time; all other parameters will return values only if specified by the logic that opened the page.

Related topics

[Xrm.Utility](#)

getResourceString (Client API reference)

Article • 11/30/2022

Returns the localized string for a given key associated with the specified web resource.

Syntax

```
Xrm.Utility.getResourceString(webResourceName, key)
```

Parameters

Name	Type	Required	Description
webResourceName	String	Yes	The name of the web resource.
key	String	Yes	The key for the localized string.

Return value

A localized string.

Remarks

When you create RESX web resources you must explicitly set the language value and include the locale identifier (LCID) for the appropriate language in the name of the web resource. For example, `new_/strings/MyAppResources.1033.resx` would contain resources for English language. See [Microsoft locale ID values](#) for a list of LCID values.

For example `Xrm.Utility.getResourceString("new_/strings/MyAppResources", "hello")` will return the localized string value for the resource key `hello` within the `new_/strings/MyAppResources.1033.resx` web resource if the user's preferred language is English. Notice that the function doesn't refer to any specific language or full name of any RESX web resource. This functionality depends on the RESX web resource being associated to the calling JavaScript web resource as a dependency. More information: [Web resource dependencies](#).

The appropriate string value will be determined by the individual user's language preference and the languages available in the organization. If a localized string is not found that matches the user's language preference, the localized string will

automatically fallback to the base language for the organization. If no matching localized string is found for the organizations base language, a null value will be returned. If no matching RESX web resource is found for user's LCID, an exception `{webResourceName} does not exist.` will be thrown.

Related topics

[Xrm.Utility](#)

[String \(RESX\) web resources](#)

[Web resource dependencies](#)

invokeProcessAction (Client API reference)

Article • 11/30/2022

Invokes an action based on the specified parameters.

For more information about actions, see [Use actions](#)

Syntax

```
Xrm.Utility.invokeProcessAction(name,parameters).then(successCallback,  
errorCallback)
```

Parameters

Name	Type	Required	Description
name	String	Yes	Name of the process action to invoke.
parameters	object	No	An object containing input parameters for the action. You define an object using <code>key:value</code> pairs of items, where <code>key</code> is of String type. To specify a target, add a pair with <code>Target</code> as the key and an object with key values <code>entityName</code> and <code>id</code> as the value.
successCallback	Function	Yes	A function to call when the action is invoked.
errorCallback	Function	Yes	A function to call when the operation fails.

Returns

On success, returns an object with the Web API result along with any action output. On failure, returns an object with error details.

Related topics

[Use actions](#)

[Xrm.Utility](#)

lookupObjects (Client API reference)

Article • 05/14/2023

Opens a lookup control to select one or more items.

Syntax

```
Xrm.Utility.lookupObjects(lookupOptions).then(successCallback, errorCallback)
```

Parameters

lookupOptions: Object. Defines the options for opening the lookup dialog. Has the following properties:

Property Name	Type	Required	Description
allowMultiSelect	Boolean	No	Indicates whether the lookup allows more than one item to be selected.
defaultEntityType	String	No	The default table type to use.
defaultViewId	String	No	The default view to use.
disableMru	Boolean	No	Decides whether to display the most recently used(MRU) item. Available only for Unified Interface.
entityTypes	Array	Yes	The table types to display.
filters	Array of objects	No	Used to filter the results. Each object in the array contains the following values: <ul style="list-style-type: none">• filterXml: String. The FetchXML filter element to apply.• entityLogicalName: String. The table type to which to apply this filter.
searchText	String	No	Indicates the default search term for the lookup control. This is supported only on Unified Interface .
viewIds	Array	No	The views to be available in the view picker. Only system views are supported.

successCallback: Function. A function to call when the lookup control is invoked. An array of objects with the following properties is passed:

- **entityType**: String. table type of the record selected in the lookup control.
- **id**: String. ID of the record selected in the lookup control.
- **name**: String. Name of the record selected in the lookup control.

errorCallback: Function. A function to call when the operation fails. It is not considered a failure if the user cancels the operation.

Example

JavaScript

```
//define data for lookupOptions
var lookupOptions =
{
    defaultEntityType: "account",
    entityTypes: ["account"],
    allowMultiSelect: false,
    defaultViewId:"0D5D377B-5E7C-47B5-BAB1-A5CB8B4AC10",
    viewIds:[ "0D5D377B-5E7C-47B5-BAB1-A5CB8B4AC10", "00000000-0000-0000-00AA-
000010001003"],
    searchText:"Allison",
    filters: [{filterXml: "<filter type='or'><condition attribute='name'
operator='like' value='A%' /></filter>",entityLogicalName: "account"}]
};

// Get account records based on the lookup Options
Xrm.Utility.lookupObjects(lookupOptions).then(
    function(success){
        console.log(success);
    },
    function(error){console.log(error);});
```

Related topics

[Xrm.Utility](#)

refreshParentGrid (Client API reference)

Article • 12/16/2022

Refreshes the parent grid containing the specified record.

Syntax

```
Xrm.Utility.refreshParentGrid(lookupOptions)
```

Parameters

lookupOptions: An object with the following properties to specify the record:

Property Name	Type	Required	Description
entityType	String	Yes	Table type of the record.
id	String	Yes	ID of the record.
name	String	No	Name of the record.

Related topics

[Xrm.Utility](#)

showProgressIndicator (Client API reference)

Article • 11/30/2022

Displays a progress dialog with the specified message.

Any subsequent call to this method will update the displayed message in the existing progress dialog with the message specified in the latest method call.

ⓘ Note

Although the showProgressIndicator function is synchronous, the UI updates asynchronously. This means that any code synchronously executed immediately after calling showProgressIndicator may execute before the progress indicator is shown in the UI.

⚠ Warning

The progress dialog blocks the UI until it is closed using the [closeProgressIndicator](#) method. So, you must use this method with caution.

Syntax

```
Xrm.Utility.showProgressIndicator(message)
```

Parameters

Name	Type	Required	Description
message	String	Yes	The message to be displayed in the progress dialog.

Related topics

[closeProgressIndicator](#)

[Xrm.Utility](#)

Xrm.WebApi (Client API reference)

Article • 11/30/2022

Provides properties and methods to use Web API to create and manage records and execute Web API actions and functions in model-driven apps.

Properties

Property	Description
online	Provides methods to use Web API to create and manage records and execute Web API actions and functions in model-driven apps when connected to the model-driven apps server (online mode).
offline	Provides methods to create and manage records in model-driven apps in mobile clients while working in the <i>offline</i> mode.

Methods

With mobile offline configured, the source for these records will depend on the current client state. In offline mode, the source is the offline data store. In online mode, the source is the server. If the client is offline-first (in preview), the methods in [online](#) can be used to access tables and records that are not part of the offline profile, as long as the client has network connectivity.

Method	Description
createRecord	Creates a table record.
deleteRecord	Deletes a table record.
retrieveRecord	Retrieves a table record.
retrieveMultipleRecords	Retrieves a collection of table records.
updateRecord	Updates a table record.
isAvailableOffline	Returns a boolean value indicating whether an entity is present in user's profile and is currently available for use in offline mode.
execute	Execute a single action, function, or CRUD operation.
executeMultiple	Execute a collection of action, function, or CRUD operations.

Related topics

[Use the Microsoft Dataverse Web API](#)
[Client API Xrm object](#)

Xrm.WebApi.online (Client API reference)

Article • 11/30/2022

Provides methods to use Web API to create and manage records and execute Web API actions and functions in model-driven apps when connected to the model-driven apps server (online mode).

```
var onlineWebApi = Xrm.WebApi.online;
```

The **onlineWebApi** object provides the following methods:

- [createRecord](#)
- [deleteRecord](#)
- [retrieveRecord](#)
- [retrieveMultipleRecords](#)
- [updateRecord](#)
- [execute](#)
- [executeMultiple](#)

Related topics

[Xrm.WebApi.offline](#)

[Xrm.WebApi](#)

Xrm.WebApi.offline (Client API reference)

Article • 11/30/2022

Provides methods to create and manage records in model-driven apps in mobile clients while working in the *offline* mode.

For information about the mobile offline feature, see [Configure mobile offline synchronization to allow users to work in offline mode on their mobile device](#)

```
var offlineWebApi = Xrm.WebApi.offline;
```

ⓘ Note

Use **Xrm.WebApi.offline** instead of the **deprecated Xrm.Mobile.Offline** namespace to create and manage records in the mobile clients while working in the offline mode.

The **offlineWebApi** object provides the following methods. When in the offline mode, these methods will work only for tables that are enabled for mobile offline synchronization and available in current user's mobile offline profile.

- [createRecord](#)
- [deleteRecord](#)
- [isAvailableOffline](#)
- [retrieveRecord](#)
- [retrieveMultipleRecords](#)
- [updateRecord](#)

ⓘ Important

While creating or updating record in the offline mode, only basic validation is performed on the input data. Basic validation includes things such as ensuring that the table column name specified is in lower case and does exist for a table, checking for data type mismatch for the specified column value, preventing records getting created with the same GUID value, checking whether the related table is offline enabled when retrieving related table records, and validating if the record that you want to retrieve, update, or delete actually exists in the offline data store. Business-level validations happen only when you are connected to the server and

the data is synchronized. A record is created or updated only if the input data is completely valid.

Related topics

[Xrm.WebApi.online](#)

[Xrm.WebApi](#)

createRecord (Client API reference)

Article • 08/22/2022

Creates a table record.

Syntax

```
Xrm.WebApi.createRecord(entityLogicalName, data).then(successCallback,  
errorCallback);
```

Parameters

Name	Type	Required	Description
entityLogicalName	String	Yes	Logical name of the table you want to create. For example: "account".
data	Object	Yes	A JSON object defining the columns and values for the new table record. See examples later in this topic to see how you can define the <code>data</code> object for various create scenarios.
successCallback	Function	No	A function to call when a record is created. An object with the following properties will be passed to identify the new record: <ul style="list-style-type: none">• entityType: String. The table logical name of the new record.• id: String. GUID of the new record.
errorCallback	Function	No	A function to call when the operation fails. An object with the following properties will be passed: <ul style="list-style-type: none">• errorCode: Number. The error code.• message: String. An error message describing the issue.

Return Value

On success, returns a promise object containing the values specified earlier in the description of the `successCallback` parameter.

Examples

These examples use the same request objects as demonstrated in [Create a table row using the Web API](#) to define the data object for creating a table record.

Basic create

Creates a sample account record.

JavaScript

```
// define the data to create new account
var data =
{
    "name": "Sample Account",
    "creditonhold": false,
    "address1_latitude": 47.639583,
    "description": "This is the description of the sample account",
    "revenue": 5000000,
    "accountcategorycode": 1
}

// create account record
Xrm.WebApi.createRecord("account", data).then(
    function success(result) {
        console.log("Account created with ID: " + result.id);
        // perform operations on record creation
    },
    function (error) {
        console.log(error.message);
        // handle error conditions
    }
);
```

Create related table records along with the primary record

You can create tables related to each other by defining them as navigation properties values. This is known as *deep insert*. In this example, we will create a sample account record along with the primary contact record and an associated opportunity record.

 **Note**

Creating related table records in a single create operation is not supported for offline mode.

JavaScript

```
// define data to create primary and related table records
var data =
{
    "name": "Sample Account",
    "primarycontactid":
    {
        "firstname": "John",
        "lastname": "Smith"
    },
    "opportunity_customer_accounts":
    [
        {
            "name": "Opportunity associated to Sample Account",
            "Opportunity_Tasks":
            [
                { "subject": "Task associated to opportunity" }
            ]
        }
    ]
}

// create account record
Xrm.WebApi.createRecord("account", data).then(
    function success(result) {
        console.log("Account created with ID: " + result.id);
        // perform operations on record creation
    },
    function (error) {
        console.log(error.message);
        // handle error conditions
    }
);

```

Associate tables on creating new records

To associate new table records to existing table records, set the value of single-valued navigation properties using the `@odata.bind` annotation.

Note

The names of single-valued navigation properties are not always the same as the `LogicalName` for the lookup attribute. You should make sure you are using the `Name` attribute value of the `NavigationProperty` element in the Web API \$metadata service document. More information: [Web API Navigation Properties](#)

Here is code example:

The following example creates an account record, and associates it to an existing contact record to set the latter as the primary contact for the new account record:

```
JavaScript

var data =
{
    "name": "Sample Account",
    "primarycontactid@odata.bind": "/contacts(465b158c-541c-e511-80d3-3863bb347ba8)"
}

// create account record
Xrm.WebApi.createRecord("account", data).then(
    function success(result) {
        console.log("Account created with ID: " + result.id);
        // perform operations on record creation
    },
    function (error) {
        console.log(error.message);
        // handle error conditions
    }
);
```

Deprecated method for mobile offline scenario

ⓘ Note

Instead of using `@odata.bind` annotation example above, the deprecated `lookup` object with case-sensitive properties (`logicalname` and `id`) is still supported for existing customizations. However, it is recommended to use `@odata.bind` annotation for both online and offline scenario instead of using this deprecated object.

The following example uses the deprecated method to create an account record, and associate it to an existing contact record to set the latter as the primary contact for the new account record from mobile clients when working in the offline mode:

```
JavaScript

var data =
{
    "name": "Sample Account",
    "primarycontactid": {
        "logicalname": "contact",
        "id": "465b158c-541c-e511-80d3-3863bb347ba8"
```

```
        }

// create account record
Xrm.WebApi.offline.createRecord("account", data).then(
    function success(result) {
        console.log("Account created with ID: " + result.id);
        // perform operations on record creation
    },
    function (error) {
        console.log(error.message);
        // handle error conditions
    }
);
```

Related topics

[Create a table row using the Web API](#)

[Xrm.WebApi](#)

deleteRecord (Client API reference)

Article • 11/30/2022

Deletes a table record.

Syntax

```
Xrm.WebApi.deleteRecord(entityLogicalName, id).then(successCallback,  
errorCallback);
```

Parameters

Name	Type	Required	Description
entityLogicalName	String	Yes	The table logical name of the record you want to delete. For example: "account".
id	String	Yes	GUID of the table record you want to delete.
successCallback	Function	No	A function to call when a record is deleted. An object with the following properties will be passed to identify the deleted record: <ul style="list-style-type: none">• entityType: String. The table type of the record.• id: String. GUID of the record.• name: String. Name of the record.
errorCallback	Function	No	A function to call when the operation fails.

Return Value

On success, returns a promise object containing the values specified earlier in the description of the **successCallback** parameter.

Examples

These examples use some of the same request objects as demonstrated in [Update and delete tables using the Web API](#) to define the data object for updating an entity record.

Deletes an account with record ID = 5531d753-95af-e711-a94e-000d3a11e605.

JavaScript

```
Xrm.WebApi.deleteRecord("account", "5531d753-95af-e711-a94e-  
000d3a11e605").then(  
    function success(result) {  
        console.log("Account deleted");  
        // perform operations on record deletion  
    },  
    function (error) {  
        console.log(error.message);  
        // handle error conditions  
    }  
)
```

Related topics

[Xrm.WebApi](#)

retrieveRecord (Client API reference)

Article • 11/30/2022

Retrieves a table record.

Syntax

```
Xrm.WebApi.retrieveRecord(entityLogicalName, id, options).then(successCallback,  
errorCallback);
```

Parameters

Name	Type	Required	Description
entityLogicalName	String	Yes	The table logical name of the record you want to retrieve. For example: "account".
id	String	Yes	GUID of the table record you want to retrieve.
options	String	No	<p>OData system query options, \$select and \$expand, to retrieve your data.</p> <ul style="list-style-type: none">• Use the \$select system query option to limit the properties returned by including a comma-separated list of property names. This is an important performance best practice. If properties aren't specified using \$select, all properties will be returned.• Use the \$expand system query option to control what data from related tables is returned. If you just include the name of the navigation property, you'll receive all the properties for related records. You can limit the properties returned for related records using the \$select system query option in parentheses after the navigation property name. Use this for both <i>single-valued</i> and <i>collection-valued</i> navigation properties. Note that for offline we only support nested \$select option inside the \$expand. <p>You specify the query options starting with <code>?.</code>. You can also specify multiple query options by using <code>&</code> to separate the query options. For example:</p> <pre>? \$select=name&\$expand=primarycontactid(\$select=contactid,fullname)</pre>
successCallback	Function	No	A function to call when a record is retrieved. A JSON object with the retrieved properties and values will be passed to the function.

errorCallback	Function	No	A function to call when the operation fails.
---------------	----------	----	--

Return Value

On success, returns a promise containing a JSON object with the retrieved columns and their values. If the requested record does not exist, returns an error.

Examples

Basic retrieve

Retrieves the name and revenue of an account record with record ID = 5531d753-95af-e711-a94e-000d3a11e605.

JavaScript

```
Xrm.WebApi.retrieveRecord("account", "a8a19cdd-88df-e311-b8e5-6c3be5a8b200", "?$select=name,revenue").then(
    function success(result) {
        console.log("Retrieved values: Name: " + result.name + ", Revenue: " +
result.revenue);
        // perform operations on record retrieval
    },
    function (error) {
        console.log(error.message);
        // handle error conditions
    }
);
```

The above example displays the following in your console; you might see other values depending on your data:

Retrieved values: Name: Sample Account, Revenue: 5000000

Retrieve related tables for a table instance by expanding single-valued navigation properties

The following example demonstrates how to retrieve the contact for an account record with record ID = a8a19cdd-88df-e311-b8e5-6c3be5a8b200. For the related contact record, we are only retrieving the **contactid** and **fullname** properties.

JavaScript

```
Xrm.WebApi.retrieveRecord("account", "a8a19cdd-88df-e311-b8e5-6c3be5a8b200", "?$select=name&$expand=primarycontactid($select=contactid,fullname)").then(
    function success(result) {
        console.log("Retrieved values: Name: " + result.name + ", Primary Contact
```

```
ID: " + result.primarycontactid.contactid +
      ", Primary Contact Name: " + result.primarycontactid.fullname);
    // perform operations on record retrieval
  },
  function (error) {
    console.log(error.message);
    // handle error conditions
  }
);
```

The above example displays the following in your console; you might see other values depending on your data:

```
Retrieved values: Name: Adventure Works, Primary Contact ID: 49a0e5b9-88df-e311-b8e5-
6c3be5a8b200, Primary Contact Name: Adrian Dumitrascu
```

Related topics

[Xrm.WebApi.retrieveMultipleRecords](#)

[Xrm.WebApi](#)

retrieveMultipleRecords (Client API reference)

Article • 01/12/2023

Retrieves a collection of table records.

Syntax

```
Xrm.WebApi.retrieveMultipleRecords(entityLogicalName, options,  
maxPageSize).then(successCallback, errorCallback);
```

Parameters

Name	Type	Required	Description
entityLogicalName	String	Yes	The table logical name of the records you want to retrieve. For example: "account".
options	String	No	<p>OData system query options or FetchXML query to retrieve your data.</p> <ul style="list-style-type: none">• Following system query options are supported: \$select, \$top, \$filter, \$expand, and \$orderby.• Use the \$expand system query option to control what data from related tables is returned. If you just include the name of the navigation property, you'll receive all the properties for related records. You can limit the properties returned for related records using the \$select system query option in parentheses after the navigation property name. Use this for both <i>single-valued</i> and <i>collection-valued</i> navigation properties. Note that for offline we only support nested \$select option inside the \$expand.• To specify a FetchXML query, use the `fetchXml` column to specify the query. <p>NOTE: You must always use the \$select system query option to limit the properties returned for a table record by including a comma-separated list of property names. This is an important performance best practice. If properties aren't specified using \$select, all properties will be returned.</p>

		<p>You specify the query options starting with `?`. You can also specify multiple system query options by using `&` to separate the query options.</p>
		<p>When you specify an OData query string for the `options` parameter, the query should be encoded for special characters.</p>
		<p>When you specify a FetchXML query for the `options` parameter, the query should not be encoded.</p>
		<p>See examples later in this article to see how you can define the `options` parameter for various retrieve multiple scenarios.</p>
maxPageSize	Number	<p>No</p> <p>Specify a positive number that indicates the number of table records to be returned per page. If you don't specify this parameter, the value is defaulted to the maximum limit of 5000 records.</p> <p>If the number of records being retrieved is more than the specified `maxPageSize` value or 5000 records, `nextLink` column in the returned promise object will contain a link to retrieve records.</p>
successCallback	Function	<p>No</p> <p>A function to call when table records are retrieved. An object with the following values is passed to the function:</p> <ul style="list-style-type: none"> • entities: An array of JSON objects, where each object represents the retrieved table record containing columns and their values as `key: value` pairs. The ID of the table record is retrieved by default. • nextLink: (optional) String. If the number of records being retrieved is more than the value specified in the `maxPageSize` parameter in the request, this returns the URL to return the next page of records. • fetchXmlPagingCookie: (optional) String. For a fetchXml-based retrieveMultipleRecords operation with paging where the total record count is greater than the paging value, this attribute returns the paging cookie that can be used for a subsequent fetchXml operation to retrieve the next page of records.
errorCallback	Function	<p>No</p> <p>A function to call when the operation fails.</p>

Return Value

For a successful OData query `retrieveMultipleRecords` operation, returns a promise that contains an array of JSON objects (**entities**) containing the retrieved table records and the **nextLink** attribute (optional) with the URL pointing to next page of records in case paging (`maxPageSize`) is specified in the request, and the record count returned exceeds the paging value.

For successful FetchXML-based `retrieveMultipleRecords` operations, the promise response will contain a **fetchXmlPagingCookie** (optional) attribute when the operation returns more records than the paging value. This attribute will contain the paging cookie string that can be included in a subsequent `fetchXml` request to fetch the next page of records.

Unsupported Attribute Types for OData query options in Mobile Offline

The following [Column types](#) aren't supported when doing a `Xrm.WebApi.retrieveMultipleRecords` operation with OData query string options (for example, `$select` and `$filter`) in mobile offline mode. You should use FetchXML if the attribute type you need to work with is in this list of unsupported attribute types.

- MultiSelectPicklist
- File
- Image
- ManagedProperty
- CalendarRules
- PartyList
- Virtual

Unsupported features in Mobile Offline

The following features aren't supported in Mobile Offline:

- Grouping and Aggregation features

Supported Filter Operations Per Attribute Type in Mobile Offline using FetchXML

The following operations are supported for all attribute types when working with FetchXML:

- Equals (`eq`)
- Not Equals (`neq`)
- Null (`null`)
- Not Null (`not-null`)

The following table lists more operations supported for each attribute type:

Attribute Type	Supported Operations
BigInt, Decimal, Double, Integer	Greater Than (<code>gt</code>) Greater Than or Equals (<code>gte</code>) Less Than (<code>lt</code>) Less Than or Equals (<code>lte</code>)
Boolean, Customer	In (<code>in</code>) Not In (<code>not-in</code>)
EntityName, Picklist, State, Status	Like (<code>like</code>) Not Like (<code>not-like</code>) Begins With (<code>begins-with</code>) Not Begin With (<code>not-begin-with</code>) Ends With (<code>ends-with</code>) Not End With (<code>not-end-with</code>) In (<code>in</code>) Not In (<code>not-in</code>)
Guid, Lookup	In (<code>in</code>) Not In (<code>not-in</code>) Equals User ID (<code>eq-userid</code>) Not Equals User ID (<code>ne-userid</code>)
Money	Greater Than (<code>gt</code>) Greater Than or Equals (<code>gte</code>) Less Than (<code>lt</code>) Less Than or Equals (<code>lte</code>) In (<code>in</code>) Not In (<code>not-in</code>)
Owner	In (<code>in</code>) Not In (<code>not-in</code>) Equals User ID (<code>eq-userid</code>) Not Equals User ID (<code>ne-userid</code>) Equals User Or Team (<code>eq-useroruserteams</code>)

Attribute Type	Supported Operations
String	Like (like) Not Like (not-like) Begins With (begins-with) Not Begin With (not-begin-with) Ends With (ends-with) Not End With (not-end-with)
DateTime	On Or After (on-or-after) On (on) On Or Before (on-or-before) Today (today) Tomorrow (tomorrow) Yesterday (yesterday) Next seven Days (next-seven-days) Last seven Days (last-seven-days) Next Week (next-week) Last Week (last-week) This Week (this-week) Next Month (next-month) Last Month (last-month) This Month (this-month) Next Year (next-year) Last Year (last-year) This Year (this-year) Last X Days (last-x-days) Next X Days (next-x-days) Last X Weeks (last-x-weeks) Next X Weeks (next-x-weeks) Last X Months (last-x-months) Next X Months (next-x-months) Last X Years (last-x-years) Next X Years (next-x-years) Greater Than (gt) Greater Than Or Equal (gte) Less Than (lt) Less Than Or Equal (lte)

Examples

Most of the scenarios/examples mentioned in [Query Data using the Web API](#) can be achieved using the **retrieveMultipleRecords** method. Some of the examples are listed below.

Basic retrieve multiple

This example queries the accounts table set and uses the `$select` and `$top` system query options to return the name property for the first three accounts:

JavaScript

```
Xrm.WebApi.retrieveMultipleRecords("account", "?$select=name&$top=3").then(  
    function success(result) {  
        for (var i = 0; i < result.entities.length; i++) {  
            console.log(result.entities[i]);  
        }  
        // perform additional operations on retrieved records  
    },  
    function (error) {  
        console.log(error.message);  
        // handle error conditions  
    }  
);
```

Basic retrieve multiple with FetchXML

This example queries the `account` entity using fetchXML.

JavaScript

```
var fetchXml = "?fetchXml=<fetch mapping='logical'><entity name='account'>  
<attribute name='accountid'/><attribute name='name'/></entity></fetch>";  
  
Xrm.WebApi.retrieveMultipleRecords("account", fetchXml).then(  
    function success(result) {  
        for (var i = 0; i < result.entities.length; i++) {  
            console.log(result.entities[i]);  
        }  
  
        // perform additional operations on retrieved records  
    },  
    function (error) {  
        console.log(error.message);  
        // handle error conditions  
    }  
);
```

Retrieve or filter by lookup properties

For most single-valued navigation properties you'll find a computed, read-only property that uses the following naming convention: `_<name>_value` where the `<name>` is the

name of the single-valued navigation property. For filtering purposes, the specific value of the single-valued navigation property can also be used. However, for mobile clients in offline mode, these syntax options aren't supported, and the single-value navigation property name should be used for both retrieving and filtering. Also, the comparison of navigation properties to null isn't supported in offline mode.

More information: [Lookup properties](#)

Here are code examples for both the scenarios:

For online scenario (connected to server)

This example queries the accounts table set and uses the `$select` and `$filter` system query options to return the name and primarycontactid property for accounts that have a particular primary contact:

JavaScript

```
Xrm.WebApi.retrieveMultipleRecords("account", "?  
$select=name,_primarycontactid_value&$filter=primarycontactid/contactid eq  
a0dbf27c-8efb-e511-80d2-00155db07c77").then(  
    function success(result) {  
        for (var i = 0; i < result.entities.length; i++) {  
            console.log(result.entities[i]);  
        }  
        // perform additional operations on retrieved records  
    },  
    function (error) {  
        console.log(error.message);  
        // handle error conditions  
    }  
);
```

For mobile offline scenario

This example queries the accounts table set and uses the `$select` and `$filter` system query options to return the name and primarycontactid property for accounts that have a particular primary contact when working in the offline mode:

JavaScript

```
Xrm.WebApi.retrieveMultipleRecords("account", "?  
$select=name,primarycontactid&$filter=primarycontactid eq a0dbf27c-8efb-  
e511-80d2-00155db07c77").then(  
    function success(result) {  
        for (var i = 0; i < result.entities.length; i++) {
```

```

        console.log(result.entities[i]);
    }
    // perform additional operations on retrieved records
},
function (error) {
    console.log(error.message);
    // handle error conditions
}
);

```

Using FetchXML to retrieve or filter by lookup properties (online and offline scenario)

You can use the `FetchXML` parameter while online or offline to retrieve the `name` and `primarycontactid` property for account records that have a primary contact that matches a condition:

JavaScript

```

var fetchXml = `?fetchXml=
<fetch mapping='logical'>
    <entity name='account'>
        <attribute name='name'/>
        <attribute name='primarycontactid'/>
        <link-entity name='contact' from='contactid'
to='primarycontactid'>
            <filter type='and'>
                <condition attribute='lastname' operator='eq'
value='Contoso' />
            </filter>
        </link-entity>
    </entity>
</fetch>`;

Xrm.WebApi.retrieveMultipleRecords("account", fetchXml).then(
    function success(result) {
        for (var i = 0; i < result.entities.length; i++) {
            console.log(result.entities[i]);
        }

        // perform additional operations on retrieved records
},
function (error) {
    console.log(error.message);
    // handle error conditions
}
);

```

Specify the number of tables to return in a page

The following example demonstrates the use of the `maxPageSize` parameter to specify the number of records (3) to be displayed in a page.

JavaScript

```
Xrm.WebApi.retrieveMultipleRecords("account", "?$select=name", 3).then(
    function success(result) {
        for (var i = 0; i < result.entities.length; i++) {
            console.log(result.entities[i]);
        }
        console.log("Next page link: " + result.nextLink);
        // perform additional operations on retrieved records
    },
    function (error) {
        console.log(error.message);
        // handle error conditions
    }
);
```

This example will display three records and a link to the next page. Here's an example output from the **Console** in the browser developer tools:

```
{@odata.etag: "W/\"1035541\"", name: "A. Datum", accountid: "475b158c-541c-e511-80d3-3863bb347ba8"}
@odata.etag: "W/\"1035541\""accountid: "475b158c-541c-e511-80d3-3863bb347ba8"name: "A. Datum"__proto__: Object
VM5595:4
{@odata.etag: "W/\"947306\"", name: "Adventure Works", accountid: "a8a19cdd-88df-e311-b8e5-6c3be5a8b200"}
VM5595:4
{@odata.etag: "W/\"1033754\"", name: "Alpine Ski House", accountid: "aaa19cdd-88df-e311-b8e5-6c3be5a8b200"}
VM5595:6
Next page link: [Organization URI]/api/data/v9.0/accounts?
$select=name&$skiptoken=%3Ccookie%20pagenumber=%222%22%20pagingcookie=%22%253ccookie%2520page%253d%25221%2522%253e%253caccountid%2520last%253d%2522%257bAAA19CDD-88DF-E311-B8E5-
6C3BE5A8B200%257d%2522%2520first%253d%2522%257b475B158C-541C-E511-80D3-3863BB347BA8%257d%2522%2520%252f%253e%253c%252fcookie%253e%22%20istracking=%22False%22%20/%3E
```

Use the query part in the URL in the `nextLink` property as the value for the `options` parameter in your subsequent `retrieveMultipleRecords` call to request the next set of records. Don't change or append any more system query options to the value. For every subsequent request for more pages, you should use the same `maxPageSize` value used in

the original retrieve multiple request. Also, cache the results returned or the value of the nextLink property so that previously retrieved pages can be returned.

For example, to get the next page of records, we'll pass in the query part of the `nextLink` URL to the `options` parameter:

JavaScript

```
Xrm.WebApi.retrieveMultipleRecords("account", "?  
$select=name&$skiptoken=%3Ccookie%20pagenumber=%222%22%20pagingcookie=%22%25  
3ccookie%2520page%253d%25221%2522%253e%253caccountid%2520last%253d%2522%257b  
AAA19CDD-88DF-E311-B8E5-  
6C3BE5A8B200%257d%2522%2520first%253d%2522%257b475B158C-541C-E511-80D3-  
3863BB347BA8%257d%2522%2520%252f%253e%253c%252fc(cookie%253e%22%20istracking=%  
22False%22%20/%3E", 3).then(  
    function success(result) {  
        for (var i = 0; i < result.entities.length; i++) {  
            console.log(result.entities[i]);  
        }  
        console.log("Next page link: " + result.nextLink);  
        // perform additional operations on retrieved records  
    },  
    function (error) {  
        console.log(error.message);  
        // handle error conditions  
    }  
);
```

This will return the next page of the resultset:

```
{@odata.etag: "W/"1035542"", name: "Blue Yonder Airlines", accountid:  
"aca19cdd-88df-e311-b8e5-6c3be5a8b200"}  
VM5597:4  
{@odata.etag: "W/"1031348"", name: "City Power & Light", accountid:  
"aea19cdd-88df-e311-b8e5-6c3be5a8b200"}  
VM5597:4  
{@odata.etag: "W/"1035543"", name: "Coho Winery", accountid: "b0a19cdd-88df-  
e311-b8e5-6c3be5a8b200"}  
VM5597:6  
Next page link: [Organization URI]/api/data/v9.0/accounts?  
$select=name&$skiptoken=%3Ccookie%20pagenumber=%223%22%20pagingcookie=%22%25  
3ccookie%2520page%253d%25222%2522%253e%253caccountid%2520last%253d%2522%257b  
B0A19CDD-88DF-E311-B8E5-  
6C3BE5A8B200%257d%2522%2520first%253d%2522%257bACA19CDD-88DF-E311-B8E5-  
6C3BE5A8B200%257d%2522%2520%252f%253e%253c%252fc(cookie%253e%22%20istracking=%  
22False%22%20/%3E
```

ⓘ Important

The value of the `nextLink` property is URI encoded. If you URI encode the value before you send it, the XML cookie information in the URL will cause an error.

FetchXML Example (online scenario)

The following example demonstrates the use of the `count` parameter of the FetchXML to specify the number of records (3) to be displayed in a page.

ⓘ Note

The FetchXML paging cookie is only returned for online `retrieveMultipleRecords` operations. (`Xrm.WebApi.online`). It is not supported offline.

JavaScript

```
var fetchXml = "?fetchXml=<fetch mapping='logical' count='3'><entity name='account'><attribute name='accountid'/><attribute name='name'/></entity></fetch>";

Xrm.WebApi.online.retrieveMultipleRecords("account", fetchXml).then(
    function success(result) {
        for (var i = 0; i < result.entities.length; i++) {
            console.log(result.entities[i]);
        }

        console.log("Paging cookie: " + result.fetchXmlPagingCookie);

        // perform additional operations on retrieved records
    },
    function (error) {
        console.log(error.message);
        // handle error conditions
    }
);
```

This example will display three records and return a FetchXML Paging Cookie to the retrieve the results of the next page if there are more records belonging to the result set. Here's an example output from the **Console** in the browser developer tools:

JSON

```
{
    "entities": [
```

```

    },
    "@odata.etag": "W/\"1035542\"",
    "accountid": "aca19cdd-88df-e311-b8e5-6c3be5a8b200",
    "name": "Blue Yonder Airlines"
},
{
    "@odata.etag": "W/\"1031348\"",
    "accountid": "aea19cdd-88df-e311-b8e5-6c3be5a8b200",
    "name": "City Power & Light"
},
{
    "@odata.etag": "W/\"1035543\"",
    "accountid": "b0a19cdd-88df-e311-b8e5-6c3be5a8b200",
    "name": "Coho Winery"
}
],
"fetchXmlPagingCookie": "<cookie pagenumber=\"2\""
pagingcookie=\"%253ccookie%2520page%253d%25221%2522%253e%253ccountid%25201
ast%253d%2522%2527b0748C6EC-55A8-EB11-B1B5-
000D3AFEF6FA%257d%2522%2520first%253d%2522%2527bFC47C6EC-55A8-EB11-B1B5-
000D3AFEF6FA%257d%2522%2520%252f%253e%253c%252fcookie%253e\""
istracking=\"False\" />"
}

```

We can use the `fetchXmlPagingCookie` as shown in the example below to fetch large result sets with paging.

JavaScript

```

function CreateXml(fetchXml, pagingCookie, page, count) {
    var domParser = new DOMParser();
    var xmlSerializer = new XMLSerializer();

    var XmlDocument = domParser.parseFromString(fetchXml, "text/xml");

    if (page) {
        XmlDocument
            .getElementsByTagName("fetch")[0]
            .setAttribute("page", page.toString());
    }

    if (count) {
        XmlDocument
            .getElementsByTagName("fetch")[0]
            .setAttribute("count", count.toString());
    }

    if (pagingCookie) {
        var cookieDoc = domParser.parseFromString(pagingCookie, "text/xml");
        var innerPagingCookie = domParser.parseFromString(
            decodeURIComponent(
                decodeURIComponent(
                    cookieDoc

```

```

        .getElementsByTagName("cookie")[0]
        .getAttribute("pagingcookie")
    )
),
"text/xml"
);
fetchXmlDocument
    .getElementsByTagName("fetch")[0]
    .setAttribute(
        "paging-cookie",
        xmlSerializer.serializeToString(innerPagingCookie)
    );
}

return xmlSerializer.serializeToString(fetchXmlDocument);
}

function retrieveAllRecords(entityName, fetchXml, page, count, pagingCookie)
{
    if (!page) {
        page = 0;
    }

    return retrievePage(entityName, fetchXml, page + 1, count,
pagingCookie).then(
    function success(pageResults) {
        if (pageResults.fetchXmlPagingCookie) {
            return retrieveAllRecords(
                entityName,
                fetchXml,
                page + 1,
                count,
                pageResults.fetchXmlPagingCookie
            ).then(
                function success(results) {
                    if (results) {
                        return pageResults.entities.concat(results);
                    }
                },
                function error(e) {
                    throw e;
                }
            );
        } else {
            return pageResults.entities;
        }
    },
    function error(e) {
        throw e;
    }
);
}

function retrievePage(entityName, fetchXml, pageNumber, count, pagingCookie)
{

```

```

var fetchXml =
    "?fetchXml=" + CreateXml(fetchXml, pagingCookie, pageNumber, count);

return Xrm.WebApi.online.retrieveMultipleRecords(entityName,
fetchXml).then(
    function success(result) {
        return result;
    },
    function error(e) {
        throw e;
    }
);
}

var count = 3;
var fetchXml =
    '<fetch mapping="logical"><entity name="account"><attribute
name="accountid"/><attribute name="name"/></entity></fetch>';

retrieveAllRecords("account", fetchXml, null, count, null).then(
    function success(result) {
        console.log(result);

        // perform additional operations on retrieved records
    },
    function error(error) {
        console.log(error.message);
        // handle error conditions
    }
);

```

Retrieve related tables by expanding navigation properties

Use the `$expand` system query option in the navigation properties to control the data that is returned from related tables. The following example demonstrates how to retrieve the contact for all the account records. For the related contact records, we're only retrieving the `contactid` and `fullname`:

JavaScript

```

Xrm.WebApi.retrieveMultipleRecords("account", "?"
$select=name&$top=3&$expand=primarycontactid($select=contactid,fullname)",
3).then(
    function success(result) {
        for (var i = 0; i < result.entities.length; i++) {
            console.log(result.entities[i]);
        }
        // perform additional operations on retrieved records
    }
);

```

```
},
function (error) {
    console.log(error.message);
    // handle error conditions
}
);
```

The above piece of code returns a result with a schema like:

JSON

```
{
    "entities": [
        {
            "@odata.etag": "W/\"1459919\"",
            "name": "Test Account",
            "accountid": "119edfac-19c6-ea11-a81a-000d3af5e732",
            "primarycontactid": {
                "contactid": "6c63a1b7-19c6-ea11-a81a-000d3af5e732",
                "fullname": "Test Contact"
            }
        }
    ]
}
```

ⓘ Note

Similar to the online scenario, use the `$expand` system query option to retrieve data from related tables in offline. However, many-to-many relationships are not supported in offline.

Deprecated method for mobile offline scenario

ⓘ Note

The `@odata.nextLink` is deprecated for mobile offline scenarios. While it is still supported for existing customizations, it is not recommended to use it anymore.

An offline `$expand` operation returns a `@odata.nextLink` annotation containing information on how to get to the related record's information. We use the `id`, `entityType`, and `options` parameter of that annotation to construct one or more additional `Xrm.WebApi.offline.retrieveRecord` request(s). The following piece of code provides a complete example of how to do this:

JavaScript

```
Xrm.WebApi.offline.retrieveMultipleRecords("account", "?$select=name&$top=3&$expand=primarycontactid($select=contactid,fullname)").then(function(resultSet) {
    /**
     * resultSet has a structure like:
     * {
     *     "entities": [
     *         {
     *             "accountid": "119edfac-19c6-ea11-a81a-000d3af5e732",
     *             "name": "Test Account",
     *             "primarycontactid@odata.nextLink": {
     *                 "API": "{Xrm.Mobile.offline}.{retrieveRecord}",
     *                 "id": "119edfac-19c6-ea11-a81a-000d3af5e732",
     *                 "entityType": "account",
     *                 "options": "?"
     *             },
     *             "primarycontactid": {}
     *         }
     *     ]
     * }
     *
     * Notice the empty `primarycontactid` property but an additional
     `primarycontactid@odata.nextLink`
     * annotation that lets us know how to get to the linked data that we
     need.
    */
    var promises = resultSet.entities.map(function(outerItem) {
        // We do a retrieveRecord() for every item in the result set of
        retrieveMultipleRecords() and then
        // combine the results into the retrieveMultipleRecords() result set
        // itself.
        return Xrm.WebApi.offline.retrieveRecord(
            outerItem["primarycontactid@odata.nextLink"].entityType,
            outerItem["primarycontactid@odata.nextLink"].id,
            outerItem["primarycontactid@odata.nextLink"].options
        ).then(function(innerResult) {
            if (innerResult.value.length === 0) {
                return outerItem;
            }
            outerItem.primarycontactid = innerResult.value[0];
            return outerItem;
        });
    });

    return Promise.all(promises);
}).then(function(allResults) {
    for (var i = 0; i < allResults.length; i++) {
        console.log(allResults[i]);
    }
})
```

```
// perform additional operations on retrieved records
}, function(error) {
    console.error(error);
    // handle error conditions
});
```

For more examples of retrieving multiple records using Web API, see [Query Data using the Web API](#).

See also

[Query Data using the Web API](#)

[Xrm.WebApi.retrieveRecord](#)

[Xrm.WebApi](#)

updateRecord (Client API reference)

Article • 08/22/2022

Updates a table record.

Syntax

```
Xrm.WebApi.updateRecord(entityLogicalName, id, data).then(successCallback,  
errorCallback);
```

Parameters

Name	Type	Required	Description
entityLogicalName	String	Yes	The table logical name of the record you want to update. For example: "account".
id	String	Yes	GUID of the table record you want to update.
data	Object	Yes	A JSON object containing <code>key: value</code> pairs, where `key` is the property of the table and <code>value</code> is the value of the property you want to update. See examples later in this topic to see how you can define the <code>data</code> object for various update scenarios.
successCallback	Function	No	A function to call when a record is updated. An object with the following properties will be passed to identify the updated record: <ul style="list-style-type: none">• <code>entityType</code>: String. The table type of the updated record.• <code>id</code>: String. GUID of the updated record.
errorCallback	Function	No	A function to call when the operation fails. An object with the following properties will be passed: <ul style="list-style-type: none">• <code>errorCode</code>: Number. The error code.• <code>message</code>: String. An error message describing the issue.

Return Value

On success, returns a promise object containing the values specified earlier in the description of the **successCallback** parameter.

Examples

These examples use some of the same request objects as demonstrated in [Update and delete table rows using the Web API](#) to define the data object for updating a table record.

Basic update

Updates an existing account record with record ID = 5531d753-95af-e711-a94e-000d3a11e605.

JavaScript

```
// define the data to update a record
var data =
{
    "name": "Updated Sample Account",
    "creditonhold": true,
    "address1_latitude": 47.639583,
    "description": "This is the updated description of the sample
account",
    "revenue": 6000000,
    "accountcategorycode": 2
}
// update the record
Xrm.WebApi.updateRecord("account", "5531d753-95af-e711-a94e-000d3a11e605",
data).then(
    function (success) {
        console.log("Account updated");
        // perform operations on record update
    },
    function (error) {
        console.log(error.message);
        // handle error conditions
    }
);
```

Update associations to the related tables

To update association to the related table records (lookups), set the value of single-valued navigation properties using the `@odata.bind` annotation to another record.

Here is code example:

The following example updates an account record to associate another contact record as the primary contact for the account:

JavaScript

```
// define the data to update a record
var data =
{
    "primarycontactid@odata.bind": "/contacts(61a0e5b9-88df-e311-b8e5-6c3be5a8b200)"
}
// update the record
Xrm.WebApi.updateRecord("account", "5531d753-95af-e711-a94e-000d3a11e605",
data).then(
    function success(result) {
        console.log("Account updated");
        // perform operations on record update
    },
    function (error) {
        console.log(error.message);
        // handle error conditions
    }
);
```

Deprecated method for mobile offline scenario

ⓘ Note

Instead of using `@odata.bind` annotation example above, the deprecated `lookup` object with case-sensitive properties (`logicalname` and `id`) is still supported for existing customizations. However, it is recommended to use `@odata.bind` annotation for both online and offline scenario instead of using this deprecated object.

The following example uses the deprecated method to update an account record to associate another contact record as the primary contact for the account from mobile clients when working in the offline mode:

JavaScript

```
// define the data to update a record
var data =
{
    "primarycontactid":
    {
        "logicalname": "contact",
        "id": "61a0e5b9-88df-e311-b8e5-6c3be5a8b200"
```

```

        }
    }
    // update the record
    Xrm.WebApi.offline.updateRecord("account", "5531d753-95af-e711-a94e-000d3a11e605", data).then(
        function success(result) {
            console.log("Account updated");
            // perform operations on record update
        },
        function (error) {
            console.log(error.message);
            // handle error conditions
        }
    );

```

Update associations to the related tables of type Activity

To update association to the related tables of type Activity, set the value of single-valued navigation properties using the `@odata.bind` annotation to another record.

Update related opportunity column on task

JavaScript

```

// define the data to update a record
var data =
{
    "new_relatedopportunities_task@odata.bind":
    "/opportunities(61a0e5b9-88df-e311-b8e5-6c3be5a8b200)"
}
// update the record
Xrm.WebApi.updateRecord("task", "5531d753-95af-e711-a94e-000d3a11e605",
data).then(
    function success(result) {
        console.log("Task updated");
        // perform operations on record update
    },
    function (error) {
        console.log(error.message);
        // handle error conditions
    }
);

```

Update Regarding column on task

JavaScript

```

// define the data to update a record
var data =
{

```

```
    "regardingobjectid_account_task@odata.bind": "/accounts(61a0e5b9-  
88df-e311-b8e5-6c3be5a8b200)"  
}  
// update the record  
Xrm.WebApi.updateRecord("task", "5531d753-95af-e711-a94e-000d3a11e605",  
data).then(  
    function success(result) {  
        console.log("Task updated");  
        // perform operations on record update  
    },  
    function (error) {  
        console.log(error.message);  
        // handle error conditions  
    }  
);
```

Update associations for collection-valued navigation properties

The [Xrm.WebApi.online.execute](#) API can be used to associate and disassociate collection-valued navigation properties. This is **NOT** supported for mobile offline scenarios.

Related topics

[Xrm.WebApi](#)

isAvailableOffline (Client API reference)

Article • 11/30/2022

Returns a boolean value indicating whether an entity is present in user's profile and is currently available for use in offline mode.

Syntax

```
Xrm.WebApi.offline.isAvailableOffline(entityLogicalName);
```

Parameters

Name	Type	Required	Description
entityLogicalName	String	Yes	Logical name of the table. For example: "account".

Return Value

Type: Boolean.

Description: true if the table is present in user's profile and is currently available for use in offline mode; otherwise false.

[Xrm.WebApi.offline](#)

[Xrm.WebApi](#)

Xrm.WebApi.online.execute (Client API reference)

Article • 11/30/2022

Execute a single action, function, or CRUD operation.

ⓘ Note

This method is supported only for the online mode (`Xrm.WebApi.online`).

Syntax

```
Xrm.WebApi.online.execute(request).then(successCallback, errorCallback);
```

Parameters

Name	Type	Required	Description
request	Object	Yes	<p>Object that will be passed to the Web API endpoint to execute an action, function, or CRUD request. The object exposes a <code>getMetadata</code> method via its prototype that lets you define the metadata for the action, function or CRUD request you want to execute. The <code>getMetadata</code> method has the following parameters:</p> <ul style="list-style-type: none">• boundParameter: (Optional) String. The name of the bound parameter for the action or function to execute.<ul style="list-style-type: none">◦ Specify <code>undefined</code> if you are executing a CRUD request.◦ Specify <code>null</code> if the action or function to execute is not bound to any table.◦ Specify <code>entity</code> in case the action or function to execute is bound to a table.• operationName: (Optional). String. Name of the action, function, or one of the following values if you are executing a CRUD request: "Create", "Retrieve", "Update", or "Delete".• operationType: (Optional). Number. Indicates the type of operation you are executing; specify one of the following values:<ul style="list-style-type: none">0: Action

1: Function

2: CRUD

- **parameterTypes**: Object. The metadata for parameter types. The object has the following values:
 - **enumProperties**: (Optional) Object. The metadata for enum types. The object has two string values: **name** and **value**
 - **structuralProperty**: Number. The category of the parameter type. Specify one of the following values:
 - 0: Unknown
 - 1: PrimitiveType
 - 2: ComplexType
 - 3: EnumerationType
 - 4: Collection
 - 5: EntityType
 - **typeName**: String. The fully qualified name of the parameter type.

successCallback	Function	No	A function to call when operation is executed successfully. A response object is passed to the function with the following values: <ul style="list-style-type: none">• body (Deprecated): Object. Response body.• headers: Object. Response headers.• ok: Boolean. Indicates whether the request was successful.• status: Number. Numeric value in the response status code. For example: 200• statusText: String. Description of the response status code. For example: OK• type (Deprecated): String. Response type. Values are: the empty string (default), "arraybuffer", "blob", "document", "json", and "text".• url: String. Request URL of the action, function, or CRUD request that was sent to the Web API endpoint.• json: Promise. Parameter to the callback delegate is of type any (JSON object).• text: Promise. Parameter to the callback delegate is a String.
errorCallback	Function	No	A function to call when the operation fails.

Return Value

On success, returns a promise object with the values specified earlier in the description of `successCallback` function.

Examples

Execute an action

The following example demonstrates how to execute the `WinOpportunity` action found in the Dynamics 365 for Sales solution. The request object is created based on the action definition here: [Unbound actions](#)

JavaScript

```
var Sdk = window.Sdk || {};
/**
 * Request to win an opportunity
 * @param {Object} opportunityClose - The opportunity close activity
associated with this state change.
 * @param {number} status - Status of the opportunity.
 */
Sdk.WinOpportunityRequest = function(opportunityClose, status) {
    this.OpportunityClose = opportunityClose;
    this.Status = status;
};

// NOTE: The getMetadata property should be attached to the function
// prototype instead of the
// function object itself.
Sdk.WinOpportunityRequest.prototype.getMetadata = function () {
    return {
        boundParameter: null,
        parameterTypes: {
            "OpportunityClose": {
                "typeName": "mscrm.opportunityclose",
                "structuralProperty": 5 // Entity Type
            },
            "Status": {
                "typeName": "Edm.Int32",
                "structuralProperty": 1 // Primitive Type
            }
        },
        operationType: 0, // This is an action. Use '1' for functions and
'2' for CRUD
        operationName: "WinOpportunity",
    };
};

var opportunityClose = {
    "opportunityid@odata.bind": "/opportunities(c60e0283-5bf2-e311-945f-
6c3be5a8dd64)",
```

```

        "description": "Product and maintenance for 2018",
        "subject": "Contract for 2018"
    }

    // Construct a request object from the metadata
    var winOpportunityRequest = new Sdk.WinOpportunityRequest(opportunityClose,
3);

    // Use the request object to execute the function
    Xrm.WebApi.online.execute(winOpportunityRequest).then(function (response) {
        if (response.ok) {
            console.log("Status: %s %s", response.status, response.statusText);
            // The WinOpportunityRequest does not return any response body
            content. So we
            // need not access the response.json() property.

            // Perform other operations as required.
        }
    })
    .catch(function(error) {
        console.log(error.message);
        // handle error conditions
    });

```

Execute a function

The following example demonstrates how to execute the [WhoAmI](#) function:

JavaScript

```

var Sdk = window.Sdk || {};
/**
 * Request to execute WhoAmI function
 */
Sdk.WhoAmIRequest = function () { };

// NOTE: The getMetadata property should be attached to the function
// prototype instead of the
//     function object itself.
Sdk.WhoAmIRequest.prototype.getMetadata = function () {
    return {
        boundParameter: null,
        parameterTypes: {},
        operationType: 1, // This is a function. Use '0' for actions and '2'
for CRUD
        operationName: "WhoAmI",
    };
};

// Construct a request object from the metadata
var whoAmIRequest = new Sdk.WhoAmIRequest();

```

```

// Use the request object to execute the function
Xrm.WebApi.online.execute(whoAmIRequest)
.then(function (response) {
    if (response.ok) {
        console.log("Status: %s %s", response.status, response.statusText);

        // Use response.json() to access the content of the response body.
        return response.json();
    }
})
.then(function (responseBody) {
    console.log("User Id: %s", responseBody.UserId);
    // perform other operations as required;
})
.catch(function (error) {
    console.log(error.message);
    // handle error conditions
});

```

The following example demonstrates how to execute the [CalculateRollupField](#) function:

JavaScript

```

var Sdk = window.Sdk || {};

Sdk.CalculateRollupFieldRequest = function(target, fieldName) {
    this.Target = target;
    this.FieldName = fieldName;
};

// NOTE: The getMetadata property should be attached to the function
// prototype instead of the
//     function object itself.
Sdk.CalculateRollupFieldRequest.prototype.getMetadata = function() {
    return {
        boundParameter: null,
        parameterTypes: {
            "Target": {
                "typeName": "mscrm.crmbaseentity",
                "structuralProperty": 5
            },
            "FieldName": {
                "typeName": "Edm.String",
                "structuralProperty": 1
            }
        },
        operationType: 1, // This is a function. Use '0' for actions and '2'
        for CRUD
            operationName: "CalculateRollupField"
        };
};

```

```

// Create variables to point to a quote record and to a specific column
var quoteId = {
    "@odata.type": "Microsoft.Dynamics.CRM.quote",
    "quoteid": "7bb01e55-2394-ea11-a811-000d3ad97943"
};

// The roll-up column for which we want to force a re-calculation
var fieldName = "new_test_rollup";

// Create variable calculateRollupFieldRequest and pass those variables
// created above
var calculateRollupFieldRequest = new
Sdk.CalculateRollupFieldRequest(quoteId, fieldName);

// Use the request object to execute the function
Xrm.WebApi.online.execute(calculateRollupFieldRequest)
.then(function(response) {
    if (response.ok) { // If a response was received.
        console.log("Status: %s %s", response.status, response.statusText);

        // Use response.json() to access the content of the response body.
        return response.json();
    }
})
.then(function(responseBody) {
    //Do something with the response
    console.log("The response is: %s", responseBody);
})
.catch(function(error) {
    console.log(error.message);
    // handle error conditions
});

```

The following example demonstrates how to execute the [RetrieveDuplicates](#) function:

JavaScript

```

var Sdk = window.Sdk || {};

Sdk.RetrieveDuplicatesRequest = function(businessEntity, matchingEntityName,
pagingInfo) {
    this.BusinessEntity = businessEntity;
    this.MatchingEntityName = matchingEntityName;
    this.PagingInfo = pagingInfo;

};

// NOTE: The getMetadata property should be attached to the function
// prototype instead of the
// function object itself.
Sdk.RetrieveDuplicatesRequest.prototype.getMetadata = function() {
    return {
        boundParameter: null,

```

```

parameterTypes: {
    "BusinessEntity": {
        "typeName": "mscrm.crmbaseentity",
        "structuralProperty": 5 // Entity Type
    },
    "MatchingEntityName": {
        "typeName": "Edm.String",
        "structuralProperty": 1 // Primitive Type
    },
    "PagingInfo": {
        "typeName": "mscrm.PagingInfo", // Complex Type
        "structuralProperty": 5
    }
},
operationType: 1, // This is a function. Use '0' for actions and '2' for CRUD
operationName: "RetrieveDuplicates",
};

// Create a variable to point to a contact record and with specific data in the needed columns
var contactRecord = {
    "@odata.type": "Microsoft.Dynamics.CRM.contact",
    "firstname": "Test",
    "lastname": "Account"
};

// Create a paging object to keep track of the current page and how many records we get per page
var pagingInfo = {
    "PageNumber": 1,
    "Count": 10
};

// Create the variable retrieveDuplicatesRequest to build the request
var retrieveDuplicatesRequest = new Sdk.RetrieveDuplicatesRequest(contactRecord, "contact", pagingInfo);

// Use the request object to execute the function
Xrm.WebApi.online.execute(retrieveDuplicatesRequest)
.then(function (response) {
    if (response.ok) {
        console.log("Status: %s %s", response.status, response.statusText);

        // Use response.json() to access the content of the response body.
        return response.json();
    }
})
.then(function(responseBody) {
    // Do something with the response
    console.log("The response is: %s", responseBody);
})
.catch(function(error) {
    console.log(error.message);
})

```

```
// handle error conditions
});
```

The following example demonstrates how to execute the [InitializeFrom](#) function:

JavaScript

```
var Sdk = window.Sdk || {};

Sdk.InitializeFromRequest = function (
    entityMoniker,
    targetEntityName,
    targetFieldType
) {
    this.EntityMoniker = entityMoniker;
    this.TargetEntityName = targetEntityName;
    this.TargetFieldType = targetFieldType;
};

// NOTE: The getMetadata property should be attached to the function
// prototype instead of the
// function object itself.
Sdk.InitializeFromRequest.prototype.getMetadata = function () {
    return {
        boundParameter: null,
        parameterTypes: {
            EntityMoniker: {
                typeName: "mscrm.crmbaseentity",
                structuralProperty: 5, //Entity Type
            },
            TargetEntityName: {
                typeName: "Edm.String",
                structuralProperty: 1, // PrimitiveType
            },
            TargetFieldType: {
                typeName: "Microsoft.Dynamics.CRM.TargetFieldType",
                structuralProperty: 3, // Enum Type
                enumProperties: [
                    {
                        name: "All",
                        value: 0,
                    },
                    {
                        name: "ValidForCreate",
                        value: 1,
                    },
                    {
                        name: "ValidForUpdate",
                        value: 2,
                    },
                    {
                        name: "ValidForRead",
                        value: 3,
                    }
                ]
            }
        }
    };
}
```

```

        },
    ],
},
},
operationType: 1, // This is a function. Use '0' for actions and '2' for
CRUD
operationName: "InitializeFrom",
};

// Create a variable to point to tje parent account record
var parentAccountRecord = {
"@odata.type": "Microsoft.Dynamics.CRM.account",
accountid: "141da047-eaad-eb11-b1b4-000d3ac581a0",
};

// Create a variable for the target entity name
var targetEntityName = "account";

// Create a variable for the target field type
var targetFieldType = 0;

// Build the request
var initializeFromRequest = new Sdk.InitializeFromRequest(
parentAccountRecord,
targetEntityName,
targetFieldType
);

// Execute the request
Xrm.WebApi.online.execute(initializeFromRequest)
.then(function (response) {
if (response.ok) {
console.log("Status: %s %s", response.status, response.statusText);

// Use response.json() to access the content of the response body.
return response.json();
}
})
.then(function(responseBody) {
// Do something with the response
console.log("The response is: %s", responseBody);
})
.catch(function(error) {
console.log(error.message);
// handle error conditions
});

```

Perform CRUD operations

Create a record

The following example demonstrates how to perform a create operation.

JavaScript

```
var Sdk = window.Sdk || {};  
  
/**  
 * Request to execute a create operation  
 */  
Sdk.CreateRequest = function(entityName, payload) {  
    this.etn = entityName;  
    this.payload = payload;  
};  
  
// NOTE: The getMetadata property should be attached to the function  
prototype instead of the  
// function object itself.  
Sdk.CreateRequest.prototype.getMetadata = function () {  
    return {  
        boundParameter: null,  
        parameterTypes: {},  
        operationType: 2, // This is a CRUD operation. Use '0' for actions  
        and '1' for functions  
        operationName: "Create",  
    };  
};  
// Construct a request object from the metadata  
var payload = {  
    name: "Fabrikam Inc."  
};  
var createRequest = new Sdk.CreateRequest("account", payload);  
  
// Use the request object to execute the function  
Xrm.WebApi.online.execute(createRequest)  
.then(function (response) {  
    if (response.ok) {  
        console.log("Status: %s %s", response.status, response.statusText);  
  
        // The Create request does not return any response body content. So  
        we  
        // need not access the response.json() property.  
  
        // Perform other operations as required.  
    }  
})  
.catch(function(error) {  
    console.log(error.message);  
    // handle error conditions  
});
```

Retrieve a record

The following example demonstrates how to perform a retrieve operation.

JavaScript

```
var Sdk = window.Sdk || {};  
  
/**  
 * Request to execute a retrieve operation  
 */  
Sdk.RetrieveRequest = function(entityReference, columns) {  
    this.entityReference = entityReference;  
    this.columns = columns;  
};  
// NOTE: The getMetadata property should be attached to the function  
prototype instead of the  
// function object itself.  
Sdk.RetrieveRequest.prototype.getMetadata = function () {  
    return {  
        boundParameter: null,  
        parameterTypes: {},  
        operationType: 2, // This is a CRUD operation. Use '0' for actions  
and '1' for functions  
        operationName: "Retrieve",  
    };  
};  
  
// Construct request object from the metadata  
var entityReference = {  
    entityType: "account",  
    id: "d2b6c3f8-b0fa-e911-a812-000d3a59fa22"  
};  
var retrieveRequest = new Sdk.RetrieveRequest(entityReference, ["name"]);  
  
// Use the request object to execute the function  
Xrm.WebApi.online.execute(retrieveRequest)  
.then(function (response) {  
    if (response.ok) {  
        console.log("Status: %s %s", response.status, response.statusText);  
  
        // Use response.json() to access the content of the response body.  
        return response.json();  
    }  
})  
.then(function(responseBody) {  
    console.log("Name: %s", responseBody.name);  
  
    // perform other operations as required;  
})  
.catch(function(error) {  
    console.log(error.message);  
    // handle error conditions  
});
```

Update a record

The following example demonstrates how to perform a update operation.

JavaScript

```
var Sdk = window.Sdk || {};  
  
/**  
 * Request to execute an update operation  
 */  
Sdk.UpdateRequest = function(entityName, entityId, payload) {  
    this.etn = entityName;  
    this.id = entityId;  
    this.payload = payload;  
};  
  
// NOTE: The getMetadata property should be attached to the function  
prototype instead of the  
// function object itself.  
Sdk.UpdateRequest.prototype.getMetadata = function () {  
    return {  
        boundParameter: null,  
        parameterTypes: {},  
        operationType: 2, // This is a CRUD operation. Use '0' for actions  
and '1' for functions  
        operationName: "Update",  
    };  
};  
  
// Construct a request object from the metadata  
var payload = {  
    name: "Updated Sample Account"  
};  
var updateRequest = new Sdk.UpdateRequest("account", "d2b6c3f8-b0fa-e911-  
a812-000d3a59fa22", payload);  
  
// Use the request object to execute the function  
Xrm.WebApi.online.execute(updateRequest)  
.then(function (response) {  
    if (response.ok) {  
        console.log("Status: %s %s", response.status, response.statusText);  
  
        // The Update request does not return any response body content. So  
we  
        // need not access the response.json() property.  
  
        // perform other operations as required;  
    }  
})
```

```
.catch(function(error) {
  console.log(error.message);
  // handle error conditions
});
```

Delete a record

The following example demonstrates how to perform a delete operation.

JavaScript

```
var Sdk = window.Sdk || {};

/**
 * Request to execute a delete operation
 */
Sdk.DeleteRequest = function(entityReference) {
  this.entityReference = entityReference;
};

// NOTE: The getMetadata property should be attached to the function
// prototype instead of the
// function object itself.
Sdk.DeleteRequest.prototype.getMetadata = function () {
  return {
    boundParameter: null,
    parameterTypes: {},
    operationType: 2, // This is a CRUD operation. Use '0' for
    actions and '1' for functions
    operationName: "Delete",
  };
};

// Construct request object from the metadata
var entityReference = {
  entityType: "account",
  id: "d2b6c3f8-b0fa-e911-a812-000d3a59fa22"
};
var deleteRequest = new Sdk.DeleteRequest(entityReference);

// Use the request object to execute the function
Xrm.WebApi.online.execute(deleteRequest)
.then(function(response) {
  if (response.ok) {
    console.log("Status: %s %s", response.status, response.statusText);

    // The Delete request does not return any response body content. So
    we
    // need not access the response.json() property.
  }
});
```

```

        // perform other operations as required;
    }
})
.catch(function(error) {
    console.log(error.message);
    // handle error conditions
});

```

Associate a record

The following code sample demonstrates how to perform an Associate operation on collection-valued navigation properties (One-To-Many and Many-To-Many relationships). For single-valued navigation properties (Many-To-One relationships a.k.a Lookup columns), you can perform an Update operation as shown above or use [Xrm.WebApi.updateRecord](#).

JavaScript

```

var Sdk = window.Sdk || {};

/*
 * Request to execute an Associate operation.
 */
Sdk.AssociateRequest = function(target, relatedEntities, relationship) {
    this.target = target;
    this.relatedEntities = relatedEntities;
    this.relationship = relationship;
};

// NOTE: The getMetadata property should be attached to the function
// prototype instead of the
// function object itself.
Sdk.AssociateRequest.prototype.getMetadata = function() {
    return {
        boundParameter: null,
        parameterTypes: {},
        operationType: 2, // Associate and Disassociate fall under the CRUD
        umbrella
        operationName: "Associate"
    }
};

// Construct the target EntityReference object
var target = {
    entityType: "account",
    id: "0b4abc7d-7619-eb11-8dff-000d3ac5c7f9"
};

// Construct the related EntityReferences that the Target will be associated
// with.

```

```

var relatedEntities = [
    {
        entityType: "contact",
        id: "180a9aad-7619-eb11-8dff-000d3ac5c7f9"
    },
    {
        entityType: "contact",
        id: "753c58b4-7619-eb11-8dff-000d3ac5c7f9"
    }
];

// The name of the existing relationship to associate on.
var relationship = "new_account_contact";

var manyToManyAssociateRequest = new Sdk.AssociateRequest(target,
relatedEntities, relationship)

Xrm.WebApi.online.execute(manyToManyAssociateRequest)
.then(function(response) {
    if (response.ok) {
        console.log("Status: %s %s", response.status, response.statusText);

        // The Associate request does not return any response body content.
So we
        // need not access the response.json() property.

        // perform other operations as required;
    }
})
.catch(function(error) {
    console.log(error.message);
    // handle error conditions
});

```

Disassociate a record

The following code sample demonstrates how to perform a Disassociate operation on collection-valued navigation properties (One-To-Many and Many-To-Many relationships). For single-valued navigation properties (Many-To-One relationships a.k.a Lookup columns), you can perform an Update operation as shown above or use [Xrm.WebApi.updateRecord](#).

Note

Unlike the Associate operation which allows associating the target entity record with multiple related entity records in a single operation, the Disassociate operation is limited to only disassociating one entity record from the target entity record per operation.

JavaScript

```
var Sdk = window.Sdk || {};  
  
/*  
 * Request to execute a Disassociate operation.  
 */  
Sdk.DisassociateRequest = function(target, relatedEntityId, relationship) {  
    this.target = target;  
    this.relatedEntityId = relatedEntityId;  
    this.relationship = relationship;  
};  
  
// NOTE: The getMetadata property should be attached to the function  
prototype instead of the  
// function object itself.  
Sdk.DisassociateRequest.prototype.getMetadata = function() {  
    return {  
        boundParameter: null,  
        parameterTypes: {},  
        operationType: 2, // Associate and Disassociate fall under the CRUD  
umbrella  
        operationName: "Disassociate"  
    }  
};  
  
// Construct the target EntityReference object  
var target = {  
    entityType: "account",  
    id: "0b4abc7d-7619-eb11-8dff-000d3ac5c7f9"  
};  
  
// The GUID of the related entity record to disassociate.  
var relatedEntityId = "180a9aad-7619-eb11-8dff-000d3ac5c7f9";  
  
// The name of the existing relationship to disassociate from.  
var relationship = "new_account_contact";  
  
var manyToManyDisassociateRequest = new Sdk.DisassociateRequest(target,  
relatedEntityId, relationship)  
  
Xrm.WebApi.online.execute(manyToManyDisassociateRequest)  
.then(function(response) {  
    if (response.ok) {  
        console.log("Status: %s %s", response.status, response.statusText);  
  
        // The Disassociate request does not return any response body  
content. So we  
        // need not access the response.json() property.  
  
        // perform other operations as required;  
    }  
})  
.catch(function(error) {
```

```
    console.log(error.message);
    // handle error conditions
});
```

Related topics

[Xrm.WebApi](#)

Xrm.WebApi.online.executeMultiple (Client API reference)

Article • 11/30/2022

Execute a collection of action, function, or CRUD operations.

ⓘ Note

This method is supported only for the online mode ([Xrm.WebApi.online](#)).

If you want to execute multiple requests in a transaction, you must pass in a change set as a parameter to this method. [Change sets](#) represent a collection of operations that are executed in a transaction. You can also pass in individual requests and change sets together as parameters to this method.

ⓘ Note

- You cannot include read operations (retrieve, retrieve multiple, and Web API functions) as part of a change set; this is as per the OData v4 specifications.
- Requests can contain up to 1000 individual requests and cannot contain other batches. More information: [Execute batch operations](#).

Syntax

Execute multiple requests:

JavaScript

```
var requests = [req1, req2, req3];
Xrm.WebApi.online.executeMultiple(requests).then(successCallback,
errorCallback);
```

Execute multiple requests in a transaction:

In this case, `req1`, `req2`, and `req3` will be executed in transaction.

JavaScript

```

var changeSet = [req1, req2, req3];
var requests = [changeSet];
Xrm.WebApi.online.executeMultiple(requests).then(successCallback,
errorCallback);

```

Execute a mix of individual requests and multiple requests in a transaction:

In this case, `req1`, `req2`, and `req3` will be executed in transaction, but `req4` and `req5` will be executed individually.

JavaScript

```

var changeSet = [req1, req2, req3];
var requests = [req4, req5, changeset];
Xrm.WebApi.online.executeMultiple(requests).then(successCallback,
errorCallback);

```

Parameters

Name	Type	Required	Description
requests	Array of objects	Yes	<p>An array of one of the following types:</p> <ul style="list-style-type: none"> objects where each object is an action, function, or CRUD request that you want to execute against the Web API endpoint. Each object exposes a getMetadata method that lets you define the metadata for the action, function, or CRUD request you want to execute. This is the same object that you pass in the <code>execute</code> method. For information about the object, see execute. Change set (an array of objects), where each object in the change set is as defined above. In this case, all the request objects specified in the change set will get executed in a transaction. <p>See request examples earlier in the Syntax section for more information.</p>
successCallback	Function	No	<p>A function to call when operation is executed successfully. An array of response objects are passed to the function where each response object has the following values:</p> <ul style="list-style-type: none"> json: (Optional). Promise. Response body in JSON format.

- **text**: (Optional). Promise. Response body in plaintext format.
- **headers**: Object. Response headers.
- **ok**: Boolean. Indicates whether the request was successful.
- **status**: Number. Numeric value in the response status code. For example: 200
- **statusText**: String. Description of the response status code. For example: OK
- **type**: String. Response type. Values are: the empty string (default), "arraybuffer", "blob", "document", "json", and "text".
- **url**: String. Request URL of the action, function, or CRUD request that was sent to the Web API endpoint.

errorCallback	Function	No	A function to call when the operation fails.
---------------	----------	----	--

Return Value

On success, returns a promise containing an array of objects with the values specified earlier in the description of **successCallback** function.

See also

[Xrm.WebApi](#)

Best practices and guidance for client-side scripting for model-driven apps

Article • 01/28/2023

This list below contains all of the Best practices and guidance for client-side scripting for model-driven apps.

Best Practice	Description
Avoid using window.top	Describes how to avoid script errors and incorrect application behavior associated with using window.top in JavaScript customizations.
Consider disabling NavBar when programmatically opening forms or views	Opening up forms or views with a URL, could lead to slower client performance on high latency networks when the navigation bar (NavBar) is enabled.
Do not use the OData v2.0 endpoint	Describes the requirement to upgrade code to use Web API OData v4.0 endpoint rather than the deprecated OData v2.0 endpoint.
Interact with HTTP and HTTPS resources asynchronously	You should interact with HTTP and HTTPS resources asynchronously when writing JavaScript client extensions for model driven apps.
Remove deactivated or disabled customizations	Deactivated or disabled customizations should be removed from a solution to improve solution management and to decrease the risk of utilizing or managing an outdated component.

See Also

[Apply business logic using client scripting](#)

Customize commands and the ribbon

Article • 10/04/2022

ⓘ Note

This topic is about classic commands.

There is a new way to define commands. See [Modern commanding overview \(preview\)](#).

Microsoft Dataverse displays commands in different ways depending on the table and the client. In most places in the web application, you will see a *command bar* instead of a ribbon. Dynamics 365 for Tablets also uses data defined as ribbons to control what commands are available using a command bar that is optimized for touch.

The command bar provides better performance. The ribbon is still displayed in the web application for certain forms and it is still used for list views in Dynamics 365 for Outlook. Both the command bar and the ribbon use the same underlying XML data to define what commands to display, when the commands are enabled, and what the commands do.

The articles in this section introduce you to key concepts that you must understand, and common tasks you perform when you customize the command bar or the ribbon.

ⓘ Note

Because the underlying XML schema was designed to display commands as ribbons, the term *ribbon* will continue to be used in the documentation.

Troubleshoot ribbon issues

If you are experiencing an issue with a ribbon command bar button, use this [troubleshooting guide](#) to find and solve the problem.

Reference documentation

You can find reference documentation for Ribbon XML elements here: [Ribbon XML reference](#). This documentation is not maintained and includes many elements that are no longer relevant. It provides information about the elements defined in the ribbon

schema files: [Ribbon core schema](#), [Ribbon types schema](#), and [Ribbon WSS schema](#).

There are some remarks within this reference that may be helpful.

Community tool

The SDK describes the process of editing the ribbon by editing the customization.xml file directly. You can also use a community tool, [Ribbon Workbench](#), to visually edit ribbons using the UI.

 **Note**

Microsoft does not provide help or support for community tools. To obtain support or help to use these programs, contact the program publisher.

See also

[Ribbon Core Schema](#)

[Ribbon Types Schema](#)

[Ribbon WSS Schema](#)

[Sample: Export Ribbon Definitions](#)

[Apply business logic using client scripting in model-driven apps](#)

Ribbons available

Article • 01/09/2023

ⓘ Note

This topic is about classic commands.

There is a new way to define commands. See [Modern commanding overview \(preview\)](#).

This article describes where ribbons are defined and modified in model-driven apps.

Ribbon definitions

Model-driven apps contain default `<RibbonDiffXml>` definitions for all ribbons in the application. You can export and view the current XML defining the ribbon for your organization, but you cannot update the XML directly. You customize the ribbon by defining how you want it to be changed. The change definitions that you specify are applied at runtime when the ribbon is displayed in the application. All of your changes will be in the `<CustomAction>` or `<HideCustomAction>` elements. These elements are applied over the default ribbon definitions provided by model-driven apps.

When you write your change definitions, you will frequently need to reference the definitions of the default ribbons. For example, if you want to hide a specific ribbon element, you will need to know the unique ID of that element. If you want to position a new ribbon element within or next to an existing ribbon element, you will need to know the ID values for those elements, as well as the sequence order that will control the relative position of the elements.

Because of this requirement to reference the definitions of existing ribbon elements, it is important to understand the current ribbon definitions in your organization. There are two messages you can use to export XML files representing the current state of your ribbons. These definitions include any customizations that have already been applied to your system so that you can customize any custom ribbons that were previously applied. For more information, see [Export Ribbon Definitions](#).

To help you get started, you can download the default ribbon definitions for model-driven apps from [Export Ribbon Definitions sample](#). The `ExportedRibbonXml` file includes the output files you would have for an organization.

Within the exported ribbon XML files, the applicationRibbon.xml file includes all the ribbons that are not defined for a specific table. These correspond to the **Application Ribbons** solution component. For each table, you will find an *table nameribbon.xml* file. This corresponds to the `RibbonDiffXml` that is included in each table. If you want to edit the ribbon for a specific table, you should locate the ribbon XML file for that table.

Table ribbons

All tables use a common ribbon definition called the *Table Ribbon Template*. The table ribbon template definition is located in the `applicationribbon.xml` file. When you create a custom table, the ribbon you see is the default ribbon defined by the table ribbon template. Each system table has a separate `<RibbonDiffXml>` definition that builds upon the table ribbon template definition.

Within the `applicationribbon.xml` file, you can see the following tabs that apply to all tables:

- `Mscrm.Form.{!EntityLogicalName}.MainTab`

Tab displays the table display name in the label.

- `Mscrm.Form.{!EntityLogicalName}.Related`

Tab has the label **Add**.

- `Mscrm.Form.{!EntityLogicalName}.Developer`

Tab has the label **Customize**.

- `Mscrm.HomepageGrid.{!EntityLogicalName}.MainTab`

Tab displays the plural table display name in the label.

- `Mscrm.HomepageGrid.{!EntityLogicalName}.View`

Tab has the label **View**.

- `Mscrm.HomepageGrid.{!EntityLogicalName}.Related`

Tab has the label **Add**.

- `Mscrm.HomepageGrid.{!EntityLogicalName}.Developer`

Tab has the label **Customize**.

- `Mscrm.SubGrid.{!EntityLogicalName}.ContextualTabs`

When a sub grid in a form or chart has focus, the contextual tab appears with the label **List Tools**.

- `Mscrm.SubGrid.{!EntityLogicalName}.MainTab`

Tab displays the plural table display name.

When you view the ribbon definitions for a specific table, you will see that the name of the table usually replaces the `{!EntityLogicalName}` token. When you see the `{!EntityLogicalName}` token in the ribbon definition for a specific table, that means there is no specific definition for that table and it simply uses the definition from the table ribbon template. When you define ribbons for a specific table, always use the actual table name. Ribbon modifications for a specific table must be defined in the `//ImportExportXml/Entities/Entity/RibbonDiffXml` node.

You can make changes that apply to all tables by defining the changes to the application ribbons substituting the token `{!EntityLogicalName}` in place of a table logical name in your `RibbonDiffXml` node. Changes to application ribbons that are defined for all tables must be defined in the `ImportExportXml/RibbonDiffXml` node. They cannot be defined in the `RibbonDiffXml` node for a specific table.

Grid ribbons

The table grid ribbon is a collection of tabs that have an ID value beginning with `Mscrm.HomepageGrid.<entity logical name>`. For example, the tab with the text "Accounts" on an account table grid is `Mscrm.HomepageGrid.account.MainTab`. All the tabs displayed on the account table grid will have an ID value that begins with `Mscrm.HomepageGrid.account`.

Subgrid ribbons

The table subgrid ribbon is a contextual group with a collection of tabs that have an ID value beginning with `Mscrm.SubGrid.<entity logical name>`. For example, the tab with the text "Accounts" on account table sub grid is `Mscrm.SubGrid.account.MainTab`.

When a list of records for a table is displayed within a sub grid on the form of another table or in a chart, there will be only three controls available directly above or within the subgrid. The behaviors for these controls can be modified by changing the commands that they are associated with.

- **Add:** The default behavior of the command with the  icon depends on whether the records in the subgrid are related to the current record.

If the records are related to the current record, the default behavior is look for existing records. If an existing record cannot be found, or if the user simply wants to create a new record, they can select **Add New**.

If the records are not related to the current record, the default behavior is to add a new record. If the table has a Quick Create form this will be displayed, otherwise a new full form will be shown.

Activities are the exception to this pattern. The add command will always prompt for the type of activity first.

Note

Offline mode in Dynamics 365 does not support many-to-many relationship on custom tables. Due to this, the **Add New** button on a sub grid in Dynamics 365 offline mode will not be displayed.

- **Show List:** The command with the  icon will open the full list where all available commands can be used.

If the subgrid is associated with the current record, the default behavior of this command is to open the associated view.

If the subgrid is not associated with the current record, the default behavior of this command is to open the view in the main list view.

- **Delete:** The  icon is shown on the right side of the row when people hover over the records in the list.

For records with a 1:N relationship or no relationship, the default behavior is to delete the record. The delete may be blocked if it is not allowed due to relationship configurations. Open activities and invoices are common examples of records that may not be deleted due to relationship configurations.

For relationships displaying N:N relationships the default behavior is to remove the relationship joining the records rather than the record itself.

You can change the default behavior by changing the actions associated with the command using `<CommandDefinition>`, but you cannot change the name of the command. For example, you could change the delete action so that it deactivates the record rather than deleting it.

It is not possible to change the icons displayed for these commands. You can hide these commands by using `<HideCustomAction>`.

Form ribbons

ⓘ Note

This feature is not supported on Unified Interface.

Each table can have multiple forms. You can define changes to the form ribbon for all forms for that table by adding your definition at the table level
(`//ImportExportXml/Entities/Entity/RibbonDiffXml`).

Each table form can have a specific ribbon definition. In the exported `customizations.xml` file, you must add your modified `<RibbonDiffXml>` to this location: `//ImportExportXml/Entities/Entity/FormXml/forms/systemform/form/RibbonDiffXml`.

The form ribbon is a collection of tabs that have an ID value beginning with `Mscrm.Form.<entity logical name>`. For example, the tab with the label **Account** on account form is `Mscrm.Form.account.MainTab`. All the tabs displayed on the account form will have an ID value that begins with `Mscrm.Form.account`.

Basic home tab

The basic home tab is displayed on the main application ribbon whenever an alternative tab is not defined because of table context or a display rule that suppresses it for specific pages. For example, this tab is displayed when you view the model-driven apps **Help**. The ID of the basic home tab is `Mscrm.BasicHomeTab`.

Customizing global commandbar

You can customize the global commandbar (`Mscrm.GlobalTab`) by adding the buttons to `Mscrm.GlobalTab`. The out of the box buttons in the global commandbar currently cannot be modified, but new buttons can be added.

When the location of the `CustomAction` is set to

`Location="Mscrm.GlobalTab.New.Controls._children"`, custom button is displayed in the global commandbar at the top of the page.

Note

This feature is supported only on Unified Interface.

Other ribbons

Several other special purpose ribbon tabs and a contextual group are defined by model-driven apps. Each tab is associated with a specific <TabDisplayRule> that controls when they will display. The following table lists these tabs.

Tab	Root Id	Description
Web Resource Edit page tab.	Mscrm.WebResourceEditTab	Displays when editing Web resources within a solution.
Form Editor tab	Mscrm.FormEditorTab	Provides Save, Edit, Select, and View groups of actions for forms.
Form Editor Insert tab	Mscrm.FormEditorInsertTab	Provides buttons to insert Sections, Tabs, and Controls in forms.
Dashboard Homepage tab	Mscrm.DashboardTab	Displays in the Workplace area.
Visualization Tools Contextual Group	Mscrm.VisualizationTools	Displays when the New Chart button is clicked on the Charts tab displayed in the table grid ribbon.
AptbookTab Homepage tab	Mscrm.AptbookTab	Displays when viewing the Service Calendar in the Service area.
Advanced Find tab	Mscrm.AdvancedFind	Displays in the Advanced Find window.
Dashboard Editor tab	Mscrm.DashboardEditorTab	Displays when editing a dashboard.
Documents tab	Mscrm.DocumentsTab	Displays if SharePoint integration has been enabled for the organization.
Chart Editor tab	Mscrm.VisualizationDesignerTab	Displays when editing a chart from the solutions window.
Search Tools Contextual Group	Mscrm.ArticleSearch	Displays when viewing the KBarticle table.

Ribbons for custom pages

You can display custom pages in the application navigation using the SiteMap. These pages will always display the [Basic home tab](#) (`Mscrm.BasicHomeTab`).

It is not possible to use a `<PageRule>` to enable or display custom ribbon components on custom pages.

See also

[Customize the ribbon](#)

[Command bar or ribbon presentation](#)

Command bar or ribbon presentation

Article • 11/30/2022

ⓘ Note

This topic is about classic commands.

There is a new way to define commands. See [Modern commanding overview \(preview\)](#).

Data defining commands in Microsoft Dataverse can be presented in several different ways depending on the client and differences in how some table are treated. You need to take these factors into consideration as you change ribbon commands or define new ones.

Different presentations of commands

There are three different ways that command data can be displayed.

Updated user experience

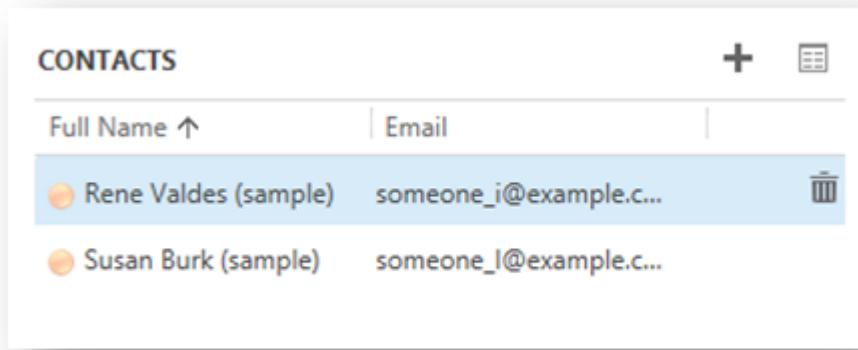
This is the presentation of the command bar throughout the application and for forms for table that have the updated user experience.



In this experience, only few commands are displayed and any remaining commands are available in a flyout menu.

Enable rules will hide commands that should not be used.

Subgrids have a limited number of controls. Only controls to allow adding records, deleting records, or opening a view of the grid are available. But these commands are still defined by ribbon data and can be customized.

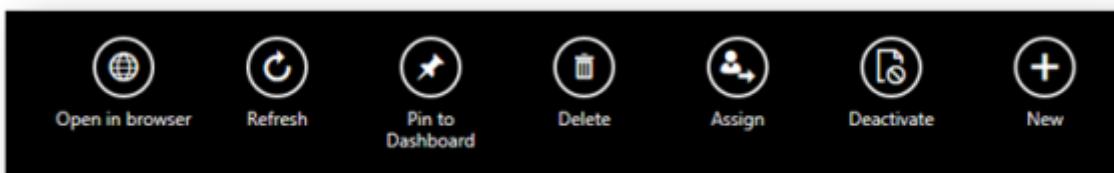


To perform more actions on the list of records displayed in a subgrid, select the option to open a view of the grid.

For more information about the behavior of subgrid controls and how they can be customized, see [Subgrid ribbons](#).

Dynamics 365 for tablets

Dynamics 365 for tablets presents commands in a manner optimized for touch experiences. Commands appear in the command bar at the bottom right of the screen in order from right to left.

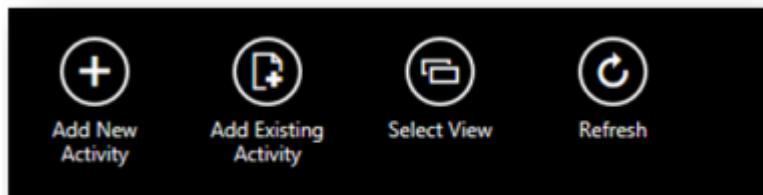


Note

Icons configured for commands will not display and labels that are too long will be truncated.

Dynamics 365 for tablets does not support adding dynamic elements to `<FlyoutAnchor>` or `<SplitButton>` elements at runtime.

Subgrid commands are displayed when people tap and press the subgrid control. These commands are shown on the bottom left of the screen in order from left to right.



Command data

Despite these different presentations, the data that defines the commands for tables is consistent regardless of how the commands are presented. It contains definitions for tabs and groups with scaling, but the visible parts of these containers for controls are only displayed in the classic user interface.

In both the updated user experience and Dynamics 365 for tablets, tabs and groups still act as containers for controls, but there is no visual indication of these containers and scaling is not applied.

Filtering commands based on presentation and client

i Important

Including some kind of rule to filter the display of your commands is necessary unless you want the command to be available for all clients and presentations.

With this release, there is a new element that can be used in display and enable rules to adapt the commands you display with the presentation.

`<CommandClientTypeRule>` contains a `Type` parameter that will be evaluated based on the presentation. The following valid options correspond to the presentation:

- `Refresh`: Updated user experience
- `Legacy`: Classic user experience
- `Modern`: Dynamics 365 for tablets

Use this element as you define commands to control whether they display in the different presentations.

There is also a pre-existing `<CrmClientTypeRule>` element, but the `Type` parameter for element can only differentiate between `Web` and `Outlook` clients. This rule will evaluate the Dynamics 365 for tablets client as the web client.

See also

[Customize commands and the ribbon](#)

[Ribbons available](#)

[Export ribbon definitions](#)

[Troubleshoot ribbon issues](#)

Export, prepare to edit, and import the ribbon

Article • 11/30/2022

ⓘ Note

This topic is about classic commands.

There is a new way to define commands. See [Modern commanding overview \(preview\)](#).

To edit the ribbon, you must perform the following steps:

1. [Export the ribbon](#)
2. [Prepare to edit the XML](#)
3. Edit the `<RibbonDiffXml>`
4. [Import the ribbon](#)

Export the ribbon

You export the ribbon by including it in a solution and then exporting the solution. You can export all the customizations, but that can represent a large amount of data. We recommend that you use an existing unmanaged solution or create a new solution.

Create a new solution

1. Go to **Settings > Customizations**.
2. Go to **Settings > Solutions**.
3. Select **New**.
4. Type a meaningful **Display Name**, **Name** and enter a **Publisher** and type a **Version** number.

ⓘ Note

You can usually use the default publisher for the organization.

5. Select the **Save** icon.
6. If you want to edit the ribbon for specific tables:
 - a. Select **Add Existing** and then select **Entity**.
 - b. Select the tables you want to include in the solution and then select **OK**.

 **Note**

For the purpose of editing table ribbons, you do not have to include required components. If you intend to export this solution and apply it to another system, you should include required components.

7. If you want to edit global ribbons or add a custom group to all tables, select **Add Existing** and then select **Application Ribbons**.
8. Select **Save and Close**.

Use an existing solution

1. Go to **Settings > Customizations**.
2. Go to **Settings > Solutions**.
3. Double-click a solution to open it.
4. If you want to edit the ribbon for specific tables:
 - a. Select **Add Existing** and then click **Entity**.
 - b. Select the tables you want to include in the solution and then select **OK**.

 **Note**

For the purpose of editing table ribbons, you do not have to include required components. If you intend to export this solution and apply it to another system, you should include required components.

5. If you want to edit global ribbons, such as to add custom button to all tables: select **Add Existing** and then select **Application Ribbons**.
6. Select **Save and Close**.

Export the ribbon

1. Go to **Settings > Customizations**.
2. Go to **Settings > Solutions**.
3. Select the solution you want and then select **Export**.
4. If you have made recent changes that have not yet been published, select **Publish All Customizations**. Otherwise, select **Next**.
5. With the **Unmanaged** option selected, select **Export**.
6. Select **Save** in the **File Download** dialog box and then select **Open Folder** in the **Download complete** dialog box.
7. Right-click the compressed .zip file that you downloaded and select **Extract All...**.
8. Select a location to extract the files and then select **Extract**.
9. The customizations.xml file is the file that you will edit.

Prepare to edit the XML

or a better experience, edit the customizations.xml file with an application that can use schema validation to provide IntelliSense support. For more information, see [Edit the customizations file with schema validation](#).

Import the ribbon

1. After you have edited the customization.xml file, from Visual Studio or Visual Web Developer 2010 Express, right-click the customization.xml tab and select **Open Containing Folder**.
2. Select all of the files or folders that were included when you extracted the solution. Right-click the selected files, select **Send to**, and then select **Compressed (zipped) folder**.

Note

This creates a compressed .zip file in the same folder. The name of the file may vary, but it will be the same as one of the other files in the folder - except with a .zip file name extension.

3. Go to **Settings > Customizations**.
4. Go to **Settings > Solutions**.
5. Select **Import**.
6. Select **Browse** and locate the compressed .zip file that you created in step 2 of this procedure.
7. Select **Next** and then select **Import**.
8. After the import has finished, you will see the message indicating that the import completed successfully. Select **Close**.
9. After you have successfully imported your solution, you must publish customizations before you can see the changes. In the solutions list, select **Publish All Customizations**.

Dealing with errors on import

1. If you receive a notification that there were errors that caused the import to fail, select **Export Log**.
2. Save the export log file. Select the file and right-click it. Click **Open With** and then select **Microsoft Office Excel**.
3. Select the **Components** worksheet and note any messages in the **ErrorText** column.

💡 Tip

The most common type of failure is an error when referencing a dependent element in the RibbonDiffXml. Perhaps you forgot to include a LocLabel that was referenced somewhere. Perhaps there is an extra blank character included at the end of an XML parameter referencing another element. All references must match exactly.

4. After you have corrected the error, complete the steps to import the ribbon again.

Troubleshoot ribbon issues

If you are experiencing an issue with a ribbon command bar button, use this [troubleshooting guide](#) to find and solve the problem.

See also

[Customize the ribbon](#)

[Export ribbon definitions](#)

[Use localized labels with ribbons](#)

Export ribbon definitions

Article • 11/30/2022

ⓘ Note

This topic is about classic commands.

There is a new way to define commands. See [Modern commanding overview \(preview\)](#).

To effectively define changes to the default RibbonXml, you must be able to reference the RibbonXml data that defines those ribbons.

Access the ribbon definitions for your organization

If the Ribbon for your organization has been modified, you should export the current definitions if you intend to work with the customized ribbon elements. To do this, use the [Export ribbon xml](#) sample.

Access the default ribbon data

The default ribbon definitions for model-driven apps can be downloaded from [Microsoft Downloads: ExportedRibbonXml.zip](#).

The applicationRibbon.xml file contains the definition of the core application ribbons.

The remaining files contain the definitions used by tables that have ribbon definitions that differ from the table template. Each file is named according to the name of the table: logical table name + Ribbon.xml.

These files represent the output of two messages using the [Sample: Export ribbon definitions](#):

[RetrieveApplicationRibbonRequest](#)

This message retrieves the core application ribbons including the table template.

[RetrieveEntityRibbonRequest](#)

This message retrieves the ribbon definition used for a specific table.

Decompress the ribbon data

The ribbon data is exported as a compressed file. To decompress the file into XML, you have to use the [System.IO.Packaging.ZipPackage](#) class. The following example is a helper method used in the SDK sample to decompress the file.

C#

```
/// <summary>
/// A helper method that decompresses the Ribbon data returned
/// </summary>
/// <param name="data">The compressed ribbon data</param>
/// <returns></returns>
public byte[] unzipRibbon(byte[] data)
{
    System.IO.Packaging.ZipPackage package = null;
    MemoryStream memStream = null;

    memStream = new MemoryStream();
    memStream.Write(data, 0, data.Length);
    package = (ZipPackage)ZipPackage.Open(memStream, FileMode.Open);

    ZipPackagePart part = (ZipPackagePart)package.GetPart(new
    Uri("/RibbonXml.xml", UriKind.Relative));
    using (Stream strm = part.GetStream())
    {
        long len = strm.Length;
        byte[] buff = new byte[len];
        strm.Read(buff, 0, (int)len);
        return buff;
    }
}
```

Retrieve the application ribbon data

The application ribbon can be retrieved using the [RetrieveApplicationRibbonRequest](#) as shown in the following sample.

C#

```
//Retrieve the Application Ribbon
var appribReq = new RetrieveApplicationRibbonRequest();
var appribResp =
(RetrieveApplicationRibbonResponse)service.Execute(appribReq);

System.String applicationRibbonPath = Path.GetFullPath(exportFolder +
"\\""\applicationRibbon.xml");
```

```
File.WriteAllBytes(applicationRibbonPath,
unzipRibbon(appribResp.CompressedApplicationRibbonXml));
```

Retrieve table ribbons

To retrieve the ribbon definition for tables, you can just include the name of the table as a parameter to the [RetrieveEntityRibbonRequest](#).

To retrieve the ribbon definitions for all tables that support the ribbon, you need a list of those system tables that have ribbon definitions that vary from the table ribbon template. The following sample shows an array of all the system tables that have ribbon definitions.

C#

```
//This array contains all of the system tables that use the ribbon.
public System.String[] entitiesWithRibbons = {"account",
"activitymimeattachment",
"activitypointer",
"appointment",
"bulkoperation",
"calendar",
"campaign",
"campaignactivity",
"campaignresponse",
"competitor",
"connection",
"contact",
"contract",
"contractdetail",
"convertrule",
"convertruleitem",
"customeraddress",
"discount",
"discounttype",
"email",
"emailserverprofile",
"entitlement",
"entitlementchannel",
"entitlementtemplate",
"entitlementtemplatechannel",
"fax",
"goal",
"goalrollupquery",
"importfile",
"incident",
"invoice",
"invoicedetail",
"kbarticle",
"kbarticlecomment",
```

```
"lead",
"letter",
"list",
"listmember",
"mailbox",
"metric",
"opportunity",
"opportunityproduct",
"partnerapplication",
"phonecall",
"postfollow",
"pricellevel",
"product",
"productpricellevel",
"queue",
"queueitem",
"quote",
"quotedetail",
"recurringappointmentmaster",
"report",
"rollupfield",
"routingrule",
"routingruleitem",
"salesliterature",
"salesliteratureitem",
"salesorder",
"salesorderdetail",
"service",
"serviceappointment",
"sharepointdocument",
"sharepointdocumentlocation",
"sharepointsite",
"site",
"sla",
"slaitem",
"socialactivity",
"socialprofile",
"systemuser",
"task",
"team",
"teamtemplate",
"territory",
"uom",
"uomschedule",
"userquery"};
```

The following sample shows how to retrieve the ribbon definitions for a set of tables.

C#

```
//Retrieve system table Ribbons
var entRibReq = new RetrieveEntityRibbonRequest() { RibbonLocationFilter =
RibbonLocationFilters.All };
```

```

foreach (System.String entityName in entitiesWithRibbons)
{
    entRibReq.EntityName = entityName;
    var entRibResp = (RetrieveEntityRibbonResponse)service.Execute(entRibReq);

    System.String entity.RibbonPath = Path.GetFullPath(exportFolder + "\\\" +
entityName + "Ribbon.xml");
    File.WriteAllBytes(entity.RibbonPath,
unzipRibbon(entRibResp.CompressedEntityXml));
    //Write the path where the file has been saved.
    Console.WriteLine(entity.RibbonPath);
}

```

Any custom tables also support ribbon customizations. To get a list of custom tables, use the [RetrieveAllEntitiesRequest](#) and retrieve the names of custom tables. The following sample shows how to retrieve ribbon definitions for all custom tables.

C#

```

//Check for custom tables
var raer = new RetrieveAllEntitiesRequest() { EntityFilters =
EntityFilters.Entity };
var resp = (RetrieveAllEntitiesResponse)service.Execute(raer);
foreach (EntityMetadata em in resp.EntityMetadata)
{
    if (em.IsCustomEntity == true && em.IsIntersect == false)
    {
        entRibReq.EntityName = em.LogicalName;
        var entRibResp =
(RetrieveEntityRibbonResponse)service.Execute(entRibReq);
        System.String entity.RibbonPath = Path.GetFullPath(exportFolder + "\\\" +
em.LogicalName + "Ribbon.xml");
        File.WriteAllBytes(entity.RibbonPath,
unzipRibbon(entRibResp.CompressedEntityXml));
        //Write the path where the file has been saved.
        Console.WriteLine(entity.RibbonPath);
    }
}

```

Troubleshoot ribbon issues

If you are experiencing an issue with a ribbon command bar button, use this [troubleshooting guide](#) to find and solve the problem.

See also

Customize the ribbon

Command bar or ribbon presentation

Export, prepare to edit, and import the ribbon

Use localized labels with ribbons

Article • 11/30/2022

ⓘ Note

This topic is about classic commands.

There is a new way to define commands. See [Modern commanding overview \(preview\)](#).

Although Ribbon elements that display text allow for direct entry of text, it is a best practice to use localized labels to define text displayed in the ribbon. This enables multi-language capabilities and better management of shared text.

Using localized labels

The `<RibbonDiffXml>` element includes the `<LocLabels>` element. As shown in the following example, this is where you can specify which text to display in the ribbon labels and tooltips using the `<Titles>` element.

XML

```
<LocLabels>
  <LocLabel Id="MyISV.account.SendToOtherSystem.LabelText">
    <Titles>
      <Title languagecode="1033"
        description="Send to Other System" />
    </Titles>
  </LocLabel>
  <LocLabel Id="MyISV.account.SendToOtherSystem.ToolTip">
    <Titles>
      <Title languagecode="1033"
        description="Sends this Record to another system" />
    </Titles>
  </LocLabel>
</LocLabels>
```

Within the definition of a ribbon element that displays text, the following example show how the localized label can be referenced using the `$LocLabels:` directive.

XML

```
ToolTipTitle="$LocLabels:MyISV.account.SendToOtherSystem.LabelText"  
ToolTipDescription="$LocLabels:MyISV.account.SendToOtherSystem.ToolTip"
```

Force a line break in a ribbon control label

If you have a ribbon control label that is very long, the text will wrap to fit the available space. You can specify where you want to include a line break by using the following characters: `​​`.

If the label text is very long without a space for the text to wrap, the width of the control expands to allow for the entire label to be displayed.

See also

[Customize commands and the ribbon](#)

[Export, prepare to edit, and import the Ribbon](#)

[Use localized labels with Ribbons](#)

[Define Ribbon Commands](#)

Define ribbon commands

Article • 11/30/2022

ⓘ Note

This topic is about classic commands.

There is a new way to define commands. See [Modern commanding overview \(preview\)](#).

A *Ribbon* command creates a reusable definition that can be referenced by ribbon control elements.

Ribbon command elements

The `<CommandDefinition>` element defines a command in the ribbon. The `Id` attribute specifies a unique identifier for the command that can be referenced by ribbon control elements by using the `Command` parameter.

A ribbon command defines three things:

- **Enable Rules:** Specifies when a specific ribbon control is enabled.
- **Display Rules:** Specifies when a specific ribbon element is visible.
- **Actions:** Specifies what code executes when a ribbon control is used.

ⓘ Important

All command definitions are downloaded to a user's computer so that they can be evaluated at run time. This means that a user without the privileges to see a particular control in the ribbon can use the browser **View Source** command, review the code, and determine that a control exists that isn't displayed to them.

See also

[Customize commands and the ribbon](#)

[Use localized labels with Ribbons](#)

[Define Ribbon enable rules Troubleshoot ribbon issues](#)

Define ribbon enable rules

Article • 11/30/2022

ⓘ Note

This topic is about classic commands.

There is a new way to define commands. See [Modern commanding overview \(preview\)](#).

When configuring ribbon elements, you can define specific rules to control when the ribbon elements are enabled. The `<EnableRule>` element is used as follows:

- Use the `/RuleDefinitions/EnableRules/EnableRule` element to define rules controlling when the ribbon element should be enabled.
- Use the `/CommandDefinitions/CommandDefinition/EnableRules/EnableRule` element to associate specific enable rules to a command definition.

What does enabled mean?

With the command bar, commands that are disabled are hidden. With the ribbon, commands that are disabled are visible but do not respond to events.

Control when ribbon elements are enabled

Enable rules are intended to be reused. By defining them with rule definitions, you can use the same enable rule for many command definitions. When more than one enable rule is defined for a command definition, all of the enable rules must evaluate as true for the ribbon element to be enabled.

All Enable rules provide an optional parameter to specify whether the default value of the rule is true or false and an optional `InvertResult` parameter to allow for returning a negative result when the item being tested returns true.

The `/RuleDefinitions/EnableRules/EnableRule` element supports the following types of rules:

Command Client Type Rule

Uses the `<CommandClientTypeRule>` element. Specifies a rule that detects the type of presentation being used.

The `Type` values correspond to the following:

Value	Presentation
Modern	The command bar is presented using Dynamics 365 for tablets.
Refresh	The command bar is presented using the updated user interface.
Legacy	The ribbon is presented in forms for tables that were not updated or in a list view in Dynamics 365 for Outlook.

Crm Client Type Rule

Uses the `<CrmClientTypeRule>` element to allow definition of rules depending on the type of client used. Type options are as follows:

- Web
- Outlook

Crm Offline Access State Rule

Uses the `<CrmOfflineAccessStateRule>` element. Use this criteria to enable a ribbon element based on whether Dynamics 365 for Microsoft Office Outlook with Offline Access is currently offline.

Crm Outlook Client Type Rule

Uses the `<CrmOutlookClientTypeRule>` element. Use this rule if you want to only display a button for a specific type of Dynamics 365 for Outlook. Type options are as follows:

- CrmForOutlook
- CrmForOutlookOfflineAccess

Custom Rule

Uses the `<customRule>` element. Use this kind of rule to call a function in a [JavaScript web resource](#) that returns a Promise (Unified Interface) or boolean (Unified Interface and web client).

JavaScript

```
function EnableRule()
{
    const value = Xrm.Page.getAttribute("column1").getValue();
    return value === "Active";
}
```

ⓘ Note

Custom rules that do not return a value quickly can affect the performance of the ribbon. If you have to perform logic that might take some time to complete (for example, a network request), use the following strategy to make your custom rule asynchronous.

Unified Interface rules support returning a Promise rather than boolean for asynchronous rule evaluation. If the promise does not resolve within 10 seconds, the rule will resolve with a false value.

ⓘ Note

Promises-based rules will only work on Unified Interface, so they cannot be used if classic Web Client is still being used.

JavaScript

```
// Old synchronous style
/*
function EnableRule() {
    const request = new XMLHttpRequest();
    request.open('GET', '/bar/foo', false);
    request.send(null);
    return request.status === 200 && request.responseText === "true";
}
*/


// New asynchronous style
function EnableRule() {
    const request = new XMLHttpRequest();
    request.open('GET', '/bar/foo');

    return new Promise(function(resolve, reject) {
        request.onload = function (e) {
            if (request.readyState === 4) {
                if (request.status === 200) {
                    resolve(request.responseText === "true");
                }
            }
        }
    });
}
```

```

        } else {
            reject(request.statusText);
        }
    };
    request.onerror = function (e) {
        reject(request.statusText);
    };

    request.send(null);
});
}

```

Entity Rule

Uses the `<EntityRule>` element. `EntityRule` allow for evaluation of the current table. This is useful when you define custom actions that apply to the table template instead of for specific tables. For example, you may want to add a ribbon element to all tables except for several specific tables. It is easier to define the custom action for the table template that applies to all tables and then use an `EntityRule` to filter out those that should be excluded.

The `EntityRule` also includes an optional context parameter to specify whether the table is being displayed in the form or a list (HomePageGrid). The optional `AppliesTo` parameter can be set to `PrimaryEntity` or `SelectedEntity` to distinguish whether the table is being displayed in a subgrid.

Form State Rule

Uses the `<FormStateRule>` element. Use the `FormState` rule to determine the current type of form that is displaying a record. State options are as follows:

- `Create`
- `Existing`
- `ReadOnly`
- `Disabled`
- `BulkEdit`

Or Rule

Uses the `<OrRule>` element. The `OrRule` lets you override the default AND comparison for multiple enable rule types. Use the `OrRule` element to define several possible valid combinations to check.

Outlook Item Tracking Rule

Uses the `<OutlookItemTrackingRule>` element. Use the `TrackedInCrm` parameter for this element to determine whether the record is being tracked in Power Apps.

Outlook Version Rule

Uses the `<OutlookVersionRule>` element. Use this to enable a ribbon element for a specific version of Office Outlook as follows:

- 2003
- 2007
- 2010

Page Rule

Uses the `<PageRule>` element. This type of rule checks the URL of the page being displayed. It returns true if the `Address` matches.

Record Privilege Rule

Uses the `<RecordPrivilegeRule>` element. Use this rule to determine whether the current user has privileges on a specific record. These privileges differ from a table privilege because they can include privileges gained by another user sharing the record with the current user.

Selection Count Rule

Uses the `<SelectionCountRule>` element. Use this kind of rule with a ribbon displayed for a list to enable a button when specific maximum and minimum numbers of records in the grid are selected. For example, if your button merges records, you should make sure at least two records are selected before enabling the ribbon control.

Value Rule

Uses the `<ValueRule>` element. Use this rule to check the value of a specific column in the record being displayed in the form. You must specify the `Field` and the `Value` to check.

 **Note**

On a form, a `ValueRule` requires the specified column to be part of the form for it to work. On a grid or subgrid, the column must be one of the grid columns.

Show On Quick Action Rule

Uses the `<EnableRule>` element. Use this rule to make the command appear only as quick action.

XML

```
<CommandDefinition Id="new.contact.Command.Call">
  <EnableRules>
    <EnableRule Id="Mscrm.SelectionCountExactlyOne" />
    <EnableRule Id="Mscrm.ShowOnQuickAction" />
  </EnableRules>
  <DisplayRules />
  <Actions>
    <JavaScriptFunction FunctionName=" simplealert" />
  </Actions>
</CommandDefinition>
```

Show On Grid and Quick Action rule

Uses the `<EnableRule>` element. Use this rule to make the command appear on the homepage grid and quick action.

XML

```
<CommandDefinition Id="new.contact.Command.Call">
  <EnableRules>
    <EnableRule Id="Mscrm.SelectionCountExactlyOne" />
    <EnableRule Id="Mscrm.ShowOnGridAndQuickAction" />
  </EnableRules>
  <DisplayRules />
  <Actions>
    <JavaScriptFunction FunctionName=" simplealert" />
  </Actions>
</CommandDefinition>
```

Show On Grid Rule

Uses the `<EnableRule>` element. Use this rule to make the quick action command appear on the homepage grid only. In other words, you can use this command to hide an existing quick action.

XML

```
<CommandDefinition Id="new.contact.Command.Call">
  <EnableRules>
    <EnableRule Id="Mscrm.SelectionCountExactlyOne" />
    <EnableRule Id="Mscrm.ShowOnGrid" />
  </EnableRules>
  <DisplayRules />
  <Actions>
    <JavaScriptFunction FunctionName=" simplealert" />
  </Actions>
</CommandDefinition>
```

See also

[Customize commands and the ribbon](#)

[Define ribbon commands](#)

[Define ribbon display rules](#)

Define ribbon display rules

Article • 11/30/2022

ⓘ Note

This topic is about classic commands.

There is a new way to define commands. See [Modern commanding overview \(preview\)](#).

When configuring ribbon elements, you can define specific rules to control when the ribbon elements will display.

- Use the `/RuleDefinitions/DisplayRules/<DisplayRule>` element to define rules controlling when the ribbon element should be displayed.
- Use the `/CommandDefinitions/CommandDefinition/DisplayRules/<DisplayRule>` element to associate specific display rules to a command definition.

Control when ribbon elements are displayed

By defining display rules in rule definitions, you can use the same display rule for many command definitions. When more than one display rule is defined for a command definition, all of the display rules must evaluate as true for the ribbon element to be displayed.

All display rules provide an optional parameter to specify whether the default value of the rule is true or false and an optional `InvertResult` parameter to enable returning a negative result when the item being tested returns true.

The `/RuleDefinitions/DisplayRules/DisplayRule` element supports the following types of rules:

`<CommandClientTypeRule>`

Specifies a rule that detects the type of presentation being used.

The `Type` values correspond to the following:

Value	Presentation
-------	--------------

Value	Presentation
Modern	The command bar is presented using Dynamics 365 for tablets.
Refresh	The command bar is presented using the updated user interface.
Legacy	The ribbon is presented in forms for tables that were not updated or in a list view in Dynamics 365 for Outlook.

`<CrmClientTypeRule>`

Allows definition of rules depending on the type of client used. `Type` options are as follows:

- Web
- Outlook

`<CrmOfflineAccessStateRule>`

Use this criteria to display a ribbon element based on whether Dynamics 365 for Microsoft Office Outlook with Offline Access is currently offline.

`<CrmOutlookClientTypeRule>`

Use this rule if you want to display a button for the specific type of Dynamics 365 for Outlook. `Type` options are as follows:

- CrmForOutlook
- CrmForOutlookOfflineAccess

`<CrmOutlookClientVersionRule>`

Detects the version of Microsoft Dynamics 365 for Microsoft Office Outlook.

Valid values:

- 2003
- 2007
- 2010

`<EntityPrivilegeRule>`

Use this kind of rule to display ribbon elements when a user has specific privileges for a table. You must specify the privilege depth and the specific privilege you want to check.

`<EntityPropertyRule>`

Allows definition of rules depending on the Boolean values of specific table properties. `PropertyName` options are as follows:

- `DuplicateDetectionEnabled`
- `GridFiltersEnabled`
- `HasStateCode`
- `IsConnectionsEnabled`
- `MailMergeEnabled`
- `WorksWithQueue`
- `HasActivities`
- `IsActivity`
- `HasNotes`

`<EntityRule>`

This rule allow for evaluation of the current table. This is useful when you define custom actions that apply to the table template instead of for specific tables. For example, you may want to add a ribbon element to all tables except for some specific tables. It is easier to define the custom action for the table template that applies to all tables and then use an `EntityRule` to filter out those that should be excluded.

The `EntityRule` also includes an optional context parameter to specify whether the table is being displayed in the form or a list (HomePageGrid). The optional `AppliesTo` parameter can be set to `PrimaryEntity` or `SelectedEntity` to distinguish whether the table is being displayed in a subgrid.

`<FormEntityContextRule>`

Specifies a rule that can detect whether a form ribbon is displayed in the context of a specific entity.

`<FormStateRule>`

Use the form state rule to determine the current type of form that is displaying a record. `State` options are as follows:

- `Create`

- Existing
- ReadOnly
- Disabled
- BulkEdit

`<FormTypeRule>`

Specifies a rule that detects the type of Microsoft Dynamics 365 form.

The `Type` values correspond to the following:

Value	Presentation
<code>Main</code>	A form displayed in the application.
<code>Preview</code>	The table preview form displayed as an expanding element in the grid.
<code>AppointmentBook</code>	Used with the appointment, equipment, serviceappointment, and systemuser tables for the Service Scheduling user interface.
<code>Dashboard</code>	The form defines a dashboard.
<code>Quick</code>	A quick view form.
<code>QuickCreate</code>	A quick create form.

`<HideForTabletExperienceRule>`

Specifies a rule that will return false when the web application is viewed in a mobile browser on a tablet device.

`<MiscellaneousPrivilegeRule>`

Use this kind of rule to check for privileges that do not apply to a specific table, such as ExportToExcel, MailMerge, or GoOffline.

`<OrganizationSettingRule>`

Use this to display a ribbon element if specific organization settings are enabled. Setting options are as follows:

- IsSharepointEnabled
- IsSOPIIntegrationEnabled
- IsFiscalCalendarDefined

`<OrRule>` This rule lets you override the default AND comparison for multiple display rule types. Use the `OrRule` element to define several possible valid combinations to check.

`<OutlookRenderTypeRule>`

Use this to display a ribbon element if the ribbon is being displayed in Outlook in a specific way. `Type` options are as follows:

- Web
- Outlook

`<OutlookVersionRule>`

Use this to display a ribbon element for a specific version of Outlook. `Version` options are as follows:

- 2003
- 2007
- 2010

`<PageRule>`

This type of rule checks the URL of the page being displayed. It returns true if the address matches.

`<RelationshipTypeRule>` This type of rule is applied to records selected in a grid. It lets you determine the type of relationship, as follows:

- OneToMany
- ManyToMany
- NoRelationship

`<SkuRule>`

Use this kind of rule to display a ribbon element for a specific SKU version of Microsoft Dataverse, as follows:

- OnPremise
- Online
- Spla

<ValueRule>

Use this rule to check the value of a specific column in the record being displayed in the form.

① Note

For commands defined for subgrid for forms using the updated user experience, value rules cannot be used within display rules. Use this element within an <EnableRule> to hide an element.

See also

[Customize commands and the ribbon](#)

[Define ribbon enable rules](#)

[Define ribbon actions](#)

Define ribbon actions

Article • 11/30/2022

ⓘ Note

This topic is about classic commands.

There is a new way to define commands. See [Modern commanding overview \(preview\)](#).

Define the actions to be performed by a command bar or ribbon control in a `<CommandDefinition>` element together with rules that control whether the control is enabled or visible in the ribbon.

A Ribbon control can perform two types actions and may include multiple actions:

- **JavaScript Functions:** A `<JavaScriptFunction>` element references a function defined in a [JavaScript web resource](#).
- **Open a URL:** The ribbon opens a URL using the value from an Address attribute in the `<Url>` element. Additional parameters can pass information about how what querystring parameters are passed and the mode in which the window opens.

You have several options to pass parameters to a URL using the ribbon. More information: [Passing Parameters to a URL using the Ribbon](#)

Passing parameters to ribbon actions

Use the following elements to define data to pass to your custom action:

`<BoolParameter>`

Specifies a value with a Boolean data type that may be passed as a parameter.

`<CrmParameter>`

Represents data retrieved from the Microsoft Dynamics 365 application that may be passed as a parameter. More information: [Pass data from a page as a parameter to Ribbon Actions](#)

`<DecimalParameter>`

Specifies a value with a decimal data type that may be passed as a parameter.

`<IntParameter>`

Specifies a value with an integer data type that may be passed as a parameter.

`<StringParameter>`

Specifies a value with a string data type that may be passed as a parameter.

When parameters are passed to a `<Url>` element they are passed as a query string. Therefore, they must include a name value to represent the "key" in the query string key/value pair.

Parameters passed to a `<JavaScriptFunction>` do not require a name but they must be included in the order expected by the function and be of the correct data type.

See also

[Customize commands and the ribbon](#)

[Define Ribbon display rules](#)

[Pass data from a page as a parameter to Ribbon actions](#)

Override the default open behavior of data rows in an entity-bound grid

Article • 12/10/2022

ⓘ Note

This topic is about classic commands.

There is a new way to define commands. See [Modern commanding overview \(preview\)](#).

By default, performing any of the following actions in a data row in an entity-bound grid opens the table record:

- Double-clicking the data row, or selecting the primary column link in the row.
- Selecting a data row, and then pressing the **Enter** key.
- On a touch-enabled device, selecting a data row.

There might be situations where you don't want the table record to open (which is the default behavior), but want a custom action to be performed such as opening a URL using JavaScript functions. You can override the default behavior and define your own custom behavior by creating a command definition for a table with

`Mscrm.OpenRecordItem` as the value of the ID parameter `CommandDefinition`, and defining a custom action on the **Actions** tab. The application looks for the `Mscrm.OpenRecordItem` command ID for a table when you try to open a record from the entity-bound grid and—if one is present—will execute the custom action instead of performing the default behavior of opening the table record.

ⓘ Note

- This feature is supported only for Unified Interface.
- You can also use Ribbon Workbench, a community tool, to visually edit ribbons by using the UI. Note that tools created by the community aren't supported by Microsoft. If you have questions or issues with community tools, contact the publisher of the tool.

To specify a custom action when a table record is selected, you'll do the following:

1. Create a web resource to perform the action.

2. Create a custom button on the form by editing the customization.xml file
3. Import the customization.xml file

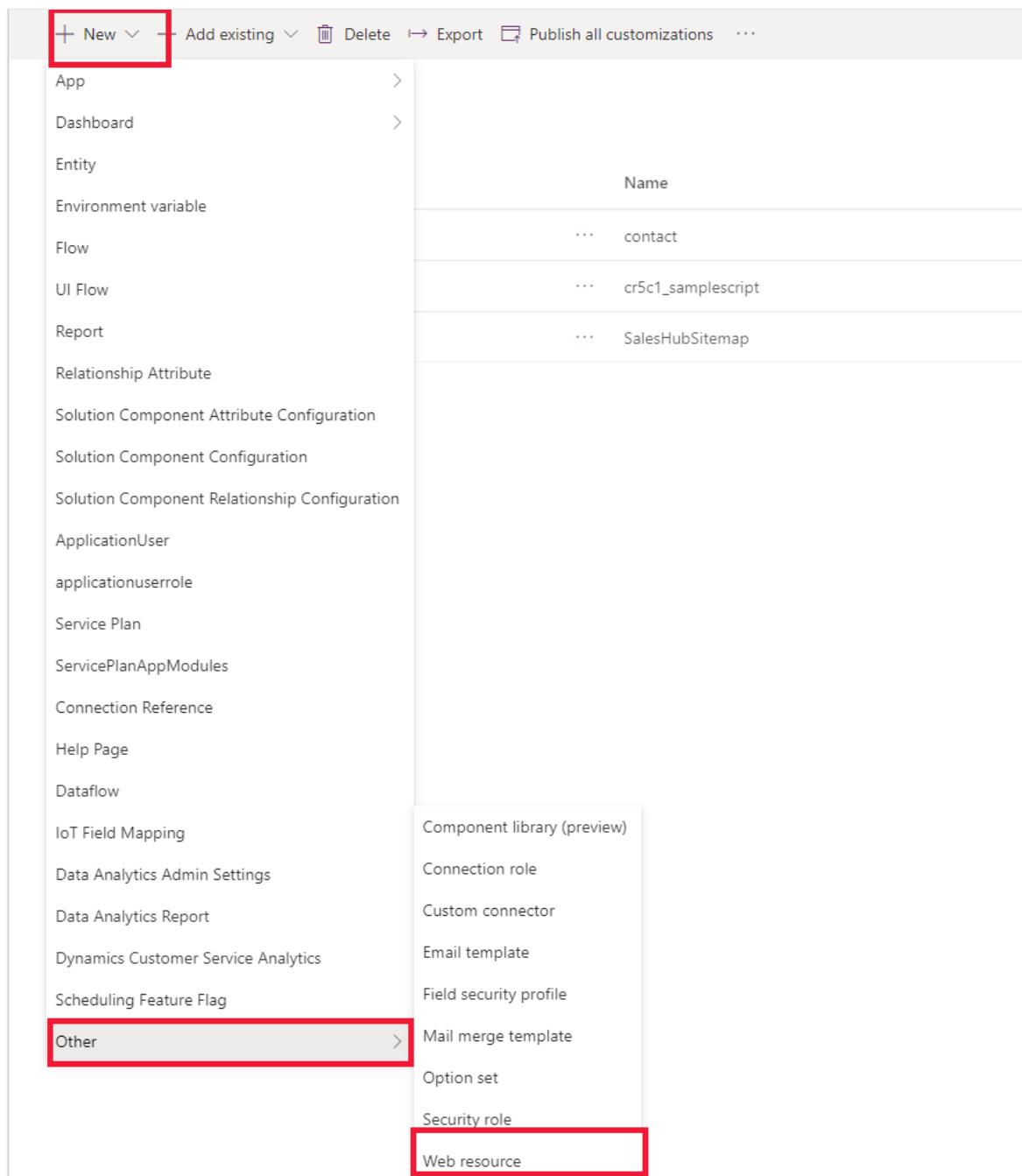
Step 1: Create a web resource

Create a web resource to change the default behavior. In the following example, if you want to open a URL instead of displaying the record, you create a JavaScript web resource to perform that action.

1. Sign in to [Power Apps](#) and select **Solutions** from the left pane.
2. Select **New solution**, and then complete the required columns for the solution.

Column	Description
Display Name	The name shown in the list of solutions. You can change this later.
Name	The unique name of the solution. This is generated by using the value you enter in the Display Name column. You can edit this before you save the solution, but after you save the solution, you can't change it.
Publisher	You can select the default publisher or create a new publisher. We recommend that you create a publisher for your organization to use consistently across the environments where you'll use the solution.
Version	Enter a number for the version of your solution. This is only important if you export your solution. The version number will be included in the file name when you export the solution.

3. Select **Save**.
4. Open the solution, and then select **New > Other > Web resource**.



5. Enter the name of the web resource, and select the **Type as JavaScript (JS)**.
6. Select **Text Editor**, copy the code shown below, paste it into the text editor, and enter the value of the URL you want to open:

```
JavaScript

function ChangeBehavior(){

    // Enter the url
    var url = "Enter the URL";
    var OpenUrlOptions = {height: 800, width: 1000};
    Xrm.Navigation.openUrl(url, openUrlOptions);
}
```

7. Save and publish the web resource.

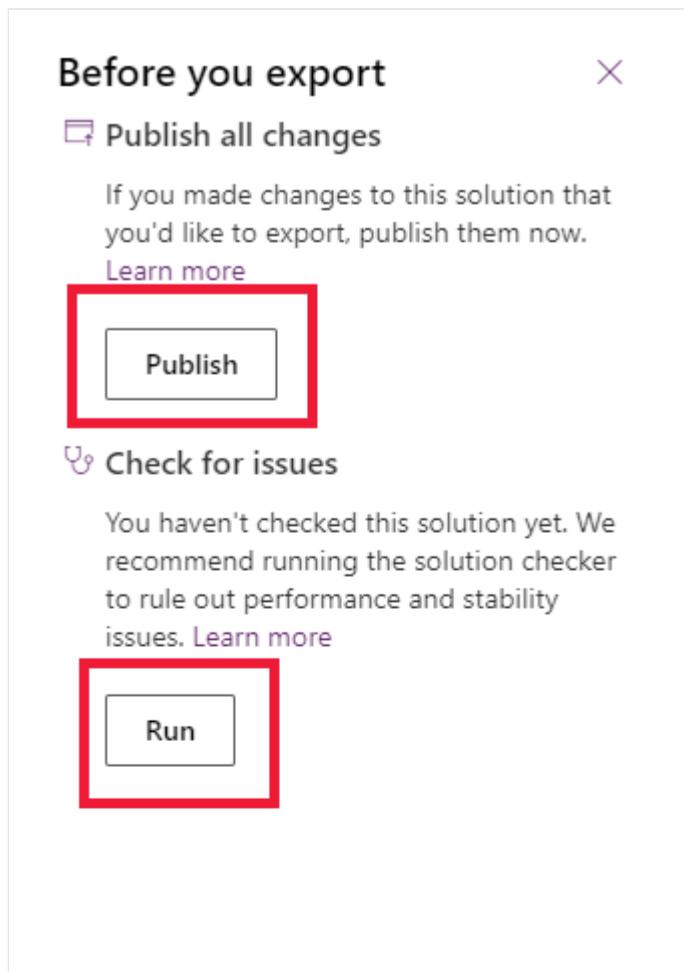
Step 2: Create a custom button

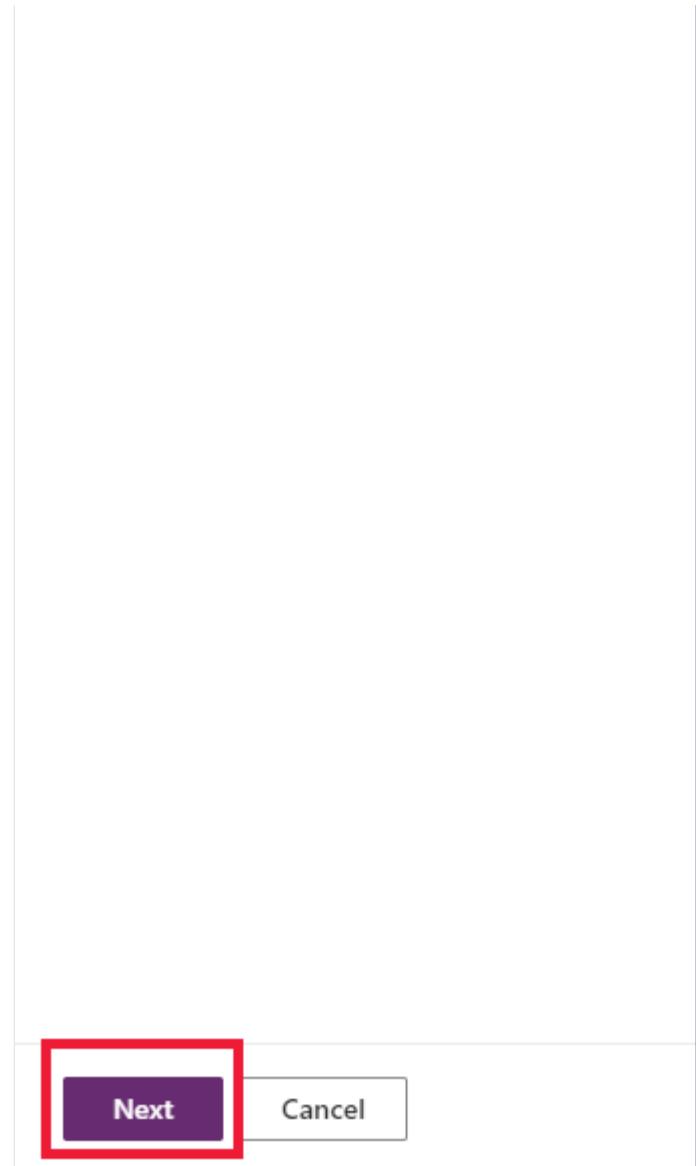
Create a custom button on the form where you want to change the default behavior. For example, if you have a subgrid on the accounts form that displays contact records in the subgrid, you need to create a button and add it to the contact form. You can create a button by editing the customization.xml file.

1. Open the solution that you created in step 1, and add the table where you want to create the button. It is not required to include all table components and metadata.
2. Select **Add existing > Table**
3. From the list, select **Contact**.
4. Save and publish the solution.
5. Select **Export** to make edits to the customization.xml file.



6. If you've made recent changes that haven't yet been published, select **Publish**, select **Run** to check whether the solution has any issues or dependencies, and then select **Next**.





7. With the **Unmanaged** option selected, select **Export**.

← Export this solution ×

Version number* ⓘ
Current version 1.0.0.0
1.0.0.1

Export as

Managed (recommended) ⓘ
The solution is moving to a test or production environment. [Learn more](#)

Unmanaged
The solution is moving to another development environment or source control. [Learn more](#)



8. In the **Download** dialog box, select **Save**, and in the **Download complete** dialog box, select **Open Folder**.
9. Right-click to select the compressed .zip file that you downloaded, and then select **Extract All**.
10. Select a location to extract the files to, and then select **Extract**.

The customizations.xml file is the file that you'll edit.

 **Note**

You can enable or disable the button; doing either will still override the open default behavior.

11. Open the customization.xml file, copy the code below, replace the code inside the

RibbonDiffXml:

XML

```
<RibbonDiffXml>
<CustomActions>
    <CustomAction Id="cr5c1.Mscrm.OpenRecordItem.CustomAction"
        Location="Mscrm.SubGrid.contact.MainTab.Management.Controls._children"
        Sequence="28">
        <CommandUIDefinition>
            <Button Alt="$LocLabels:Mscrm.OpenRecordItem.Alt"
                Command="Mscrm.OpenRecordItem"
                Id="Mscrm.OpenRecordItem"
                LabelText="$LocLabels:Mscrm.OpenRecordItem.LabelText"
                Sequence="28"
                TemplateAlias="o1"
                ToolTipTitle="$LocLabels:Mscrm.OpenRecordItem.ToolTipTitle"
                ToolTipDescription="$LocLabels:Mscrm.OpenRecordItem.ToolTipDescription"
            />
        </CommandUIDefinition>
    </CustomAction>
</CustomActions>
<Templates>
    <RibbonTemplates Id="Mscrm.Templates" />
</Templates>
<CommandDefinitions>
    <CommandDefinition Id="Mscrm.OpenRecordItem">
        <EnableRules />
        <DisplayRules />
        <Actions>
            <JavaScriptFunction FunctionName="ChangeBehavior"
                Library="$webresource:cr5c1_samplescript" />
        </Actions>
    </CommandDefinition>
</CommandDefinitions>
</RibbonDiffXml>
```

① Note

You need to replace the function name and also the name of the web resource in the above XML file. Edit the above example XML file to replace it with your own default publisher.

This example is to change the button for a button for a subgrid on the accounts form that displays contact records in the subgrid. Therefore the **Location** is

Location="Mscrm.SubGrid.contact.MainTab.Management.Controls._children".

You will need to change this to have this apply to a different button.

Step 3: Import the XML file

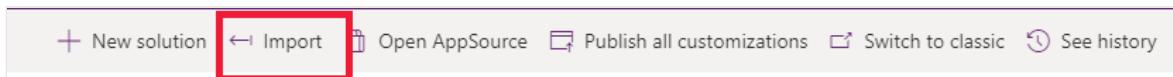
1. After you've edited the customization.xml file, right-click to select the **customization.xml** tab, and then select **Open Containing Folder**.
2. Select all the files or folders that were included when you extracted the solution. Right-click the selected files, select **Send To**, and then select **Compressed (zipped) folder**.

ⓘ Note

This creates a compressed .zip file in the same folder. The name of the file will vary, but will be the same as one of the other files in the folder except with a .zip file name extension.

3. Sign in to [Power Apps](#), and select **Solutions** from the left pane.

4. On the command bar, select **Import**.



5. On the **Select Solution Package** page, select **Browse** to locate the compressed (.zip or .cab) file that contains the solution you want to import.
6. Select **Next**.
7. On the page that displays information about the solution, select **Import**.
8. You might need to wait a few moments while the import is completed. View the results, and then select **Close**.

If you've imported any changes that require publishing, you must publish customizations before they're available.

If the import isn't successful, you'll see a report showing any errors or warnings that were captured. Select **Download Log File** to see details about what caused the import to fail. The most common cause for an import to fail is that the solution didn't contain some required components.

When you download the log file, you'll find an XML file that you can open with Excel to view the contents.

See also

[Ribbon Workbench ↗](#)

[Customize the ribbon](#)

Pass data from a page as a parameter to ribbon actions

Article • 11/30/2022

ⓘ Note

This topic is about classic commands.

There is a new way to define commands. See [Modern commanding overview \(preview\)](#).

When you define an action in a ribbon, you frequently have to pass data from the page to either a JavaScript function or a URL. This article describes options for using the [`<CrmParameter>`](#) element to retrieve these values.

Form and grid context in ribbon actions

To pass in the execution context (*form context* or *grid context*) information to JavaScript function for your ribbon actions, specify **PrimaryControl** for the form context, or **SelectedControl** for the grid context as the `<CrmParameter>` value in your ribbon definition. **SelectedControl** will pass in the grid context, for both subgrids and homepage grids. The passed in **PrimaryControl** or the **SelectedControl** value is used as an argument in your JavaScript function for *form context* or *grid context* respectively.

For example, here is a sample ribbon definition where we pass in the **PrimaryControl** parameter to the JavaScript function:

XML

```
<CommandDefinition Id="SampleCommand">
  <EnableRules/>
  <DisplayRules/>
  <Actions>
    <JavaScriptFunction Library="$webresource:new_mySampleScript.js"
FunctionName="mySampleFunction">
      <CrmParameter Value="PrimaryControl" />
    </JavaScriptFunction>
  </Actions>
</CommandDefinition>
```

Next, in the `new_mySampleScript.js` web resource file referenced in the example above, define your JavaScript function with the `primaryControl` variable as an argument. This argument provides the `form` context where the ribbon command is executed:

JavaScript

```
function mySampleFunction(primaryControl) {  
    var formContext = primaryControl;  
    // Perform operations using the formContext object  
}
```

You can also specify `CommandProperties` as `<CrmParameter>` value in your ribbon definition to pass details about the event from the ribbon control. You can use this to send contextual information to your JavaScript function where specific actions can be determined based on the context of the event.

 **Note**

Getting *form context* and *grid context* for JavaScript functions for ribbon actions is different from how you get these values in form scripting. For information about form scripting and how to get these contexts, see [Client API form context](#) and [Client API grid context](#).

Form values

With a form ribbon, you can use the `data.entity.attributes` collection and the `ui.controls` collection to retrieve and set values for known columns.

For example, the following sample code shows how to retrieve the account name column on the account form, and then set a value in the websiteurl column based on the account name value:

JavaScript

```
function mySampleFunction(primaryControl) {  
    var formContext = primaryControl;  
    var accountName =  
        formContext.getControl("name").getAttribute().getValue();  
  
    // Set the WebSiteURL column if account name contains "Contoso"  
    if (accountName.toLowerCase().search("contoso") != -1) {  
  
        formContext.getAttribute("websiteurl").setValue("https://www.contoso.com");  
    }  
}
```

```
        else {
            Xrm.Navigation.openAlertDialog({ text: "Account name does not
contain 'Contoso'." });
        }
    }
```

Grid values

The majority of the values available for the `<CrmParameter>` element are related to working with data displayed in a grid or hierarchy chart. By using the `Value` parameter enumeration options, you can easily isolate items by:

- **Selected items**
 - `SelectedControlSelectedItemCount`
 - `SelectedControlSelectedItemIds`
 - `SelectedControlSelectedItemReferences`
- **All items**
 - `SelectedControlAllItemCount`
 - `SelectedControlAllItemIds`
 - `SelectedControlAllItemReferences`
- **Unselected items**
 - `SelectedControlUnselectedItemCount`
 - `SelectedControlUnselectedItemIds`
 - `SelectedControlUnselectedItemReferences`

For each of these groupings, you can gather the number of items and the GUID identifier. If you are passing the values to a URL, you can also retrieve `EntityReference` objects that contain all the information that you need to uniquely identify the objects in the grid. These parameters apply whether the page viewed is the main grid (`HomepageGrid`) or a sub grid located in a form. When used together with the `SelectedEntityType` parameter, you have all the information that you must have to pass to another application.

Other context information

In addition to data values, you can retrieve client context information by using `<CrmParameter>`. You can use the following options as the value for the `CrmParameter` element: `OrgName`, `OrgLcid`, and `UserLcid`.

For a `<Url>` action, you can also use the `PassParams` to include contextual information.

The `Value` options `PrimaryEntityTypeCode` and `FirstPrimaryItemId` provide information for a table record. You can use `PrimaryItemIds` for a `HomepageGrid` ribbon to get a list of all the displayed items.

See also

[Customize the ribbon](#)

[Passing parameters to a URL using the ribbon](#)

[Define ribbon actions](#)

[Define custom actions to modify the ribbon](#)

[Client API form context](#)

[Client API grid context](#)

Define custom actions to modify the ribbon

Article • 11/30/2022

ⓘ Note

This topic is about classic commands.

There is a new way to define commands. See [Modern commanding overview \(preview\)](#).

The default, an application command bar or ribbon is defined by Microsoft Dataverse metadata. This default data can't be changed, but you can include definitions of specific actions that will override the default ribbon.

Types of custom actions

There are two types of custom actions for ribbons:

- `<CustomAction>`: Defines an action to add or replace items in the ribbon.
- `<HideCustomAction>` : Removes an existing ribbon element from being processed for the ribbon.

Custom actions

A custom action is a statement of how you want to change the default ribbon definition. It is evaluated and applied to the ribbon at runtime. To set the context for a custom action, you must include information about the location of the items that you want to change. Use the `Location` parameter to specify where your change applies.

When you add a new ribbon element, you refer to the containing element, for example, an existing tab or group. You then include the suffix `._children` to indicate that this custom action will add something to an existing item.

When you change the definition of an existing item, the `Location` value will match the ID of that item.

You must also specify a unique identifier for the custom action. Use the `Id` parameter to set this value. We strongly recommend that you use a naming convention that will

guarantee a unique value. For consistency and readability, we recommend that you use a period to separate consistent components. The first item in your naming convention should be something related to your solutions publisher or solution, for example, `Contoso.contact.form.CustomButton.CustomButton`.

💡 Tip

Consistently applying your `Id` parameter naming conventions will greatly enhance your productivity while editing RibbonDiffXml.

Based on the location information that you provide, the `Sequence` value determines the order in which to render items. If you want a custom control to appear between two existing controls, you must select a sequence value that is in between the sequence values of the existing items.

Hide custom actions

A `<HideCustomAction>` is a statement that you use when you want to remove an existing ribbon element so that it is not rendered. This does not hide the ribbon element, it actually removes the ribbon element at runtime so that it doesn't exist in the ribbon.

The `HideActionId` element provides a unique ID for the action. For consistency and readability, you should follow the same naming convention described for `<CustomAction>` elements. The `Location` parameter must match the `Id` of the ribbon element you want to remove.

ⓘ Note

Because the `HideCustomAction` element removes a specified node from the ribbon, removing ribbon elements in this manner may not be the best option for every situation.

- If you want to remove a button that is associated with a specific privilege, you should adjust the privileges for the table in the security roles in your implementation. This will allow the default ribbon display and enables rules to hide or disable ribbon elements from users who do not have the necessary privileges to perform those actions.
 - If you want to replace an existing ribbon element with a custom ribbon element, you can overwrite that element by specifying a `CustomAction.Location` value identical to the existing element.

- To remove the `HideCustomAction` element you need to create a new updated version of the same solution that installed the `HideCustomAction` element. A new patch of the solution cannot remove the `HideCustomAction` element.

The `HideCustomAction` element cannot be removed, once added, except by creating a new updated solution. Instead, ribbon buttons should be hidden using a `DisplayRule` element that always evaluate to false. Having both `Mscrm.HideOnModern` and `Mscrm.ShowOnlyOnModern` would always evaluate to false. For example, to hide a deactivate button:

XML

```
<CommandDefinition Id="Mscrm.HomepageGrid.Deactivate">
  <EnableRules>
    </EnableRules>
  <DisplayRules>
    <DisplayRule Id="Mscrm.HideOnModern" />
    <DisplayRule Id="Mscrm.ShowOnlyOnModern" />
  </DisplayRules>
  <Actions>
    </Actions>
  </CommandDefinition>
```

See also

[Customize commands and the ribbon](#)

[Pass data from a page as a parameter to ribbon actions](#)

[Define scaling for ribbon elements](#)

Define scaling for ribbon elements

Article • 11/30/2022

ⓘ Note

This topic is about classic commands.

There is a new way to define commands. See [Modern commanding overview \(preview\)](#).

For application ribbons and updated form ribbons there is no scaling. Scaling only applies to forms for tables that weren't updated and list ribbons displayed using Dynamics 365 for Outlook.

The goal of the ribbon is to maintain visibility of relevant controls even when the horizontal size of the window changes. To achieve this, the UI definition allows you to control how controls in a group change size in response to changes in the size of the window. This is known as *scaling*.

Associate groups and controls to layout templates

Each `<Group>` element in the ribbon is associated with a `<GroupTemplate>`. The `GroupTemplate` specifies one or more ways the controls in the group can be presented using `<Layout>` elements. Each `Layout` may contain one of two types of definition for how the controls in the group are displayed.

- An `<overflowSection>` allows for controls to change relative position depending on the available space.
- A `<Section>` controls the number of rows to display and where each control is displayed.

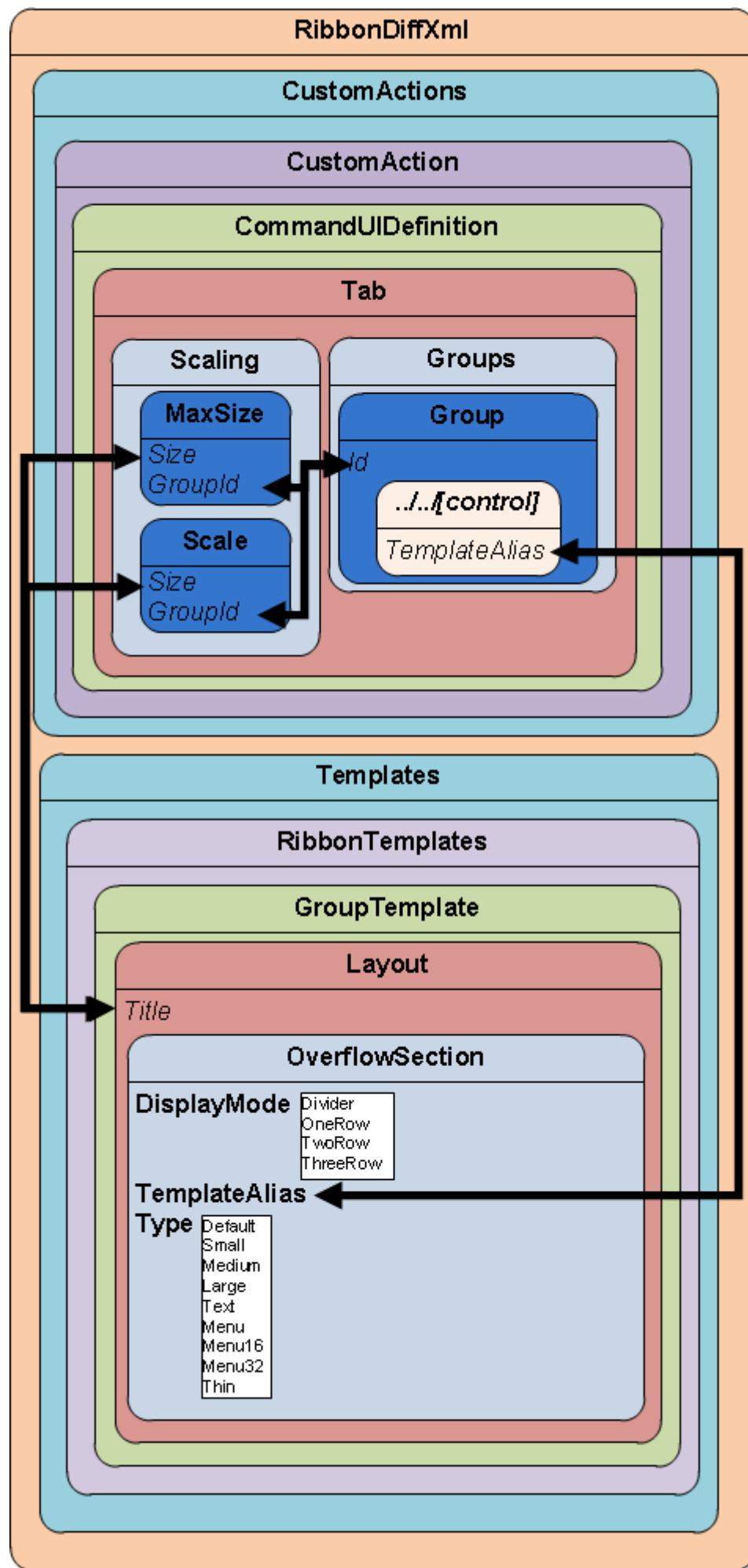
Almost all the `Layout` elements used in ribbons use `OverflowSection` elements.

Each `<Tab>` element must contain one `<MaxSize>` in the `<Scaling>`. The `MaxSize` element is required because it establishes the default presentation of each `Group` in a `Tab` without any scaling applied. Scaling occurs when a `Tab` is associated with one or more `<Scale>`. Each `MaxSize` and `Scale` element is associated via the `Size` parameter with one of the `Layout` elements in the `GroupTemplate` used by each `Group` within a `Tab`.

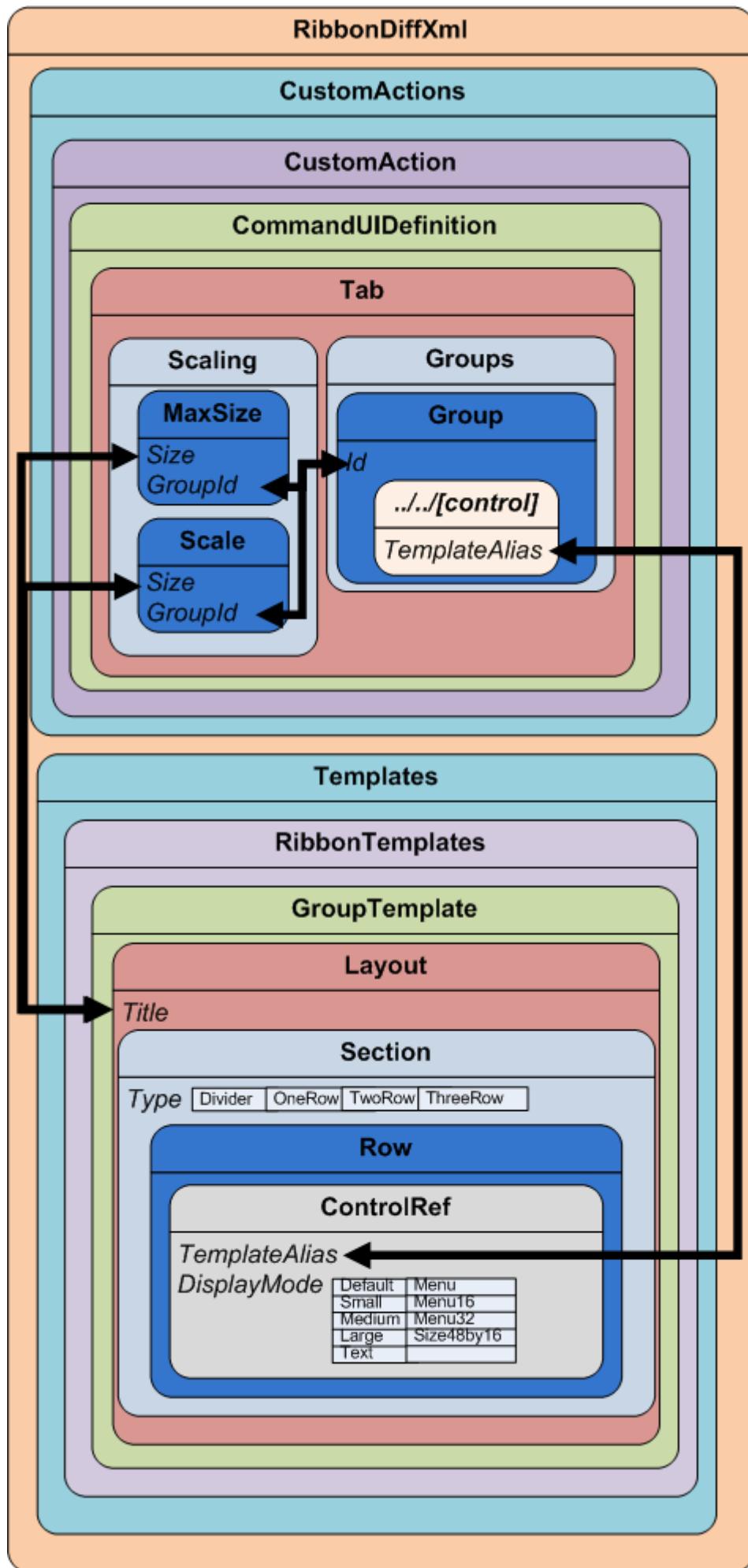
Note

The value of the `Size` parameter of any `MaxSize` or `Scale` element must match the `Title` of the available `Layout` elements specified in the `GroupTemplate`. These values are strings and there is no validation in the XSD to help you select values that are a match. The XML is always case-sensitive.

The following diagram shows how `MaxSize`, `Scale`, `Group`, `Layout` and `OverflowSection` elements must reference each other to enable scaling when you are using a `<OverflowSection>` element.



The following diagram shows how `MaxSize`, `Scale`, `Group`, `Layout` and `ControlRef` elements must reference each other to enable scaling when you are using a `<Section>` element.



Use existing group templates

When creating a new group, instead of defining new group templates, you can re-use existing `GroupTemplate` elements.

Associate your new group to that template. For each control in the group, use a `TemplateAlias` value from one of the `<Section>` or `<OverflowSection>` elements found in one of the `Layout` elements used by that `GroupTemplate`. Each `<OverflowSection>` includes an `isv``TemplateAlias` that is not used. This `TemplateAlias` is provided to allow ISVs to add controls to that group.

Control how scaling is applied

Each `Scale` element in the `Scaling` element for a particular tab represents one scale step. Each `Scale` is applied sequentially by the order in which the `Scale` element appears. When reducing the horizontal space available for the ribbon, each scale element is applied in order from top down. When increasing the horizontal space available, from the smallest space the bottom scale element is in effect. Each of the available `Scale` elements are applied in order from the bottom to the top until all the `MaxSize` elements are in effect.

ⓘ Note

The `Scale` element `Sequence` values aren't used to determine the order in which scaling is applied. Scaling is applied by the relative order the `MaxSize` and `Scale` elements appear in the RibbonDiffXML. The `Sequence` value is important for both `MaxSize` and `Scale` elements because all the `MaxSize` elements must be grouped together above the `Scale` elements. When you add new `MaxSize` or `Scale` elements, be sure that you review the `Sequence` default value ranges assigned to all the `MaxSize` elements and the `Scale` elements. A common error is to assign `Sequence` values that could cause the ranges to overlap.

See also

[Customize commands and the ribbon](#)

[Define custom actions to modify the ribbon](#)

[Define ribbon tab display rules](#)

Define ribbon tab display rules

Article • 11/30/2022

ⓘ Note

This topic is about classic commands.

There is a new way to define commands. See [Modern commanding overview \(preview\)](#).

Tab display rules control whether a specific tab is displayed for a ribbon or not.

Unlike other ribbon elements like groups or specific controls, you must explicitly provide a tab display rule for a tab to be displayed in the ribbon. By default, other ribbon elements will always display unless a display rule removes them.

`<TabDisplayRule>` elements require that the `TabCommand` parameter matches a `<Tab> Command` value.

In the `RibbonDiffXml`, tabs can be defined for specific tables or defined globally. If the tab is defined for a table, the only valid type of rule is `<EntityRule>`. Because defining a tab in the scope of a particular table already limits the tab to only that table, the only valid parameters are `AppliesTo (PrimaryEntity or SelectedEntity)` and `Context (Form, HomePageGrid, SubGridStandard, or SubGridAssociated)`.

When you define a tab display rule globally in the `RibbonDiffXml` for the application ribbons, you can apply both `EntityRule` elements and `<PageRule>` elements.

See also

[Customize commands and the ribbon](#)

[Define scaling for ribbon elements](#)

[Pass parameters to a URL by using the ribbon](#)

Pass parameters to a URL by using the ribbon

Article • 11/30/2022

ⓘ Note

This topic is about classic commands.

There is a new way to define commands. See [Modern commanding overview \(preview\)](#).

Ribbon actions are defined in the `<Actions>` element of a `<CommandDefinition>` element. There are several ways to pass contextual model-driven apps information as query string parameters to a URL by using the ribbon.

- Use a `<Url>` element. Within the `Url` element, use the **PassParams** parameter.
- Use a `<Url>` element together with a `<CrmParameter>` element. When used from a `Url` element, the name parameter value must be set.
- Use a `<JavaScriptFunction>` element together with a `<CrmParameter>` element.

Use the PassParams parameter to set dynamic values

Passing parameters to the target URL by using the **PassParams** parameter provides information to the target application about the context of the record or the user. All the parameters are passed if the ribbon control is configured by using the **PassParams**. The following table lists the parameters that are passed.

Parameter	Name	Description
<code>typename</code>	Table Name	Name of the table. For custom tables, this includes the customization prefix, for example, new_tablename.
<code>type</code>	Entity Type Code	Integer that uniquely identifies the table in the current organization. Note: <code>Entity Type Code</code> values are determined by the order in which a table is created in an organization. <code>Entity Type Codes</code> for custom tables are usually different in different organizations.
<code>id</code>	Object GUID	Globally unique identifier (GUID) that represents a record.

Parameter	Name	Description
<code>orgname</code>	Organization Name	Unique name of the organization.
<code>userlcid</code>	User Language Code	Language code identifier that is used by the current user.
<code>orglcid</code>	Organization Language Code	Language code identifier that represents the base language for the organization.

Language codes are four-digit or five-digit locale IDs. Valid locale ID values can be found at [Locale ID \(LCID\) Chart](#).

ⓘ Note

We recommend that you use the table name instead of the entity type code because the entity type code may be different between model-driven apps installations.

Example

The following sample shows the URL without parameters:

```
https://myserver/mypage.aspx
```

The following sample shows the parameters included when the ribbon control is presented for the account table, for an organization called 'AdventureWorksCycle', when the user's language and the organization base language is English, and the GUID for the account record is DBD5DBFB-0666-DC11-A5D9-0003FF9CE217:

```
https://myserver/mypage.aspx?
orgname=AdventureWorksCycle&userlcid=1033&orglcid=1033&type=1&typename=account&id=%7BDBD5DBFB-0666-DC11-A5D9-0003FF9CE217%7D
```

Use a Querystring parameter in the URL

You can include a `querystring` parameter in the URL. This can be very useful if you want to open a specific record or view by using [Open forms, views, dialogs, and reports with a URL](#).

 **Note**

You will not be able to import the ribbon if the URL includes the ampersand (&) character that is used to separate multiple `querystring` parameters in the URL. This character makes the XML invalid. You must escape the ampersand character in the URL value with "&".

Reading passed parameters

Passed parameters are usually read in the target .aspx page by using the `HttpRequest.QueryString` property. More information: [HttpRequest.QueryString Property](#)

 **Note**

If the target of your URL is a Web resource, it can accept only the parameters identified in the topic [Passing Parameters to HTMLWeb Resources](#). The only opportunity to pass custom values is by including them within the `data` parameter. Some special handling is required to include multiple values in a single parameter. More information: [Sample: Passing Multiple Values to a Web Page Web Resource Through the Data Parameter](#)

See also

- [Customize commands and the ribbon](#)
- [Open forms and views with a URL](#)
- [Define ribbon tab display rules](#)
- [Sample: Export ribbon definitions](#)

Web resources in model-driven apps

Article • 12/16/2022

Web resources are *virtual files* that are stored in the Microsoft Dataverse database and that you can retrieve by using a unique URL address.

Note

IFRAMEing content that is behind an authentication boundary is not supported through web resources or Power Apps component framework. Some embedded IFRAMES might work in a browser client if the user directly logs into the external service, but this is not supported in mobile or tablet applications. The specific scenario of embedding a form within an IFRAME, embedded in another form, is not supported. We recommend the use of **form as a component** for such scenarios.

In general, use of **Power Apps component framework** and **custom pages** is encouraged to build configurable, reusable and more tighter external integrations. More information: [IFRAME component](#)

Capabilities of web resources

Web resources represent files that can be used to extend the Dataverse web application such as html files, JavaScript, and CSS, and several image formats. You can use web resources in form customizations, the [SiteMap](#), or the application ribbon because they can be referenced by using URL syntax.

The URL syntax for web resources allows for relative path references. With your development tools, you can create a group of interdependent files on a development server by using file types compatible with web resources. Then, if you use a consistent naming convention and relative path references, the website will function after you upload all the files into Dataverse.

Because web resources are stored in Dataverse and are solution components, they can be easily exported and installed to other Dataverse orgs. Web resources are also available to users of Dataverse for Microsoft Office Outlook with Offline Access when offline because they are synchronized with the user's data.

You can use the form editor to add and configure form-enabled web resources into your forms.

Because web resources are stored as records in the database, they can be managed programmatically by using the standard techniques to create, retrieve, and update records. Text-based web resources (JScript, CSS, XML, XSL, RESX, and HTML) can be edited and saved in the application.

Limitations of web resources

There is no type of web resource that supports the capabilities of an ASP.NET(.aspx) page to execute code on the server. Web resources are limited to static files or files that are processed in the browser. A web resource can contain code that is processed in the browser to execute web service calls to interact with Dataverse data.

Web resources are only available by using the Dataverse web application security context. Only licensed Dataverse users who have the necessary privileges can access them.

Size limitations

The maximum size of files that can be uploaded is determined by the Organization.MaxUploadFileSize property. This property is set in the Email tab of the System Settings in the Dynamics 365 application. This setting limits the size of files that can be attached to email messages, notes, and web resources. The default setting is 5 MB.

Web resource types

You can use ten file formats to create web resources. The following table lists each file format, the allowed file extensions, and the type value that you use for each.

File	File extensions	Type
Webpage (HTML)	.htm, .html	1
Style Sheet (CSS)	.css	2
Script (JScript)	.js	3
Data (XML)	.xml	4
Image (PNG)	.png	5
Image (JPG)	.jpg	6
Image (GIF)	.gif	7

File	File extensions	Type
Silverlight (XAP)	.xap	8
StyleSheet (XSL)	.xsl, .xslt	9
Image (ICO)	.ico	10
Vector format (SVG)	.svg	11
String (RESX)	.resx	12

Reference web resources

There are several methods that you can use to reference web resources.

ⓘ Note

- When possible, use the `$webresource` directive. Only references that use the `$webresource` directive in the site map or ribbon commands will establish dependencies. Dependencies are not created when web resources reference each other.
 - To display a Silverlight web resource outside a form or chart, create an HTML web resource to be the host page for the Silverlight web resource. Then use the `$webresource:` directive to open the HTML web resource.

\$webresource directive

You should always use the `$webresource` directive when referencing a web resource from a ribbon control or from a `SiteMap` sub area. Use the `$webresource` directive anywhere the XML allows a URL value. The following sample shows how to use it.

XML

```
$webresource:<name of Web Resource>
```

ⓘ Note

When using the `$webresource` directive, Dataverse will create or update solution dependencies.

Xrm.Navigation.openWebResource

The Xrm.Navigation.openWebResource function will open an HTML web resource in a new window with parameters to pass the name of the web resource, any query string data to be passed in the data parameter, and information about height and width of the window.

The URL generated includes the unique GUID token so that the cached web resource will be loaded.

Relative URL

When referencing a web resource from areas that do not support using the \$webresource: directive, a relative URL can be used. To enable this, we recommend that you use a consistent naming convention for the web resources that reflect a virtual file structure. The solution publisher's customization prefix will always be included as a prefix to the name of the web resource. This can represent a virtual "root" folder for all web resources added by that publisher. You can then use the forward slash character (/) to simulate a folder structure that will be honored by the web server.

From another web resource, you should always use relative URLs to reference each other. For example, for the web page web resource new_/content/contentpage.htm to reference the CSS web resource new_/Styles/styles.css, create the link as follows:

HTML

```
<link rel="stylesheet" type="text/css" href="../styles/styles.css" />
```

For the web page web resource new_/content/contentpage.htm to open the webpage web resource isv_/foldername/dialogpage.htm, create the link as follows:

HTML

```
<a href=".../.../isv_/foldername/dialogpage.htm">Dialog Page</a>
```

ⓘ Note

Do not use a relative URL using the WebResources folder as the root path for the URL. For example, do not use this: /WebResources/<name of web resource>. When a user belongs to more than one organization on a server, this path will always refer to the users default organization. If the user is not using their default organization

and the expected web resource is not included in the user's default organization, a "File Not Found" error occurs even though the web resource does occur in the organization the user is currently working in.

Full URL

The following sample shows the style of URL you can use to view web resources.

```
<Dataverse Environment URL>/WebResources/<name of web resource>
```

The application will process this URL and return the file that contains the latest version of the web resource. This URL will look like this:

```
<Dataverse Environment URL>%7B<version value>%7D/WebResources/<name of web resource>
```

The version value is updated when you publish customizations and ensures that the browser uses the latest cached version of the web resource. Because of this, use a relative path to a web resource, the Xrm.Navigation.[openWebResource](#) function, or the [\\$webresource Directive](#) (when possible) because the version value will automatically be included. For large web resources there can be significant performance implications if you don't use the cached version of the file.

The following sample shows a URL for Dataverse, where `MyOrganization` is the name of your Dataverse Environment, and `new_/test/test.htm` is the name of the web resource:

```
https://MyOrganization.crm.dynamics.com/WebResources/new_/test/test.htm
```

ⓘ Note

Including the '/' character and a file name extension in the name of the web resource is an optional best practice.

When you write code to reference a web resource that works for Dataverse, you should use the `getClientUrl` function.

Layout differences between the legacy web client and Unified Interface

A web resource control configured to use a certain number of rows will have different heights in a Unified Client application compared to a web client application. This is because there is a difference in the height of a row between Unified Interface and web client. If a form is needed in both the web client and Unified Interface, you can use different forms in the Unified Interface app and the web client app with the control configured to use the appropriate number of rows in each form.

Community tools

WebResources Manager is a tool that XrmToolbox community developed for Dataverse. Please see the [Developer tools](#) topic for community developed tools.

Note

The community tools are not a product of Dataverse and does not extend support to the community tools. If you have questions pertaining to the tool, please contact the publisher. More Information: [XrmToolBox](#).

See also

[Create Accessible web resources](#)

[Web Page \(HTML\) web resources](#)

[JavaScript web resources](#)

[Image web resources](#)

[Stylesheet \(XSL\) web resources](#)

[Data \(XML\) web resources](#)

[Style Sheet \(CSS\) web resources](#)

[Web resource table reference](#)

[Sample: Passing multiple values to a web resource through the data parameter](#)

[Sample: Importing files as web resources](#)

[Streamline web resource development using Fiddler AutoResponder](#)

Use web resources

Article • 12/16/2022

There is a virtual folder called `webresources` within each Microsoft Dataverse instance where you can request HTML, JS, CSS, image, and other files by name and access them in your browser. You can upload these files using the application or programmatically add them as [Web resource table](#) records. The [XrmToolBox web resources manager](#) is a community tool which can facilitate working with these records.

These records can refer to each other using relative path names in their content. This ability to upload files and request them by name provides all the building blocks you need to make web applications using files that are processed within the authenticated session of your browser. Using only client-side code with AJAX techniques, you can create rich applications that can run within a browser window or within an IFrame in a form or dashboard.

Most commonly, you will use JavaScript web resources to add event handler functions to forms and commands.

More information:

- [Client scripting with model-driven apps](#)
- [Web resources](#)

Webpage (HTML) web resources

Article • 12/16/2022

Use webpage (HTML) web resources to create user interface elements for client extensions.

Capabilities of HTML web resources

Because an HTML web resource is just streamed to the user's browser, it can include any content that is rendered on the user's browser.

Limitations of HTML web resources

- An HTML web resource can't contain any code that must be executed on the server. ASP.NET pages can't be uploaded as HTML web resources.
- HTML web resources can only accept a limited number of query string parameters.
[Pass parameters to HTML web resources](#)
- HTML web resources embedded as controls in a form can be reloaded by the form runtime for performance reasons. For example, the form runtime may destroy and re-initialize the control during tab navigations.

Use the text editor for HTML web resources

The text editor provided in the web resource form is intended for use with very simple HTML editing. For more sophisticated HTML documents, you should edit the code in an external editor and use the **Browse** button to upload the contents of your file.

For example, a more complex HTML page that requires script to render the contents of the page will begin like the following sample.

HTML

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"https://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html>
<head>
<title></title>
<script src="Script/Script.js" type="text/javascript"></script>
<link href="CSS/Styles.css" rel="stylesheet" type="text/css" />
</head>
```

```
<body onload="SDK.ImportWebResources.showData()">
<div id="results" />
</body>
</html>
```

After the document is opened in the text editor and saved, the HTML will be changed to this.

HTML

```
<HTML><HEAD><TITLE></TITLE>
<META charset=utf-8></HEAD>
<BODY contentEditable=true onload=SDK.ImportWebResources.showData()>
<SCRIPT type=text/javascript src="Script/Script.js"></SCRIPT>
<LINK rel=stylesheet type=text/css href="CSS/Styles.css">
<DIV id=results></DIV></BODY></HTML>
```

Prevent editing of web resources for managed solutions

Because of the capability for the HTML in web resources to be changed by using the text editor, it's recommended that you use managed properties to set complex HTML web resources as not customizable for managed solutions. When viewing web resources in the solutions window, open the **Managed Properties** dialog box to set the **Customizable** property to `false`.

Reference other web resources from an HTML web resource

You can create a set of related files outside of Model Driven Apps that use any of the web resource file types. If you're careful to always use relative paths and import each web resource with a consistent naming convention that reflects the folder structure of your website, you'll find that the HTML web resource will maintain links to related CSS, XML, JScript, image, and Silverlight files that have been imported as web resources.

For example, if you create a web application project that uses the following [folder]/file structure:

- page.htm
- [Styles]
 - style.css

- [Scripts]
 - script.js

When you import these files as web resources, you can name where your solution publisher customization prefix is “new” in the following manner:

- new_/page.htm
- new_/Styles/style.css
- new_/Scripts/script.js

When you follow this pattern, your new_/page.htm HTML web resource can reference the other files the most common way using relative paths as shown in the following example.

HTML

```
<script src="Scripts/script.js" type="text/javascript"></script>
<link href="Styles/style.css" rel="stylesheet" type="text/css" />
```

The solution publisher customization prefix becomes a virtual root folder for all the web resources in your solution. If you change your customization prefix, the relative paths within your HTML web resources won’t be changed.

① Note

- An HTML web resource added to a form can’t use global objects defined by the JavaScript library loaded in the form. An HTML web resource may interact with the Xrm.Page or Xrm.Utility objects within the form by using parent.Xrm.Page or parent.Xrm.Utility, but global objects defined by form scripts won’t be accessible using the parent. You should load any libraries that an HTML web resource needs within the HTML web resource so they’re not dependent on scripts loaded in the form.
- References included in code between web resources aren’t tracked as solution dependencies.

Because web resources are also downloaded for users of Dynamics 365 for Microsoft Office Outlook with Offline Access, users will have access to web resource content while they’re working offline.

Pass parameters to HTML web resources

An HTML web resource can accept only the parameters in the following table.

Parameter	Name	Description
typename	Table Name	The name of the table.
type	Table Type Code	An integer that uniquely identifies the table in a specific organization.
id	Object GUID	The GUID that represents a record.
orgname	Organization Name	The unique name of the organization.
userid	User Language Code	The language code identifier being used by the current user.
orglcid	Organization Language Code	The language code identifier that represents the base language for the organization.
data	Optional Data Parameter	An optional value that may be passed.
formid	Form Id	The GUID that represents a form ID.
entrypoint	Entry Point	A string value. This parameter is intended to be passed as an optional value to web resources opened as custom help content for a table. When enabled, the custom help URL will include a value of either "form" or "hierarchychart".
pagemode		For internal use only.
security		For internal use only.
tabSet		For internal use only.

If multiple values are passed in the data parameter, they will be automatically encoded. Logic must also be included to decode the multiple parameters using script in your HTML web resource. The [Sample: Passing multiple values to a web resource through the data parameter](#) topic demonstrates one approach to address passing multiple parameter values.

 **Note**

All characters included in the query string go through validation to ensure the validity of the parameters passed. If there are any parameters found to be not valid, the request will fail. For example, passing text values enclosed in angular brackets is considered an invalid parameter type.

See also

[Web resources](#)

[Create accessible web resources](#)

[Using Style Sheet \(CSS\) web resources](#)

[Using JavaScript web resources](#)

[Using Data \(XML\) web resources](#)

[Using Image \(JPG, PNG, GIF\) web resources](#)

[Using Stylesheet \(XSL\) web resources](#)

Create accessible web resources

Article • 12/16/2022

When you include web resources that provide user interface elements in your solution, make sure that you include requirements that let people with disabilities use your web resources.

The application user interface elements follow standards and best practices that will allow for equivalent functionality for all users. People with disabilities may rely on the use of assistive technology (AT) such as screen readers or a variety of alternative input devices to interact with applications.

This topic introduces general guidance and links to more resources that will help you design web resource user interface elements that are accessible to people with disabilities.

Assistive technology

There are a variety of assistive technology (AT) applications that include screen readers, Braille terminals, and speech recognition software. These applications provide an intermediary with your user interface elements so that people using the AT application can use your program.

For Windows applications, the Microsoft UI Automation (UIA) classes provide programmatic access to user interface elements. These classes support both automated testing and AT. AT applications can use the properties and elements defined by the developer and exposed through UIA. A windows application developer has considerable control over how their UI elements are exposed by using UIA.

For web applications, certain HTML elements are exposed through the Document Object Model (DOM). The browser converts DOM elements to UIA objects with properties and events that AT can use to enable the user to use the web application. The developer has limited control over how the UI elements are exposed by the browser that uses UIA.

Accessible HTML web resources

The HTML in your web resources is processed by the browser and made available to AT applications.

The first thing to consider is making sure that your HTML follows expected patterns of usage. For example, you can define an HTML `div` element with a click event so that it

functions exactly like an HTML `button` element. However, the browser will not expect that a `div` element is being used as a button and will not expose the same properties and events to an AT application.

It is important that you use the correct HTML elements for the types of interactions users will have with your web resources. This is known as [semantic HTML](#).

However, semantic HTML can only go so far. Modern web applications typically include custom controls that are composed of many HTML elements working together. Page content that is frequently updated dynamically using asynchronous JavaScript is confusing for AT applications that rely only on semantic HTML. [Accessible Rich Internet Applications \(ARIA\)](#) technology provides a solution by extending HTML with additional attributes that communicate custom semantics.

ARIA provides a standard set of extended attributes that can be applied to HTML elements that are used in a control, or “widget.” These attributes describe the role that the HTML element plays in the control. ARIA also provides capabilities to improve the navigation experience and make the user aware of elements that may be updated dynamically. The recommended practice is to layer ARIA over semantic HTML.

In addition to including support for AT, there are other requirements you have to consider. For example, how does the UI adjust when the user increases the text size? Does your UI require that the user be able to differentiate colors to perform tasks? Can all actions be performed by using a keyboard? For more information, see [Introduction to Web Accessibility](#).

Accessibility testing tools

The following list provides some publicly available accessibility testing tools:

[Visual Studio Accessibility Checker](#)

If you are using Visual Studio to edit your HTML web resource files, you will find that there are built-in tools to check for issues related to accessibility. In the **Tools** menu, select **Check Accessibility** to see a report that will provide guidance about accessibility related issues.

[UI Accessibility Checker](#)

UI Accessibility Checker (or AccChecker) enables testers to easily discover accessibility problems with Microsoft Active Accessibility (MSAA) and other user interface (UI) implementations for Windows. AccChecker was born from the realization that existing Windows Automation API tools, such as Inspect, provided in-depth details on the

implementation, but no information about whether that implementation is correct or not.

[Inspect \(Inspect.exe\)](#)

Inspect (Inspect.exe) is a Windows-based tool that enables you select any UI element and view the element's accessibility data. You can view Microsoft UI Automation properties and control patterns in addition to Microsoft Active Accessibility properties. Inspect also enables you to test the navigational structure of the automation elements in the UI Automation tree, and the accessible objects in the Microsoft Active Accessibility hierarchy

[Accessible Event Watcher \(AccEvent.exe\)](#)

The Accessible Event Watcher (AccEvent) tool allows developers and testers to validate that an application's UI elements raise appropriate Microsoft UI Automation and Microsoft Active Accessibility events when UI changes occur. Changes in the UI can occur when the focus changes, or when a UI element is invoked, selected, or has a state or property change.

Additional resources

The following resources provide a starting point for defining requirements for making your web resources accessible:

- [Introduction to web accessibility](#)
- [Accessibility in Visual Studio and ASP.NET](#)
- [Accessibility overview](#)
- [Accessibility - W3C ↗](#)
- [Web Content Accessibility Guidelines \(WCAG\) 2.0 ↗](#)

See also

[Web Page \(HTML\) web resources](#)

[Web resources](#)

JavaScript web resources

Article • 12/16/2022

Use JavaScript web resources to create a library of JavaScript functions that can be accessed from anywhere.

Capabilities of JavaScript web resources

With JavaScript web resources, you can more efficiently manage code used in form scripts, webpage (HTML) web resources, or ribbon commands by linking them to shared library of JavaScript functions.

Limitations of JavaScript web resources

Like all web resources, JavaScript web resources use the model-driven apps security context. Only licensed users who have the necessary privileges can access them.

ⓘ Note

References included in code between web resources aren't tracked as solution dependencies.

Using JavaScript libraries

For information about developing and testing JavaScript libraries as well as how to associate them with ribbon commands and form events, see [Client scripting using JavaScript](#).

Referencing a script web resource from a webpage web resource

All web resources can use relative URLs to reference each other. In the following example, for the webpage web resource `new_/content/contentpage.htm` to reference the JavaScript web resource `new_/scripts/myScript.js`, add the following HTML code to the head element of `new_/content/contentpage.htm`.

HTML

```
<script type="text/jscript" src="../scripts/myScript.js"></script>
```

To reference a JavaScript from a different publisher, the path must include the customization prefix for that publisher. For example, for the `new_/content/contentpage.htm` page to reference the `MyIsv_/scripts/customscripts.js` page, the `src` attribute value should be `../../MyIsv_/scripts/customscripts.js`.

See also

[Client scripting using JavaScript](#)

[Web resources](#)

[Using Web Page \(HTML\) web resources](#)

[Using Style Sheet \(CSS\) web resources](#)

[Using Data \(XML\) web resources](#)

[Using Image \(JPG, PNG, GIF\) web resources](#)

[Using Stylesheet \(XSL\) web resources](#)

[Streamline web resource development using Fiddler AutoResponder](#)

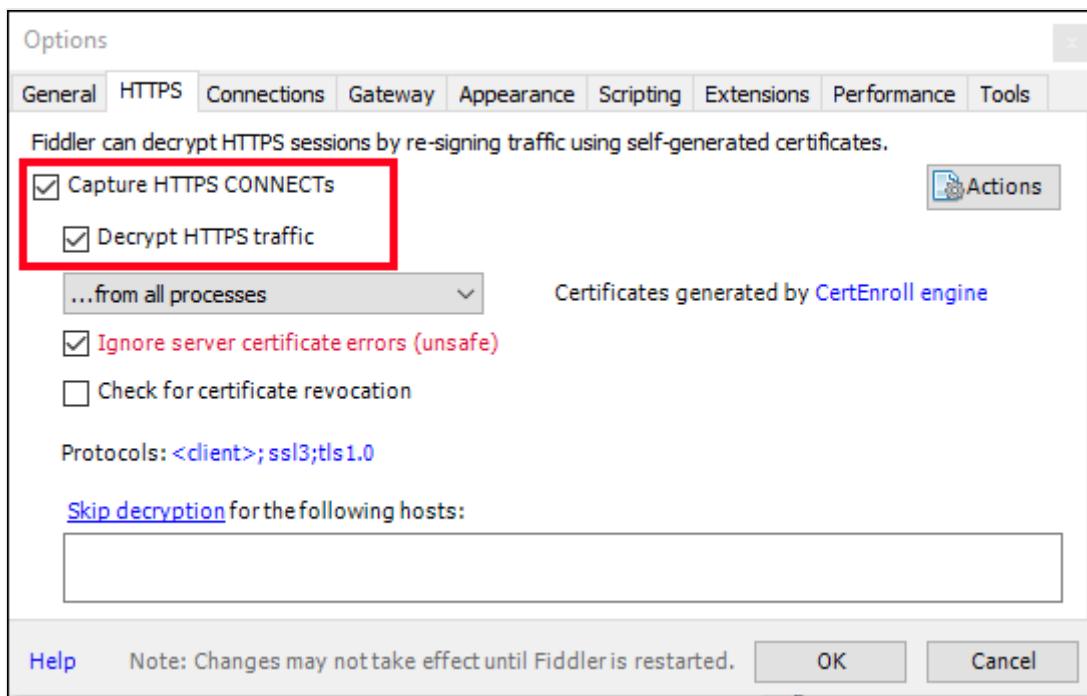
Script web resource development using Fiddler AutoResponder

Article • 12/16/2022

While developing and debugging JavaScript web resources, you can use AutoResponder in [Telerik Fiddler](#) to replace the content of a web resource with content from a local file rather than uploading it in your Model-driven Apps instance and publishing each time. Use the following steps below to setup AutoResponder in Fiddler.

Install and configure Fiddler

1. [Download](#) and install Fiddler.
2. Open Fiddler and from the menu bar, go to **Tools**, and then select **Options**.
3. Select the **HTTPS** tab in the dialog box and check the **Capture HTTPS CONNECTS** and **Decrypt HTTPS traffic** checkboxes so that the HTTPS traffic is captured and then decrypted.



4. Click **OK** to close the dialog box.

ⓘ Note

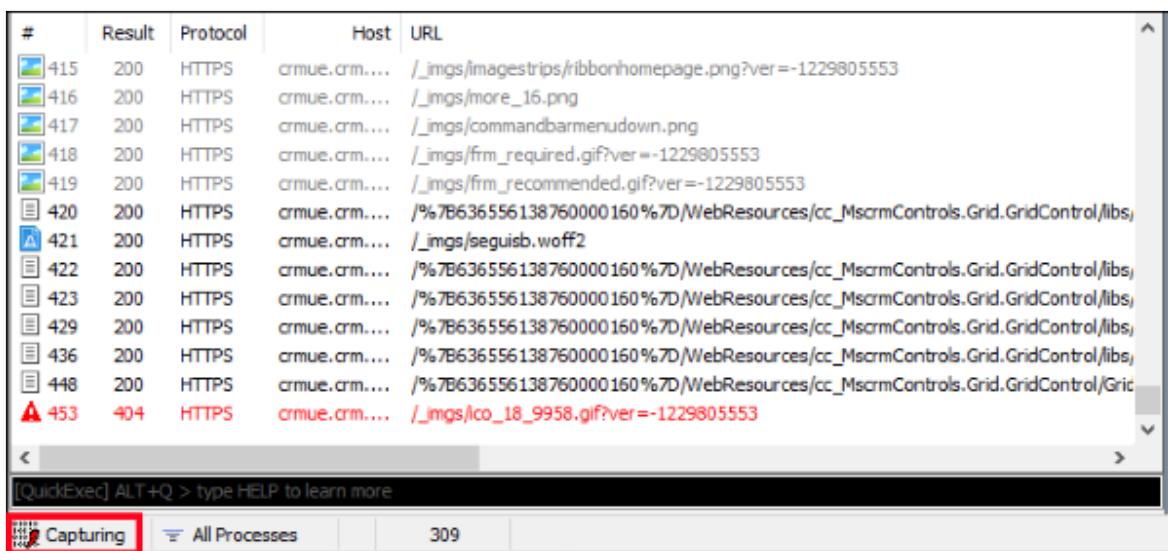
If it is the first time you are enabling this setting, Fiddler will prompt you to install a certificate. Install the certificate and restart Fiddler so that the new settings take

effect.

If you have run Fiddler in the past and get a `NET:::ERR_CERT_AUTHORITY_INVALID` error, in the **HTTPS** tab, click the **Actions** button and choose **Reset All Certificates**. This will also present a number of prompts for the new certificates to be installed.

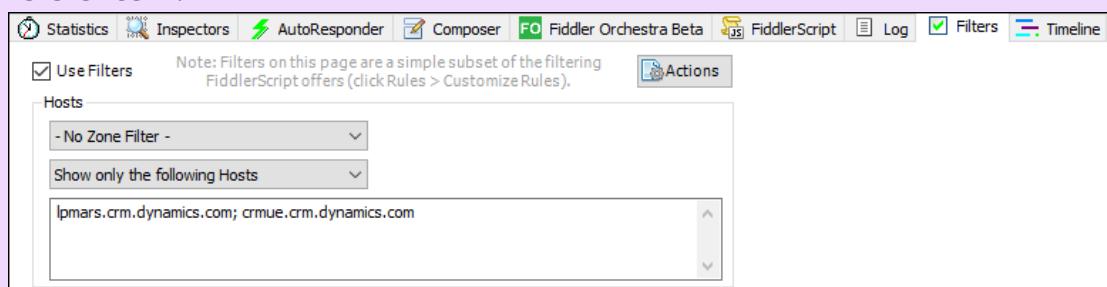
Configure AutoResponder

1. Open the page in the model-driven app that you want to debug.
2. Start the Fiddler trace capture by clicking the **Capturing** button in the bottom left corner.



ⓘ Note

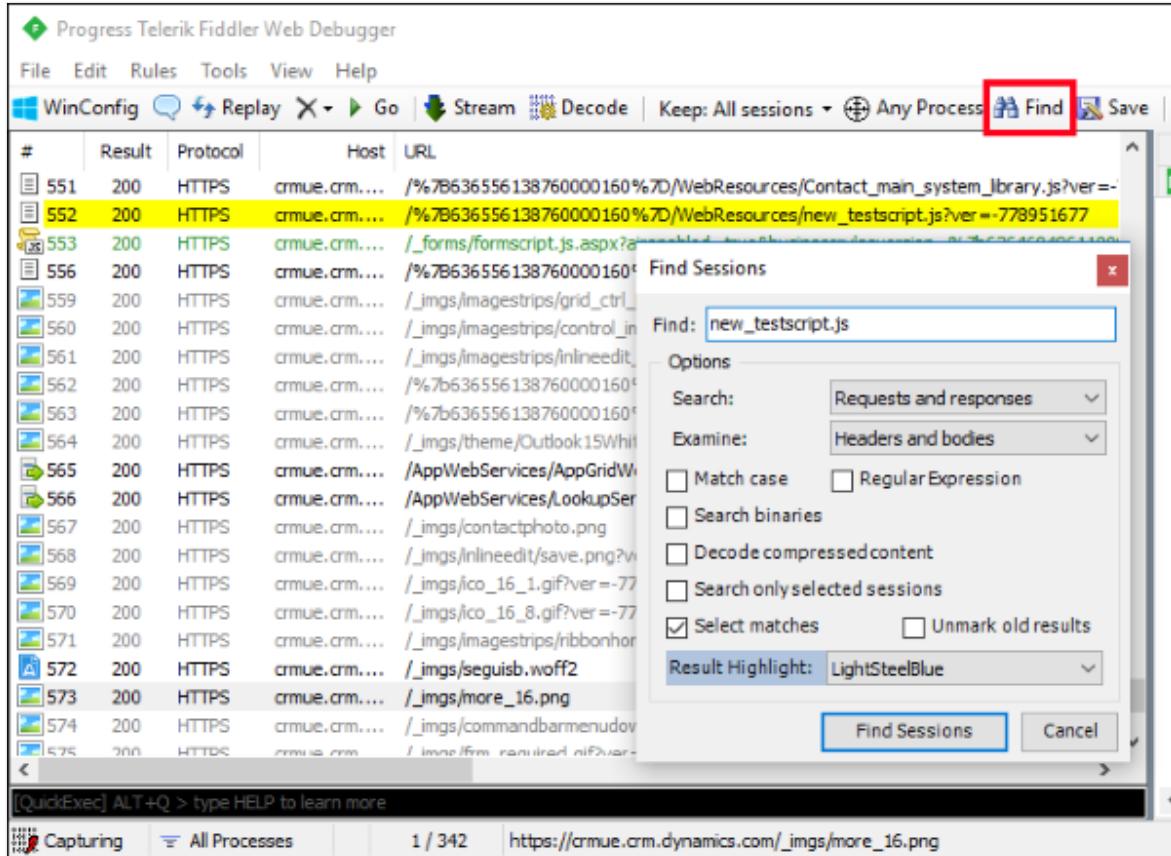
If you want to capture HTTPS traffic only from a particular host, on the **Filters** tab, in the **Hosts** area, in the **-No Host Filter-** drop-down select **Show only the following Hosts** from the menu and enter the list of domains from which you wish to see traffic, separated by semi-colon. More information: [Filters reference](#).



3. Perform any operation necessary to load the script you are testing. You can stop the capture by clicking the same **Capturing** button again.

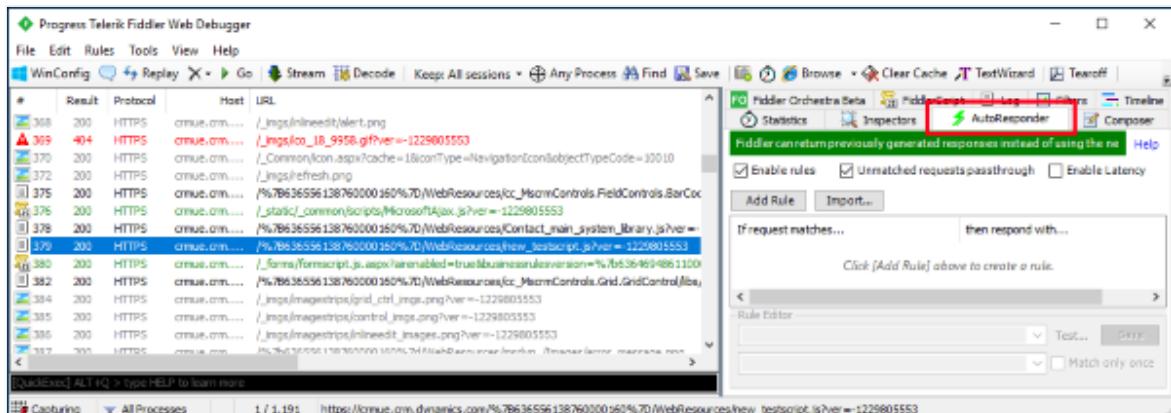
- Select the trace log sessions from the left pane and search for the file you want to setup the AutoResponder for.

For example, if the code you want to debug is in a JavaScript web resource named `new_testscript.js`, use the **Find** button to open the **Find Sessions** dialog box and search for the name of the webresource.

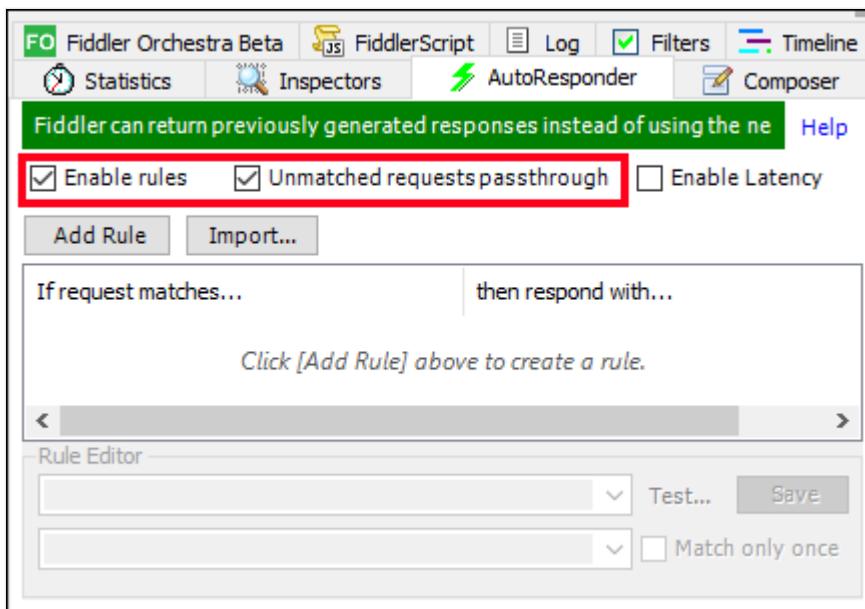


You will see the rows that match with your search criteria highlighted in the left pane.

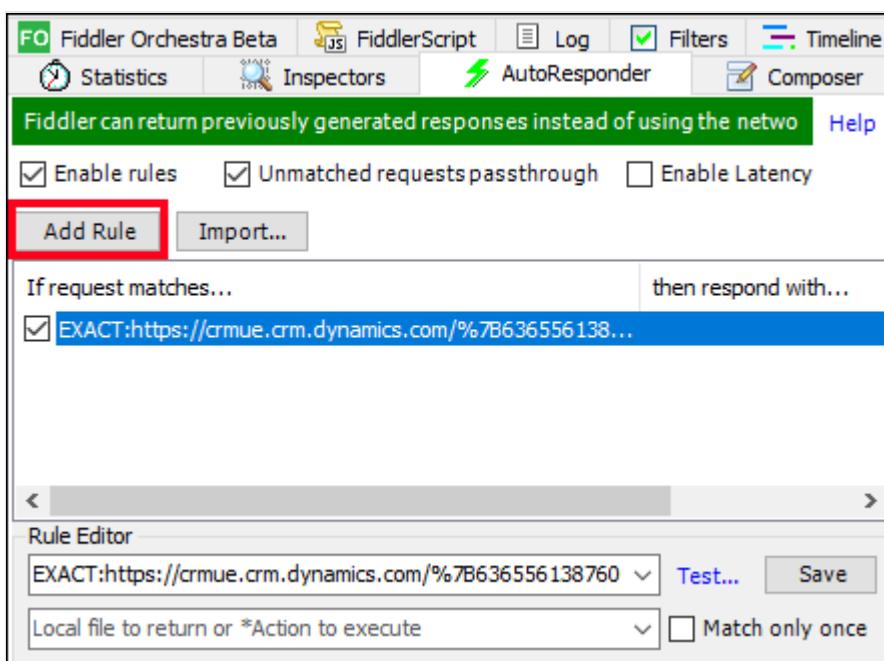
- Select that row. In the right pane, select the **AutoResponder** tab.



- In the AutoResponder tab, select the **Enable rules** and **Unmatched requests passthrough** check boxes.



7. Ensure that you still have the session related to your target file selected and then click on the **Add Rule** button in the **AutoResponder** section. This adds a new entry into the rules table.



8. When the rule is selected, the **Rule Editor** at the bottom has the top row populated with the Session URL related to your file and prefixed with a string like **EXACT:**.

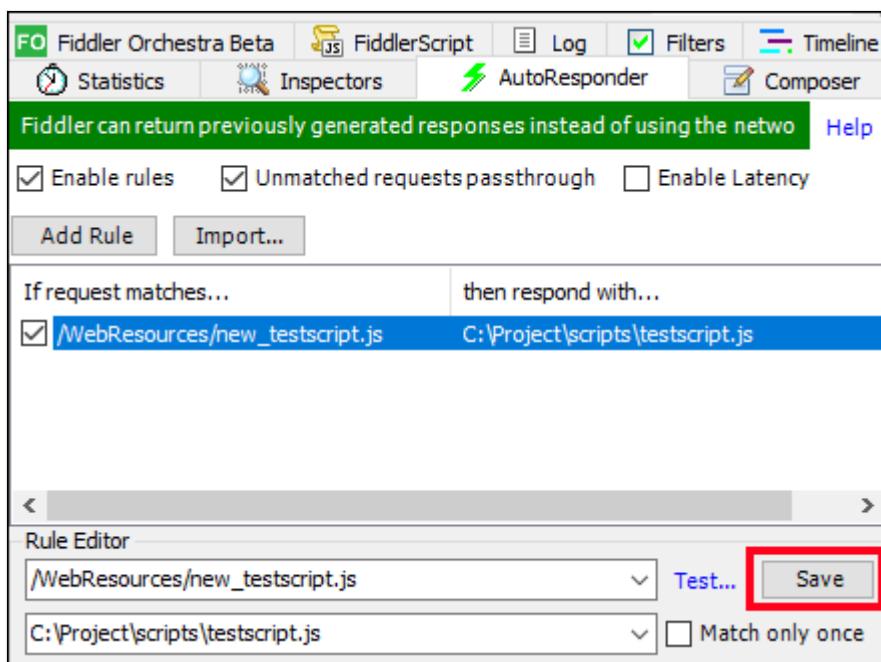
You can then edit the string to match to simplify it. With web resources, the URL will contain generated values in the URL or in a query string to make sure that the latest published version is included in the response. You will probably see the **EXACT** value will look something like this:

```
EXACT:https://<org  
URL>/%7B63655613876000160%7D/WebResources/new_testscript.js?  
ver=-1229805553
```

You can simplify this to remove the generated values and use this instead:

```
/WebResources/new_testscript.js
```

The bottom row is left blank. Type the path to your local file on your disk on this bottom row and **Save**.



By following the steps above, you have configured Fiddler to listen to the requests and responds with the local file instead of passing the request over the network.

Update and test your code

1. Apply changes to your local file.
2. Start Fiddler trace capture again and go back to your browser and hard reload the page with empty cache.
3. In the browser developer tools you can see that the file that is now received will be the local one.
4. Continue repeating this process while updating your code until you get the results you require.

See Also

[Web resources](#)

[Client scripting using JavaScript](#)

CSS web resources

Article • 12/16/2022

Use cascading style sheet (CSS) web resources to create style sheets for use in webpage web resources.

Capabilities of CSS web resources

With CSS web resources, you can manage the appearance of webpage web resources by linking them to a shared library of CSS styles.

Limitations of CSS web resources

Like all web resources, CSS web resources are only available in the security context. Only licensed users who have the necessary privileges can access them.

Referencing a style sheet web resource from a webpage web resource

All web resources can use relative URLs to reference each other. In the following example, for the webpage web resource `sample_/content/contentpage.htm` to reference the style sheet web resource `sample_/styles/styles.css`, add the following example to the head element of `sample_/content/contentpage.htm`:

HTML

```
<link rel="stylesheet" type="text/css" href="../styles/styles.css" />
```

To reference a style sheet from a different publisher, the path must include that solution publisher customization prefix. For example, for the `sample_/content/contentpage.htm` page to reference the `MyIsv_/styles/styles.css` page, the `href` parameter value should be `.../.../MyIsv_/styles/styles.css`.

ⓘ Note

References included in code between web resources aren't tracked as solution dependencies.

See also

[Web resources](#)

[Using Web Page \(HTML\) web resources](#)

[Using JavaScript web resources](#)

[Using Data \(XML\) web resources](#)

[Using Image \(JPG, PNG, GIF\) web resources](#)

[Using Silverlight \(XAP\) web resources](#)

[Using Stylesheet \(XSL\) web resources](#)

[WebResource table](#)

Image web resources

Article • 12/16/2022

Use image web resources to make images available for use in model-driven apps.

There are 5 types of image web resources:

- PNG Format
- JPG Format
- GIF Format
- ICO Format
- Vector Format (SVG)

ⓘ Note

Vector Format (SVG) web resources were added with the model-driven apps.

Capabilities of image web resources

With image web resources you can add images where you need them. Common uses include the following:

- Custom table icons
- Icons for custom ribbon controls and `SiteMap` subareas
- Decorative graphics for forms and webpage web resources.
- Background images that are used by CSS web resources.

Use Vector Format (SVG) web resources for any icon presented in the application. Vector images are defined as Scalable Vector Graphics (SVG) an XML-based vector image format. The advantage of SVG over other image web resources is scale, smaller in size, and if the fill color is removed the model-driven app can control the icon color to avoid contrast issues. When defining the SVG or before uploading it into the web resource please remove any "fill color" attributes. We recommend using SVG over other image types like PNG and JPG. You can define one vector image and re-use it rather than provide multiple sizes of images. You will use these in with a new `EntityMetadata.IconVectorName` property to define the icon for a custom table instead of the `IconLargeName`, `IconMediumName`, or `IconSmallName` properties.

ⓘ Note

Vector Format (SVG) web resources are treated like the Script (JScript) web resources, and carry the same security risks as JavaScript web resources because SVG files allow JScript embedding.

Limitations of image web resources

Like all web resources, image web resources use the security context. Only licensed users who have the necessary privileges can access them.

Reference an image web resource from a webpage web resource

All web resources can use relative URLs to reference each other. In the following example, for the webpage (HTML) web resource new/_content/contentpage.htm to reference the image web resource new/_Images/image1.png, add the following HTML code to new/_content/contentpage.htm:

HTML

```

```

Reference an image web resource from a form

Add an image to a form

1. Navigate to the form editor for a table.
2. Select where you want to add the image in the form.
3. On the **Insert tab**, select **Web Resource**.
4. On the **General** tab, select the web resource image that you want to add.
5. Specify a name for the web resource. You can also specify a label and alternative text.
6. On the **Formatting** tab, you can define:
 - The number of columns the images should use.

- The number of rows the images should use, or if it should automatically expand to use available space.
- The size of the image using the following options:
 - **Stretch to fit**
 - **Stretch to fit (maintain aspect ratio)**
 - **Original**
 - **Specific**
- If you select "Specific," you can enter the desired height and width in pixels.

7. Select **OK**.

8. You must save your changes and publish the form before users can see the image in the form.

Reference an image web resource from a ribbon element or from the Site Map subarea

Use the `$webresource:` directive to specify a web resource image to use as an icon in the ribbon or in the application navigation using Site Map. The following sample shows how to specify icons for a button in the ribbon.

XML

```
<Button Id="MyISV.opportunity.form.actions.FlyoutAnchor.Button.1"
Image16by16="$webresource:new_/icons/oneIcon16.png"
Image32by32="$webresource:new_/icons/oneIcon32.png"/>
```

Note

Using the `$webresource:` directive adds a solution dependency that prevents the referenced image web resources from being deleted as long as they are used by another solution component.

See also

Web resources

- [Using Web Page \(HTML\) web resources](#)
- [Using Style Sheet \(CSS\) web resources](#)
- [Using JavaScript web resources](#)
- [Using Data \(XML\) web resources](#)
- [Using Stylesheet \(XSL\) web resources](#)

Data (XML) Web resources

Article • 12/16/2022

Use Data (XML) Web resources to save and access data.

Capabilities of XML Web resources

Use XML Web resources to cache data that you want to use in your solution.

Limitations of XML Web resources

Use XML Web resources to cache configuration settings or metadata for your solutions.

An XML Web resource does not represent a robust solution for data that is frequently updated by multiple users. While one user is updating an XML Web resource, another user (or automated process) could update the Web resource and that data would be lost when the first user saves their changes.

All XML files must use the .xml file name extension. Files that use XML data but do not use the .xml file name extension cannot be uploaded as Web resources unless the file name extension is changed.

See also

[Web resources](#)

[Using Web Page \(HTML\) web resources](#)

[Using Style Sheet \(CSS\) web resources](#)

[Using JavaScript web resources](#)

[Using Image \(JPG, PNG, GIF\) web resources](#)

[Using Stylesheet \(XSL\) web resources](#)

RESX web resources

Article • 12/16/2022

Use these web resources to manage localized strings in any user interface you define or with error messages you will display.

Using RESX web resources

RESX web resources contain the keys and localized string values for a single language defined using the RESX XML format. RESX is a common format for defining localized resources for windows applications, so there is common tooling available to work with this type of file and localization vendors will be familiar with working with them. When the file is published as a web resource in CRM it will be converted to a JSON format which will be downloaded to the application when needed.

When you create RESX web resources you must explicitly set the language value and include the locale identifier (LCID) for the appropriate language in the name of the web resource. For example, `new_/strings/MyAppResources.1033.resx` would contain resources for English language. See [Microsoft locale ID values](#) for a list of LCID values.

ⓘ Note

If you have multiple RESX web resources with the same name for multiple languages, ensure there is a localized string value for each resource key.

The appropriate string value will be determined by the individual user's language preference and the languages available in the organization. This is done in two steps.

1. Determining the right RESX web resource: If there is a RESX web resource for user's preferred language, that RESX will be used. If a RESX web resource for user's preferred language is not found, then the RESX web resource for base language is chosen.
2. Returning the string value: Within the chosen RESX web resource in Step 1, the string corresponding to the resource key is returned. If the RESX web resource that matches the user's preferred language does not have the resource key, a null response is returned.

For example, `Xrm.Utility.getResourceString("new_/strings/MyAppResources", "hello")` will return the localized string value for the resource key hello within the `new_/strings/MyAppResources.1033.resx` web resource if the user's preferred language is

English. If the user's preferred language is Spanish/Spain, then the localized string value for the resource key hello within the `new_/strings/MyAppResources.1034.resx` web resource is returned. If there is no resource key hello in `new_/strings/MyAppResources.1034.resx` web resource, then a null response is returned. You can see that the function doesn't refer to any specific language or full name of any RESX web resource. This functionality depends on the RESX web resource being associated to the calling JavaScript web resource as a dependency. More information: [Web resource dependencies](#)

See also

[Web resources](#)

[Create accessible web resources](#)

[Web resource dependencies](#)

[Webpage \(HTML\) web resources](#)

[JavaScript web resources](#)

[Image \(JPG, PNG, GIF, ICO\) web resources](#)

[Stylesheet \(XSL\) web resources](#)

[Data \(XML\) Web resources](#)

[CSS web resources](#)

[WebResource table messages and methods](#)

[Sample: Pass multiple values to a web resource through the data parameter](#)

[Sample: Import files as web resources](#)

Stylesheet (XSL) web resources

Article • 12/16/2022

Use Stylesheet (XSL) Web resources to transform XML data.

Capabilities of XSL Web resources

Use XSL Web resources to transform XML data used by your solution.

The following Web resources work together to render a page that displays a table using the data in the XML Web resource. The source files for these Web resources are part of the Import Web Resources sample under the `filestoimport` folder. Download the sample of [Import files as web resources](#).

HTML Web resource: sample/_ImportWebResources/Content>ShowData.htm

HTML

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"https://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html>
  <head>
    <title></title>
    <script src="Script/Script.js" type="text/javascript"></script>
    <link href="CSS/Styles.css" rel="stylesheet" type="text/css" />
  </head>
  <body onload="SDK.ImportWebResources.showData()">
    <div id="results" />
  </body>
</html>
```

XSL Web resource: sample/_ImportWebResources/XSL/Transform.xslt

XML

```
<?xml version="1.0" encoding="utf-8"?>
<xsl:stylesheet version="1.0"
  xmlns:xsl="https://www.w3.org/1999/XSL/Transform"
  xmlns:msxsl="urn:schemas-microsoft-com:xslt"
  exclude-result-prefixes="msxsl"
>
<xsl:output method="xml"
  indent="yes"/>

<xsl:template match="@* | node()">
  <xsl:copy>
    <xsl:apply-templates select="@* | node()"/>
  </xsl:copy>
</xsl:template>
```

```

</xsl:copy>
</xsl:template>

<xsl:template match="people">
<xsl:element name="table">
<xsl:element name="thead">
<xsl:element name="tr">
<xsl:element name="th">
<xsl:text>First Name</xsl:text>
</xsl:element>
<xsl:element name="th">
<xsl:text>Last Name</xsl:text>
</xsl:element>
</xsl:element>
<xsl:element name="tbody">
<xsl:apply-templates />
</xsl:element>
</xsl:element>
</xsl:template>

<xsl:template match="person">
<xsl:element name="tr">
<xsl:element name="td">
<xsl:value-of select="@firstName"/>
</xsl:element>
<xsl:element name="td">
<xsl:value-of select="@lastName"/>
</xsl:element>
</xsl:element>
</xsl:template>

</xsl:stylesheet>

```

XML Web resource: sample_/_ImportWebResources/Data/Data.xml

XML

```

<?xml version="1.0" encoding="utf-8" ?>
<people>
<person firstName="Apurva"
        lastName="Dalia" />
<person firstName="Ofer"
        lastName="Daliot" />
<person firstName="Jim"
        lastName="Daly" />
<person firstName="Ryan"
        lastName="Danner" />
<person firstName="Mike"
        lastName="Danseglio" />

```

```
<person firstName="Alex"
         lastName="Darrow" />
</people>
```

Script Web resource: sample/_ImportWebResources/Script/Script.js

JavaScript

```
//If the SDK namespace object is not defined, create it.
if (typeof SDK == "undefined") {
    SDK = {};
}

// Create Namespace container for functions in this library;
SDK.ImportWebResources = {
    dataFile: "Data/Data.xml",
    transformFile: "XSL/Transform.xslt",
    showData: function () {
        //Create an XML document from the Data.xml file
        var dataXml = new ActiveXObject("Msxml2.DOMDocument.6.0");
        dataXml.async = false;
        dataXml.load(this.dataFile);

        //Create an XML document from the Transform.xslt file
        var transformXSLT = new ActiveXObject("Msxml2.DOMDocument.6.0");
        transformXSLT.async = false;
        transformXSLT.load(this.transformFile);

        // Set the innerHTML of the results area to the output of the
        // transformation.
        var resultsArea = document.getElementById("results");
        resultsArea.innerHTML = dataXml.transformNode(transformXSLT);
    },
};
```

CSS Web resource: sample/_ImportWebResources/CSS/Styles.css

CSS

```
body {
    font-family: Calibri;
}
table {
    border: 1px solid gray;
    border-collapse: collapse;
}
th {
    text-align: left;
    border: 1px solid gray;
}
td {
    border: 1px solid gray;
}
```

See also

[Web Resources](#)

[Using Web Page \(HTML\) Web Resources](#)

[Using Style Sheet \(CSS\) Web Resources](#)

[Using JavaScript Web Resources](#)

[Using Data \(XML\) Web Resources](#)

[Using Image \(JPG, PNG, GIF\) Web Resources](#)

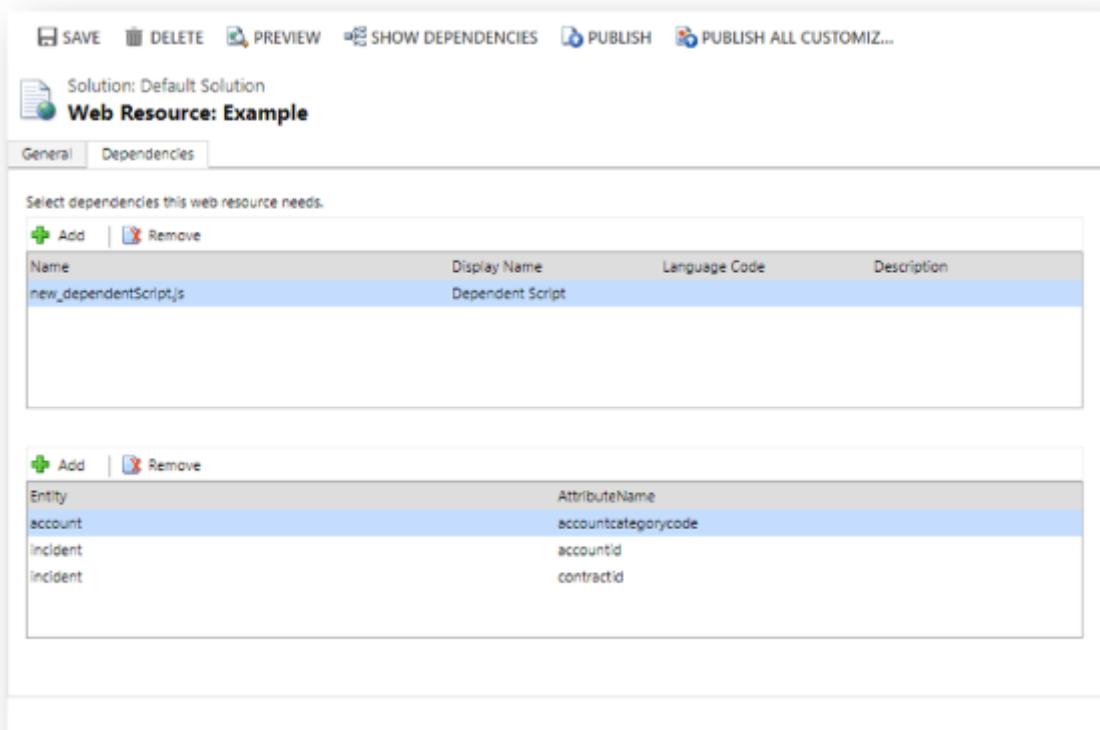
Web resource dependencies

Article • 12/16/2022

You can define dependencies between other web resources. The primary purpose of this feature is to allow association of String (RESX) web resources with the JavaScript web resources that will use them. This is also the way that web resources required by HTML web resources for use offline can be configured to also be available offline.

However there are some other behaviors which developers using JavaScript web resources can take advantage of.

The following image shows the dependencies tab within the web resource form. Dependencies between web resources are set in the top list. Column dependencies are set using the lower list. Column dependencies are only available for JavaScript web resources. More information [Column dependencies](#)



Within a solution you can define dependencies within solution components. Up until Mode-driven Apps the main purpose of these dependencies was to prevent the deletion of a solution component when another solution component depended on it. With Model-driven apps the behavior for JavaScript web resources is enhanced so that any other web resource listed as a dependency to the JavaScript web resource will be loaded along with the JavaScript web resource.

Note

The dependency is only established after it is configured and the web resource is published. Dependencies for unpublished web resources will not take effect until the web resource is published.

The most common scenario is to associate string (RESX) web resources with a JavaScript web resource that depends on it. There will be a String (RESX) web resource for each language that is associated with the JavaScript web resource that uses it. When that JavaScript web resource is loaded, the localized values will also automatically be loaded for the user's preferred language and the organization base language so that they are available for use. Since you should be creating solution dependencies between these resources anyway, you will have the additional benefit of knowing that the dependent RESX resources will be automatically loaded when you need them.

However, web resource dependencies are not limited to just RESX web resources. You can associate a JavaScript web resource to any other type of web resource to create dependencies that will cause the associated web resource to be loaded together with the JavaScript web resource. This will save time because you will not need to explicitly load multiple dependent web resources when you register a script for a [form event](#), [ribbon command](#), or ribbon [enable rules](#), just register the primary script and let the dependency configuration load the rest. You can even create a chain of dependencies because any JavaScript web resources that are loaded because of the primary JavaScript web resource will include any web resources that are associated to them.

Important

Web resource dependencies does not provide any control over the order in which the web resources are loaded. All the web resources are loaded asynchronously and in parallel. If you have a JavaScript web resource which depends on another JavaScript web resource to be loaded and initialized before it can be initialized, you will need to manage that dependency in another way.

Column dependencies

Starting with model-driven apps, if your JavaScript web resource depends on a table column value that you don't want to display in the form, you can set the column as a dependency for the JavaScript web resource. This means that the column will be available within the client API columns collection so you can get or set the value in your code. When you add a dependency this way, the controls collection of the column will be empty because there will be no control on the form.

Before this feature you would need to manually add the column to the form and then configure the control to be hidden. Now you can establish this dependency more directly and eliminate the possibility that someone will remove the hidden column from the form.

See also

[Web resources](#)

[Create accessible web resources](#)

[Webpage \(HTML\) web resources](#)

[JavaScript web resources](#)

[Image \(JPG, PNG, GIF, ICO\) web resources](#)

[Stylesheet \(XSL\) web resources](#)

[Data \(XML\) web resources](#)

[CSS web resources](#)

[RESX web resources](#)

[Web resource table reference](#)

[Sample: Pass multiple values to a web resource through the data parameter](#)

[Sample: Import files as web resources](#)

Create, manage, and publish model-driven apps using code

Article • 11/30/2022

In addition to creating a model-driven app using the Power Apps app designer, you can programmatically create and manage model-driven apps.

Important

You don't have to write code to build model-driven apps if you don't need to! The app designer provides a much simpler and intuitive experience for building model-driven apps without having to write code by providing a tile-based information structure and simplified interface. Check it out here: [Design model-driven apps by using the app designer](#)

Creating a model-driven app involves the following steps:

1. Create an [AppModule table](#) instance to define your app and its properties.
2. Add or remove components to your app such as table, sitemap, and other components for your custom app using the [AddAppComponent](#)s and [RemoveAppComponent](#)s actions.
3. Check your app for any required components that are missing by using the [ValidateApp](#) function.
4. Publish your app.
5. Associate appropriate security roles to your model-driven app to provide access to users.

Create your model-driven app and define its properties

You must have the System Administrator or System Customizer security role or equivalent permissions to be able to create an app.

You must specify the following properties at a minimum to create an app:

- **name**: Unique for your app.
- **uniquename**: This can be different than the name of your app, and can only have English characters and numbers. On creating this app, the name is automatically prefixed with your solution publisher prefix (for example 'new_').

- **webresourceid**: ID of the web resource that you want to be set as the image icon for your app. The system provides you with a default web resource (ID: 953b9fac-1e5e-e611-80d6-00155ded156f) that you can use as an icon for your app.

The following Web API request creates a Unified Interface type of app:

HTTP

```
POST [Organization URI]/api/data/v9.0/appmodules HTTP/1.1
Content-Type: application/json; charset=utf-8
OData-MaxVersion: 4.0
OData-Version: 4.0
Accept: application/json

{
    "name": "SDKTestApp",
    "uniquename": "SDKTestApp",
    "webresourceid": "953b9fac-1e5e-e611-80d6-00155ded156f"
}
```

The response **OData-EntityId** header contains the Uri of the created app.

HTTP

```
HTTP/1.1 204 No Content
OData-Version: 4.0
OData-EntityId: [Organization URI]/api/data/v9.0/appmodules(dd621d4a-d898-e711-80e7-00155db763be)
```

Add or remove components from your model-driven app

You can add or remove components in an app such as sitemap, table, dashboard, business process flows, views, and forms that you want to be included in your model-driven app. For detailed information about components that can be added to a model-driven app, see [Add or edit app components in the app designer](#).

Use the [AddAppComponents](#) action or the [AddAppComponentsRequest](#) message to add components to your model-driven app. The action requires you to specify the following:

- **AppId**: ID of the app where you want to add components.
- **Components** A collection of components to be added. You need to specify the ID and the table type of the component you want to add. For a list of table types in Microsoft Dataverse Web API, see [Web API Entity Type Reference](#).

The following Web API request adds a view (savedquery) and a form (systemform) to your app:

HTTP

```
POST [Organization URI]/api/data/v9.0/AddAppComponents HTTP/1.1
Content-Type: application/json; charset=utf-8
OData-MaxVersion: 4.0
OData-Version: 4.0
Accept: application/json

{
    "AppId": "dd621d4a-d898-e711-80e7-00155db763be",
    "Components": [
        {
            "savedqueryid": "00000000-0000-0000-00aa-000000666000",
            "@odata.type": "Microsoft.Dynamics.CRM.savedquery"
        },
        {
            "formid": "c9e7ec2d-efca-4e4c-b3e3-f63c4bba5e4b",
            "@odata.type": "Microsoft.Dynamics.CRM.systemform"
        }
    ]
}
```

To remove a component from an app, use the [RemoveAppComponents](#) action or the [RemoveAppComponentsRequest](#) message. This action takes the same set of parameters as the [AddAppComponents](#) action.

The following Web API request removes a view (savedquery) from your app:

HTTP

```
POST [Organization URI]/api/data/v9.0/RemoveAppComponents HTTP/1.1
Content-Type: application/json; charset=utf-8
OData-MaxVersion: 4.0
OData-Version: 4.0
Accept: application/json

{
    "AppId": "dd621d4a-d898-e711-80e7-00155db763be",
    "Components": [
        {
            "savedqueryid": "00000000-0000-0000-00aa-000000666000",
            "@odata.type": "Microsoft.Dynamics.CRM.savedquery"
        }
    ]
}
```

Validate your model-driven app

Validating an app involves checking for any dependencies for the components you have added in your model-driven app to ensure that your app works fine. This is the same as selecting **Validate** in the app designer. More information: [Validate your app](#)

Use the [ValidateApp](#) function or the [ValidateAppRequest](#) message to validate your app. The following Web API request shows how to validate your model-driven app with ID: dd621d4a-d898-e711-80e7-00155db763be:

```
GET [Organization URI]/api/data/v9.0/ValidateApp(AppModuleId=dd621d4a-d898-e711-80e7-00155db763be)
```

If there are no validation errors, the response is as follows:

```
HTTP  
  
HTTP/1.1 200 OK  
OData-Version: 4.0  
  
{  
    "@odata.context": "[Organization  
URI]/api/data/v9.0/$metadata#Microsoft.Dynamics.CRM.ValidateAppResponse",  
    "AppValidationResponse": {  
        "ValidationSuccess": true,  
        "ValidationIssueList": []  
    }  
}
```

If there are validation issues in your app, the response displays errors/warnings in the **ValidationIssueList** collection:

```
HTTP  
  
HTTP/1.1 200 OK  
OData-Version: 4.0  
  
{  
    "@odata.context": "[Organization  
URI]/api/data/v9.0/$metadata#Microsoft.Dynamics.CRM.ValidateAppResponse",  
    "AppValidationResponse": {  
        "ValidationSuccess": false,  
        "ValidationIssueList": [  
            {  
                "ErrorType": "Error",  
                "Message": "App does not contain Site Map",  
                "DisplayName": null,  
                "ComponentId": "00000000-0000-0000-0000-000000000000",  
                "ComponentType": 0,  
                "ValidationIssueType": 1  
            }  
        ]  
    }  
}
```

```

        "ComponentSubType": 0,
        "ParentEntityId": "00000000-0000-0000-0000-000000000000",
        "ParentEntityName": null,
        "CRMErrorCode": -2147155684,
        "RequiredComponents": []
    },
    {
        "ErrorType": "Warning",
        "Message": "Account doesn't reference a form or view. App users will see all forms and views.",
        "DisplayName": null,
        "ComponentId": "00000000-0000-0000-0000-000000000000",
        "ComponentType": 0,
        "ComponentSubType": 0,
        "ParentEntityId": "00000000-0000-0000-0000-000000000000",
        "ParentEntityName": null,
        "CRMErrorCode": -2147155691,
        "RequiredComponents": []
    }
]
}

```

Publish your model-driven app

After you have added required components to your model-driven app and validated it, you must publish it to make it available to users.

Use the [PublishXml](#) action or the [PublishXmlRequest](#) message to publish your model-driven app. The following request shows how to publish your model-driven app with ID: dd621d4a-d898-e711-80e7-00155db763be:

HTTP

```

POST [Organization URI]/api/data/v9.0/PublishXml HTTP/1.1
Content-Type: application/json; charset=utf-8
OData-MaxVersion: 4.0
OData-Version: 4.0
Accept: application/json

{
    "ParameterXml": "<importexportxml><appmodules><appmodule>dd621d4a-d898-e711-80e7-00155db763be</appmodule></appmodules></importexportxml>"
}

```

Manage access to model-driven app using security roles

To provide users access to your apps so that they can access it from their **Settings > My Apps** area or the home page, you can associate security roles to your model-driven apps. Users assigned to the associated security roles and can see and use your model-driven apps in Dataverse.

Use the **appmoduleroles_association** navigation property of the [AppModule table](#) entity to associate a model-driven app with a security role. The following request shows how to associate a model-driven app with a security role:

HTTP

```
POST [Organization URI]/api/data/v9.0/appmodules(dd621d4a-d898-e711-80e7-00155db763be)/appmoduleroles_association/$ref HTTP/1.1
Content-Type: application/json; charset=utf-8
OData-MaxVersion: 4.0
OData-Version: 4.0
Accept: application/json

{
  "@odata.id": "[Organization URI]/api/data/v9.0/roles(<roleId>)"
}
```

To disassociate a security role from a model-driven app, you use the DELETE request with the same navigation property. For example:

```
DELETE [Organization URI]/api/data/v9.0/appmodules(dd621d4a-d898-e711-80e7-00155db763be)/appmoduleroles_association/$ref?$id=[Organization URI]/api/data/v9.0/roles(<roleId>)
```

Manage your model-driven apps and its components

This section provides you information about retrieving your apps, updating app properties, retrieving app components, and deleting apps.

Retrieve published apps

To retrieve published apps, use the following request:

```
GET [Organization URI]/api/data/v9.0/appmodules?$select=name,clienttype
```

Retrieve unpublished apps

To retrieve unpublished apps, use the [RetrieveUnpublishedMultiple](#) function. For example:

```
GET [Organization  
URI]/api/data/v9.0/appmodules/Microsoft.Dynamics.CRM.RetrieveUnpublishedMultiple()  
$select=name,clienttype
```

Retrieve components in a published model-driven app

To retrieve app components for a model-driven app, use the [RetrieveAppComponents](#) function or the [RetrieveAppComponentsRequest](#) message. For example:

```
GET [Organization URI]/api/data/v9.0/RetrieveAppComponents(AppModuleId=dd621d4a-  
d898-e711-80e7-00155db763be)
```

Retrieve security roles associated with published model-driven app

To retrieve the security roles associated with your model-driven app, use the `$expand` system query option with the `appmoduleroles_association` navigation property. For example, here is the request to retrieve all the security roles associated to a model-driven app with ID: dd621d4a-d898-e711-80e7-00155db763be:

```
GET [Organization URI]/api/data/v9.0/appmodules(dd621d4a-d898-e711-80e7-  
00155db763be)?  
$expand=appmoduleroles_association&$select=name,appmoduleroles_association
```

Delete model-driven apps

Use the DELETE request to delete a model-driven app. For example:

```
DELETE [Organization URI]/api/data/v9.0/appmodules(dd621d4a-d898-e711-80e7-  
00155db763be)
```

Client API support for model-driven apps

You can use the following client APIs to work with model-driven apps:

- [getCurrentAppName](#)

- [getCurrentAppProperties](#)
- [getCurrentAppUrl](#)

See also

[Design model-driven apps by using the app designer](#)

When to edit the customizations file

Article • 11/30/2022

The customizations.xml file that is exported as part of an unmanaged solution can be edited to perform specific customization tasks. After editing the file you can compress the modified file together with the other files exported in the unmanaged solution. You apply the changes by importing that modified unmanaged solution.

Editing a complex XML file like the customizations.xml file is much easier and less prone to errors if you use a program designed to support schema validation. While it is possible to edit this file using a simple text editor like Notepad, this is not recommended unless you are very familiar with editing this file. For more information, see [Edit the Customizations file with Schema Validation](#).

ⓘ Important

Invalid XML or incorrect definition of solution components can cause errors that will prevent importing a manually edited unmanaged solution.

Supported tasks

You can edit the customization.xml file to perform the following tasks.

Editing the ribbon

This documentation describes the process of editing the ribbon by editing the customization.xml file directly. Several people have created ribbon editors that provide a user interface to make editing the ribbon easier. The most popular one so far is the [Ribbon Workbench](#). For support using this program, contact the program publisher.

For more information about editing the ribbon by editing the customization.xml manually, see [Customize the Ribbon](#).

Editing the SiteMap

The SDK describes the process of editing the SiteMap by editing the customization.xml file directly. However, its recommended that you use the site map designer in Model-driven apps to create or update site maps. More information: [Tutorial: Create a Model-driven app site map for an app using the site map designer](#)

You can also use one of the community-developed site map editors, such as the [XrmToolBox Site Map Editor](#).

For more information, see [Change Application Navigation using the SiteMap](#)

Editing FormXml

FormXml is used to define forms and dashboards. The form editor and dashboard designer in the application are the most commonly used tools for this purpose. Editing the customizations.xml file is an alternative method. For more information, see [Customize forms](#) and [Create a Dashboard](#).

Editing saved queries

Definitions of views for tables are included in the customizations.xml file and may be manually edited. The view editor in the application is the most commonly used tool for this purpose. Editing customizations.xml is an alternative method. For more information, see [Customize views](#).

Editing the ISV.config

For Microsoft Dataverse, the Ribbon provides the way to extend the application. The only remaining capability left in ISV.Config is to customize the appearance of the Service Calendar. For more information, see [Service calendar appearance configuration](#).

Unsupported tasks

Defining any other solution components by editing the exported customizations.xml file is not supported. This includes the following:

- Tables
- Columns
- Table Relationships
- Table Messages
- Choice
- Web Resources
- Processes (Workflows)
- Plugin Assemblies
- SDK Message Processing steps
- Service Endpoints
- Reports

- Connection Roles
- Article Templates
- Contract Templates
- E-mail Templates
- Mail Merge Templates
- Security Roles
- Field Security Profiles

See also

[Customization XML reference](#)

[Customization solutions file schema](#)

[Ribbon core schema](#) [Ribbon types schema](#) [Ribbon WSS schema](#)

[Form XML schema](#)

[Schema Support for the Customization File](#)

Edit the customizations XML file with schema validation

Article • 11/30/2022

The `customizations.xml` file is included within the compressed `.zip` file exported as a solution. Certain portions of the `customizations.xml` file can be edited manually.

Information about the schema helps you confirm that any modifications you make are valid.

XSD schema files

[Download the schemas ↗](#) for the XSD schema files used to validate the `customization.xml` file in a solution. The necessary files are:

- `CustomizationsSolution.xsd`
- `fetch.xsd`
- `FormXml.xsd`
- `isv.config.xsd`
- `RibbonCore.xsd`
- `RibbonTypes.xsd`
- `RibbonWSS.xsd`
- `SiteMap.xsd`
- `SiteMapType.xsd`
- `VisualizationDataDescription.xsd`

Using schema validation

Because the exported XML is a text file, you can edit it using a text editor such as Notepad. However, we strongly recommend that you use an application that supports XSD schema validation such as Visual Studio. XSD validation in Visual Studio provides IntelliSense information and schema validation to help prevent errors.

The XSD schema files that are used to validate the customization.xml file in a solution are available here. [Download the schemas](#). Make sure to copy all the files from that folder into the same directory. You will need to associate the customizations.xml file to the CustomizationsSolution.xsd file. That file has links to all the other XSD files in the folder.

1. Download the XSD schema files and copy all of them to your computer. Save them in the location that Visual Studio uses for XSD validation files. This location is probably `[install drive]\Program Files (x86)\Microsoft Visual Studio X.0\Xml\Schemas` where `X` represents the version of Visual Studio.
2. Right-click the customizations.xml file and select **Open With** and then select the version of Visual Studio.
3. Select **View** and then select **Properties Window**.
4. In the **Properties** window, in the **Schemas** column, click the ellipsis [...] button.
5. In the **Xml Schemas** dialog box you should see the customizationsSolution.xsd. In the **Use** column, select **Use this schema**.

 **Note**

If you do not see it, click **Add** and browse to where you saved it.

6. Click **OK**.

You are now ready to begin editing the XML with XSD validation.

See also

- [When to edit the customizations file for Microsoft Dataverse](#)
- [Ribbon core schema](#)
- [Ribbon types schema](#)
- [Ribbon WSS schema](#)
- [Form XML schema](#)
- [ISV configuration file schema](#)

Publish customizations

Article • 11/30/2022

Publishing customizations makes the web application aware of changes to the data that affects the user interface.

When to publish customizations

Customizations are automatically published when new items are created or existing items are deleted. You must publish changes after updating table definitions or tables that affect the user interface. You can decide to wait and publish a set of related changes together. Only published customizations are exported with a solution. You should always publish customizations before exporting a solution.

When you perform customizations that will appear in Dynamics 365 for tablets, you should always explicitly publish your customizations to make sure that every item is synchronized with the Dynamics 365 for tablets application.

 **Note**

Publishing customizations can interfere with normal system operation. In a production environment, we recommend that you schedule publishing customizations when it's least disruptive to users.

Publishing programmatically

The following table lists the two messages that you can use to publish customizations.

Message	Description
PublishAllXmlRequest	Publishes all customizations.
PublishXmlRequest	Publishes the specified customizations.

When you use the `PublishXmlRequest` message, you specify which items you want to publish by using the `ParameterXml` parameter. `ParameterXML` must comply with the [Publish Request Schema](#).

Retrieving unpublished metadata

If you want to create an application to edit customizable items in model-driven apps, you must retrieve any unpublished definitions of those items. If a developer defines some changes but does not publish them, your application must be able to retrieve them to display them in the user interface.

Use the following two methods to retrieve unpublished metadata:

RetrieveAsIfPublished parameter

Retrieves table, column, table relationship, and choice data by using the following messages:

- [RetrieveAllEntitiesRequest](#)
- [RetrieveAllOptionSetsRequest](#)
- [RetrieveAttributeRequest](#)
- [RetrieveEntityRequest](#)
- [RetrieveOptionSetRequest](#)
- [RetrieveRelationshipRequest](#)

RetrieveUnpublished Request

Retrieves user interface items, such as form, template, visualization and Web resource definitions, by using the following messages:

- [RetrieveUnpublishedRequest](#)
- [RetrieveUnpublishedMultipleRequest](#)

See also

[Extend table definitions model](#)

[Publish request schema](#)

[Customize forms](#)

[Customize views](#)

[Change application navigation using the SiteMap](#)

Best practices and guidance for model-driven apps

Article • 12/16/2022

Model-driven apps is a component-focused approach to app development which can be extended by a developer to achieve a much more tailored experience. While customizing model-driven apps, a developer should be aware of the established guidance and best practices.

Within this section you will learn about the issues we have identified, their impact, and understand the guidance to resolve them. We will explain the background about why things should be done in a certain way and avoid potential problems in the future. This can benefit the usability, supportability, and performance of your environment. The guidance documentation supports the existing information within the Developer and Administration guides.

ⓘ Note

Right now, only client scripting best practices are documented in the TOC and rest of them will be added eventually. This document covers the overall structure of how the best practice page should look with the sections and guidelines.

Targeted customization types

The documentation targets the following customization types:

- Model-driven app design
- Form design
- Client scripting
- Web resources

Sections

Each guidance article includes most or all of the following sections:

- Title - description of the guidance
- Category - one or more areas impacted by not following the guidance
- Impact potential - the level of risk (high, medium, or low) of affecting the environment by not following the guidance

- Symptoms - possible indications that the guidance has not been followed
- Guidance - recommendations that may also include examples
- Problematic patterns - description or examples of not following the guidance
- Additional information - supporting details for a more extensive view
- See also - references to learn more about something mentioned in the article

Categories

Each guidance article is classified with one or more of the following categories:

- Usage – improper usage of a particular API, pattern, or configuration
- Design – design flaws in a customization
- Performance – customization or pattern that may produce a negative effect on performance in areas such as memory management, CPU utilization, network traffic, or user experience
- Security – potential vulnerabilities in a customization that could be exploited in a runtime environment
- Upgrade Readiness - customization or pattern that may increase risk of having an unsuccessful version upgrade
- Online Migration - customization or pattern that may increase risk of having an unsuccessful online migration
- Maintainability – customization that unnecessarily increases the amount of developer effort required to make changes, the frequency of required changes, or the chance of introducing regressions
- Supportability – customization or pattern that falls outside the boundaries of published supportability statements, including usage of removed APIs or implementation of forbidden techniques

Sample: Export ribbon definitions

Article • 08/22/2023

This sample shows how to export ribbon definitions. It uses the Organization Service `RetrieveApplicationRibbonRequest` and `RetrieveEntityRibbonRequest` classes.

[SDK for .NET: Export ribbon definitions sample code](#)

How to run this sample

1. Download or clone the [Samples](#) repo so that you have a local copy.
2. (Optional) Edit the dataverse/App.config file to define a connection string specifying the Microsoft Dataverse instance you want to connect to.
3. Open the sample solution in Visual Studio and press **F5** to run the sample. After you specify a connection string in dataverse/App.config, any sample you run will use that connection information.

If you do not specify a connection string in dataverse/App.config file, a dialog will open each time you run the sample and you will need to enter information about which Dataverse instance you want to connect to and which credentials you want to use. This dialog will cache previous connections so that you can choose a previously used connection.

Those samples in this repo that require a connection to a Dataverse instance to run will include a linked reference to the dataverse/App.config file.

What this sample does

The `RetrieveApplicationRibbon` message is intended to be used in a scenario where it contains data that is needed to retrieve the data that defines the content and behavior of the application ribbon. The `RetrieveEntityRibbon` message is intended to be used in a scenario where it contains data that is needed to retrieve ribbon definitions for a table.

How this sample works

In order to simulate the scenario described in [What this sample does](#), the sample will do the following:

Setup

Checks for the current version of the org.

Demonstrate

1. The `RetrieveApplicationRibbon` message retrieves the application ribbon.
2. The `RetrieveEntityRibbon` message retrieves the system table ribbons

Clean up

No clean up is required for this sample

See also

[Customize commands and the ribbon](#)

[Pass parameters to a URL by using the ribbon](#)

[Ribbon core schema](#)

[Ribbon types schema](#)

[Ribbon WSS schema](#)

[RetrieveApplicationRibbonRequest](#)

[RetrieveEntityRibbonRequest](#)

Sample: Pass multiple values to a web resource through the data parameter

Article • 11/30/2022

An (HTML) web resource page can only accept a single custom parameter called `data`. To pass more than one value in the data parameter, you need to encode the parameters and decode the parameters in your page.

The page here represents a technique to pass the additional values within a single parameter and then process them within your web resource.

Sample HTML web resource

The HTML code below represents a webpage (HTML) web resource that includes a script that defines three functions:

- **getDataParam**: Called from the `body.onload` event, this function retrieves any query string parameters passed to the page and locates one named `data`.
- **parseDataValue**: Receives the data parameter from `getDataParam` and builds a DHTML table to display any values passed within the `data` parameter.

ⓘ Note

All characters included in the query string will be encoded using the [encodeURIComponent function](#). This function uses the JavaScript [decodeURIComponent function](#) to decode the values passed.

- **noParams**: Displays a message when no parameters are passed to the page.

HTML

```
<!DOCTYPE html>
<html lang="en-us">
<head>
  <title>Show Data Parameters Page</title>
  <style type="text/css">
    body
    {
      font-family: Segoe UI, Tahoma, Arial;
      background-color: #d6e8ff;
    }
  </style>
</head>
<body>
  <h1>Show Data Parameters Page</h1>
  <table border="1">
    <tr>
      <td>Name</td>
      <td>Value</td>
    </tr>
  </table>
</body>
</html>
```

```

tbody
{
  background-color: white;
}
th
{
  background-color: black;
  color: White;
}
</style>
<script type="text/javascript">
document.onreadystatechange = function () {
  if (document.readyState == "complete") {
    getDataParam();
  }
}

function getDataParam() {
  //Get the any query string parameters and load them
  //into the vals array

  var vals = new Array();
  if (location.search != "") {
    vals = location.search.substr(1).split("&");
    for (var i in vals) {
      vals[i] = vals[i].replace(/\+/g, " ").split("=");
    }
    //look for the parameter named 'data'
    var found = false;
    for (var i in vals) {
      if (vals[i][0].toLowerCase() == "data") {
        parseDataValue(vals[i][1]);
        found = true;
        break;
      }
    }
    if (!found)
    { noParams(); }
  }
  else {
    noParams();
  }
}

function parseDataValue(datavalue) {
  if (datavalue != "") {
    var vals = new Array();

    var message = document.createElement("p");
    setText(message, "These are the data parameters values that were passed
to this page:");
    document.body.appendChild(message);

    vals = decodeURIComponent(datavalue).split("&");
    for (var i in vals) {

```

```

        vals[i] = vals[i].replace(/\+/g, " ").split("=");
    }

    //Create a table and header using the DOM
    var oTable = document.createElement("table");
    var othead = document.createElement("thead");
    var otheadTR = document.createElement("tr");
    var otheadTRTH1 = document.createElement("th");
    setText(otheadTRTH1, "Parameter");
    var otheadTRTH2 = document.createElement("th");
    setText(otheadTRTH2, "Value");
    otheadTR.appendChild(otheadTRTH1);
    otheadTR.appendChild(otheadTRTH2);
    othead.appendChild(otheadTR);
    oTable.appendChild(thead);
    var otbody = document.createElement("tbody");
    //Loop through vals and create rows for the table
    for (var i in vals) {
        var otrRow = document.createElement("tr");
        var otrRowTD1 = document.createElement("td");
        setText(otrRowTD1, vals[i][0]);
        var otrRowTD2 = document.createElement("td");
        setText(otrRowTD2, vals[i][1]);

        otrRow.appendChild(otrRowTD1);
        otrRow.appendChild(otrRowTD2);
        otbody.appendChild(otrRow);
    }

    oTable.appendChild(otbody);
    document.body.appendChild(oTable);
}
else {
    noParams();
}
}

function noParams() {
    var message = document.createElement("p");
    setText(message, "No data parameter was passed to this page");

    document.body.appendChild(message);
}
//Added for cross browser support.
function setText(element, text) {
    if (typeof element.innerText != "undefined") {
        element.innerText = text;
    }
    else {
        element.textContent = text;
    }

}
</script>
</head>
```

```
<body>  
</body>  
</html>
```

Using this page

1. Create a webpage web resource called "new/_ShowDataParams.htm" using the sample code.

The parameters you want to pass are: `first=First Value&second=Second Value&third=Third Value`

⚠ Note

If you're adding static parameters using the Web Resource Properties dialog box from the form editor, you can simply paste the parameters without encoding them into the **Custom Parameter(data)** column. These values will be encoded for you, but you'll still need to decode them and extract the values in your page.

2. For dynamic values generated in code, use the `encodeURIComponent` method on the parameters. The encoded values should be:

`first%3DFirst%20Value%26second%3DSecond%20Value%26third%3DThird%20Value`

Open the page passing the encoded parameters as the value of the data parameter:

```
https://<server name>/WebResources/new/_ShowDataParams.htm?  
data=first%3DFirst%20Value%26second%3DSecond%20Value%26third%3DThird%20  
Value
```

⚠ Note

If you have added the web resource to a form and have pasted the un-encoded parameters into the **Custom Parameters(data)** column, you can just preview the form.

3. The `new/_ShowDataParams.htm` will display a dynamically generated table:

Parameter	Value
first	First Value
second	Second Value
third	Third Value

How it works

To access the values embedded within the data query string parameter value, in your webpage web resource you can extract the value of the data parameter and then use code to split the string into an array so you can access each name value pair individually.

When the page loads the `getDataParam` function is called. This function simply identifies the data parameter and passes the value to the `ParseDataValue` function. If no data parameter is found the `noParams` function will add a message to the page in place of the table.

The `ParseDataValue` function uses similar logic found in `getDataParam` to locate the custom parameter delimiters to create an array of name value pairs. Then it generates a table and appends it to the otherwise empty document.body.

See also

[Web Resources](#)

[Sample: Import Files as Web Resources](#)

[Web Page \(HTML\) Web Resources](#)

Sample: Import files as web resources

Article • 11/30/2022

When you develop a large number of files to use as Web resources you can save yourself the work of manually adding them through the application. Many Web resources can be developed and tested outside of Model-driven apps and then imported.

This sample provides a simplified example of this process.

Download the sample: [Import files as web resources ↗](#).

Prerequisites

Internet connection is required to download the sample project and to restore the NuGet packages used in the sample project.

Requirements

The sample code included in the SDK download package includes the following files required by this sample:

ImportJob.xml

This file provides data about the Web Resource records that will be created. For each file it contains the following data:

- **path:** The path to each file from the FilesToImport folder.
- **displayName:** The display name for the Web resource.
- **description:** A description of what each file does.
- **name:** The name that will be used for the Web Resource.

(!) Note

- Each of these names begin with an underscore character. The customization prefix of the solution publisher will be prepended to the name when the Web resource is created. Rather than hard-coding a specific customization prefix, this sample will detect the current

customization prefix for a publisher record that may already exist in the organization.

- Because each of these files was developed outside of Model-driven apps and depend on relative paths to access each other, the names include backslash "/" characters to create a virtual folder structure so the relative links will continue to function within Model-driven apps.

- **type:** Specifies the type of Web Resource to create using the integer values found in [Web Resource Types](#).

FilesToImport/ShowData.htm

This HTML Web resource requires each of the other files to display the following table.

First Name	Last Name
Apurva	Dalia
Ofer	Daliot
Jim	Daly
Ryan	Danner
Mike	Danseglio
Alex	Darrow

FilesToImport/CSS/Styles.css

This file provides the CSS Styles used in ShowData.htm.

FilesToImport/Data/Data.xml

This file contains the list of names that is displayed in the table.

FilesToImport/Script/Script.js

This file contains a JScript library that contains information about the relative location of the Data.xml file and the Transform.xslt file. It also contains the `showData` function that transforms the data and adds it to the ShowData.htm page.

FilesToImport/XSL/Transform.xslt

This file contains the XSL definition of how to transform the data into an HTML table.

Demonstrates

Creating Web Resources in the Context of a Solution

Web Resources are organization-owned records so they can be created using either the `IOrganizationService.Create` method or by using the `CreateRequest` message and the `IOrganizationService.Execute` method. This sample will show how to use the `SolutionUniqueName` optional parameter to associate a Web resource with a specific solution when it is created. This requires using the `CreateRequest` message.

Uploading Files from Disk

The `WebResource.Content` property requires a Base 64 string representing the binary contents of the file. The following sample is the method used to convert the file into the required type.

C#

```
//Encodes the Web Resource File
static public string getEncodedFileContents(String pathToFile)
{
    FileStream fs = new FileStream(pathToFile, FileMode.Open,
FileAccess.Read);
    byte[] binaryData = new byte[fs.Length];
    long bytesRead = fs.Read(binaryData, 0, (int)fs.Length);
    fs.Close();
    return System.Convert.ToBase64String(binaryData, 0, binaryData.Length);
}
```

Combining Web Resource Record Data with File Data

The `ImportJob.xml` file demonstrates how the data about the files being imported and the data about the Web Resource to create are combined. In particular, in order for relative links between related files to continue to function, the name of the Web resources you create must preserve information about the relative position of the files on disk using simulated directories in the file name. Because of the data in the `ImportJob.xml` file all these related Web resource files will be created under a common virtual folder.

Note

It is not necessary to publish Web resources when they are created. It is necessary to publish them when they are updated.

Example

The following portion of the ImportWebResources.cs file expects the following variables:

- `_customizationPrefix` : The customization prefix of the **SDK Samples** publisher. If this publisher does not exist it will be created with the customization prefix of "sample".
- `_ImportWebResourcesSolutionUniqueName` : The unique name of the **Import Web Resources Sample Solution** created in this sample. The value is `ImportWebResourcesSample`.

C#

```
//Read the descriptive data from the XML file
XDocument xmlDoc = XDocument.Load("../ImportJob.xml");

//Create a collection of anonymous type references to each of the Web
Resources
var webResources = from webResource in xmlDoc.Descendants("webResource")
                   select new
                   {
                       path = webResource.Element("path").Value,
                       displayName =
webResource.Element("displayName").Value,
                       description =
webResource.Element("description").Value,
                       name = webResource.Element("name").Value,
                       type = webResource.Element("type").Value
                   };

// Loop through the collection creating Web Resources
int counter = 0;
foreach (var webResource in webResources)
{
    //Set the Web Resource properties
    WebResource wr = new WebResource
    {
        Content = getEncodedFileContents(@"../../" + webResource.path),
        DisplayName = webResource.displayName,
        Description = webResource.description,
        Name = _customizationPrefix + webResource.name,
        LogicalName = WebResource.EntityLogicalName,
        WebResourceType = new OptionSetValue(Int32.Parse(webResource.type))
    };

    // Using CreateRequest because we want to add an optional parameter
    CreateRequest cr = new CreateRequest
    {
        Target = wr
    };
    //Set the SolutionUniqueName optional parameter so the Web Resources
```

```
will be
    // created in the context of a specific solution.
    cr.Parameters.Add("SolutionUniqueName",
_IImportWebResourcesSolutionUniqueName);

    CreateResponse cresp = (CreateResponse)_serviceProxy.Execute(cr);
    // Capture the id values for the Web Resources so the sample can delete
    them.
    _webResourceIds[counter] = cresp.id;
    counter++;
    Console.WriteLine("Created Web Resource: {0}", webResource.displayName);
}
```

It is not necessary to publish Web resources when they are created. It is necessary to publish them when they are updated.

See also

[Web resource table reference](#)

[Web resources](#)

Developer tools

Article • 09/05/2022

The Microsoft Dataverse community creates tools! Many of the most popular ones are distributed via the [XrmToolBox](#). XrmToolBox is a Windows application that connects to Dataverse, providing tools to ease customization, configuration and operation tasks. It is shipped with more than 30 plugins to make administration, customization or configuration tasks easier and less time consuming.

The following is a selected list of community tools distributed via the XrmToolBox you can use when working with model-driven apps.

Tool	Description
FetchXML Builder	Build queries for Microsoft Dataverse, Dynamics 365 and the Power Platform. Investigate data. Get code.
Easy Translator	Exports and Imports translations with contextual information.
Export to Excel	Easily export records from the selected view/fetchxml to Excel.
Iconator	Manage custom tables icons in a single screen.
Ribbon Workbench 2016	Edit the ribbon or command bar from inside the XrmToolbox.
View Designer	Easy UI to design view layouts and alter queries using FetchXML builder.
View Layout Replicator	Apply same layout to multiple views of the same table in a single operation.
WebResources Manager	Manage your web resources easily

Another tool that is not distributed via the XrmToolBox is the [Dataverse REST Builder](#). This tool generates JavaScript code for use with the Web API.

ⓘ Important

Tools created by the community are not supported by Microsoft. If you have questions or issues with community tools, contact the publisher of the tool.

Testing tools for client-side development

Article • 12/16/2022

Microsoft provides an automated UI testing framework specifically for model-driven apps called [Easy Repro](#). This framework is built using the [SeleniumHQ](#) browser automation open-source project.

Easy Repro provides a set of classes and methods to work with various pages in model-driven apps so you don't need to parse the HTML elements of the application when writing test cases. This makes your tests resilient to changes made in the HTML elements that compose the application pages.

Benefits of unit testing

Unit testing is strongly recommended but not required. When you are just getting started or if the amount of code in your solution is relatively small, you may feel that you spend more time writing tests than the functionality included in your solution.

The benefits of unit testing begin to accrue when your solution becomes larger and more complex. For client-side development, an automated UI framework for testing can help you detect issues initiated by user actions.

When a solution is developed with unit testing, developers report greater productivity and a better quality product.

See also

[Testing tools for server-side development](#)

[Video: Creating and running UI test](#)

[Blog post: Easy Repro: what is it?](#)

[Video: Introduction to DevOps](#)

Client API Reference for model-driven apps

Article • 11/30/2022

This section contains reference documentation for client API object model that can be used with JavaScript libraries.

Important

- The Client API object model also contains the **Xrm.Internal** namespace, and use of the objects/methods in this namespace isn't supported. These objects, and any parts of the HTML Document Object Model (DOM), are subject to change without notice. We recommend that you don't use these functions or any script that depends on the DOM.
- Also, while debugging, you may find methods and objects in the Client API object model that aren't documented. Only documented objects and methods are supported.
- Most of the client scripting APIs available in this documentation also apply to Dynamics 365 Customer Engagement (on-premises). For a list of client scripting APIs not available for Customer Engagement (on-premises), see [Client scripting reference for Dynamics 365 Customer Engagement \(on-premises\)](#).

The topics under this section are organized as follows:

- Starts with reference for all the events, collections, and the execution context object.
- Continues on to provide information about methods for **attributes** and **controls** in Customer Engagement that are actually collections that appear under different objects in the Client API object model.
- Provides reference for properties and methods for the **formContext** and **gridContext** objects.
- Finally provides reference for namespaces in the **Xrm** object model.

Related topics

[Apply business logic using client scripting in model-driven apps](#)

[Understand the Client API object model](#)

[Model-driven apps Developer Overview](#)

Customization XML reference

Article • 11/30/2022

The `customizations.xml` file is one of the files included in an exported unmanaged solution. The file contains all or selected portions of the customizations and configurations for your system.

The solutions file is exported as a compressed `.zip` file. The contents of the unmanaged solutions file must be extracted before the `customizations` file can be edited. All the files of the unmanaged solution must be added to a compressed `.zip` file before it can be re-imported.

ⓘ Note

- Editing a managed solution file is not supported.
- Not all elements of the `customizations.xml` file can be edited. More information: [Support for Editing the Customization File](#)

In This Section

[Ribbon core schema](#) [Ribbon types schema](#)

[Ribbon WSS schema](#)

[Form XML schema](#)

[FetchXML schema](#)

Related Sections

[When to edit the customizations file](#)

[Edit the Customizations file with schema validation](#)

[Customize the Ribbon for Dynamics 365](#)

[Change application navigation using the SiteMap](#)

Ribbon core schema

Article • 11/30/2022

The following is the schema definition for the ribbon core portion of an import/export customization file. It is included from the [Customization solutions file schema](#).

`RibbonCore.xsd` schema includes `RibbonTypes.xsd` and `RibbonWss.xsd` and you can find schema in the `Schemas\9.0.0.2090\RibbonCore.xsd` folder when you download the Schemas zip file.

Download the [Schemas](#).

For more information, see [Package and distribute extensions with solutions](#).

Ribbon Core Schema

XML

```
<?xml version="1.0" encoding="utf-8"?>
<xs:schema id="CrmRibbonCore" xmlns:xs="https://www.w3.org/2001/XMLSchema">
    <xs:include schemaLocation="RibbonTypes.xsd" />
    <xs:include schemaLocation="RibbonWSS.xsd" />

    <!-- Root Element-->
    <xs:element name="RibbonDiffXml" type="RibbonGlobalDiffXmlType" />
    <xs:element name="CommandDefinitions" type="CommandDefinitionsType" />
    <xs:element name="RuleDefinitions" type="RuleDefinitionsGlobalType" />
    <xs:element name="Templates" type="TemplatesType" />
    <xs:element name="CustomActions" type="CustomActionsType" />

    <xs:element name="UI" type="CommandUIType">
        </xs:element>

    <!-- Element Types -->
    <xs:complexType name="RibbonEntityDiffXmlType">
        <xs:choice minOccurs="1" maxOccurs="1">
            <xs:sequence>
                <xs:element name="CustomActions" minOccurs="0" maxOccurs="1"
type="CustomActionsType" />
                <xs:element name="Templates" type="TemplatesType"
minOccurs="0" maxOccurs="1" />
                <xs:element name="CommandDefinitions" minOccurs="0"
maxOccurs="1" type="CommandDefinitionsType" />
                <xs:element name="RuleDefinitions" minOccurs="0"
maxOccurs="1" type="RuleDefinitionsEntityType" />
                <xs:element name="LocLabels" minOccurs="0" maxOccurs="1"
type="RibbonLocLabelsType" />
            </xs:sequence>
        </xs:choice>
    </xs:complexType>
```

```

        <xss:element name="RibbonNotSupported" minOccurs="1"
maxOccurs="1">
            <xss:complexType>
                <xss:sequence />
            </xss:complexType>
        </xss:element>
    </xss:choice>
</xss:complexType>

<xss:complexType name="RibbonGlobalDiffXmlType">
    <xss:sequence>
        <xss:element name="CustomActions" minOccurs="0" maxOccurs="1"
type="CustomActionsType" />
            <xss:element name="Templates" type="TemplatesType" minOccurs="0"
maxOccurs="1" />
            <xss:element name="CommandDefinitions" minOccurs="0"
maxOccurs="1" type="CommandDefinitionsType" />
            <xss:element name="RuleDefinitions" minOccurs="0" maxOccurs="1"
type="RuleDefinitionsGlobalType" />
            <xss:element name="LocLabels" minOccurs="0" maxOccurs="1"
type="RibbonLocLabelsType" />
    </xss:sequence>
</xss:complexType>

<xss:complexType name="CustomActionsType">
    <xss:choice minOccurs="0" maxOccurs="unbounded">
        <xss:element name="CustomAction" type="CustomActionType" />
        <xss:element name="HideCustomAction" type="HideCustomActionType"
/>
    </xss:choice>
</xss:complexType>

<xss:complexType name="CustomActionType">
    <xss:sequence>
        <xss:element name="CommandUIDefinition" minOccurs="1"
maxOccurs="1">
            <xss:complexType>
                <xss:choice>
                    <xss:element name="Button" type="ButtonType" />
                    <xss:element name="CheckBox" type="CheckBoxType" />
                    <xss:element name="ComboBox" type="ComboBoxType" />
                    <xss:element name="ColorPicker"
type="ColorPickerType" />
                    <xss:element name="ContextualGroup"
type="ContextualGroupType" />
                    <!-- <xss:element name="ContextualTabs"
type="ContextualTabsType" /> -->
                    <xss:element name="Controls" type="ControlsType" />
                    <xss:element name="DropDown" type="DropDownType" />
                    <xss:element name="FlyoutAnchor"
type="FlyoutAnchorType" />
                    <xss:element name="Gallery" type="GalleryType" />
                    <xss:element name="GalleryButton"
type="GalleryButtonType" />
                </xss:choice>
            </xss:complexType>
        </xss:element>
    </xss:sequence>
</xss:complexType>

```

```

        <xs:element name="GroupTemplate"
type="GroupTemplateType" />
            <xs:element name="Group" type="GroupType" />
            <xs:element name="Groups" type="GroupsType" />
            <xs:element name="InsertTable"
type="InsertTableType" />
                <!-- <xs:element name="Jewel" type="JewelType" /> -->
<
            <xs:element name="Label" type="LabelType" />
            <xs:element name="MRUSplitButton"
type="MRUSplitButtonType" />
                <xs:element name="MaxSize" type="MaxSizeType" />
                <xs:element name="Menu" type="MenuType" />
                <xs:element name="MenuSection"
type="MenuSectionType" />
                    <!-- <xs:element name="QAT" type="QATTType" /> -->
                    <!-- <xs:element name="Ribbon" type="RibbonType" />
-->
                <xs:element name="Scale" type="ScaleType" />
                <xs:element name="Scaling" type="ScalingType" />
                <xs:element name="Spinner" type="SpinnerType" />
                <xs:element name="SplitButton"
type="SplitButtonType" />
                    <xs:element name="Tab" type="TabType" />
                    <!-- <xs:element name="Tabs" type="TabsType" /> -->
                    <xs:element name="TextBox" type="TextBoxType" />
                    <xs:element name="ToggleButton"
type="ToggleButtonType" />
                        </xs:choice>
                    </xs:complexType>
                </xs:element>
            </xs:sequence>
            <xs:attribute name="Id" type="xs:string" />
            <xs:attribute name="Location" type="xs:string" />
            <xs:attribute name="Sequence" type="xs:int" />
            <xs:attribute name="Title" type="xs:string" />
        </xs:complexType>

<xs:complexType name="HideCustomActionType">
    <xs:attribute name="HideActionId" type="xs:string" />
    <xs:attribute name="Location" type="xs:string" />
    <xs:attribute name="Sequence" type="xs:int" />
    <xs:attribute name="Title" type="xs:string" />
</xs:complexType>
</xs:schema>

```

See also

- [Customize commands and the ribbon](#)
- [Ribbon core schema](#)
- [Ribbon types schema](#)

[Ribbon WSS schema](#)

[Customization solutions file schema](#)

Ribbon types schema

Article • 03/10/2023

The following is the schema definition for the ribbon types portion of an import/export customization file. Ribbon types schema is included from the [Ribbon core schema](#). You can find schema in the `Schemas\9.0.0.2090\RibbonTypes.xsd` folder when you download the Schemas zip file.

Download the [Schemas](#).

For more information, see [Package and distribute extensions with solutions](#).

Schema

XML

```
<?xml version="1.0" encoding="utf-8"?>
<xs:schema id="CrmRibbonTypes" xmlns:xs="https://www.w3.org/2001/XMLSchema"
>

    <!-- Command Definition Types -->
    <xs:complexType name="ActionsType">
        <xs:choice minOccurs="0" maxOccurs="unbounded">
            <xs:element name="JavaScriptFunction"
type="JavaScriptFunctionType" />
            <xs:element name="Url" type="UrlType" />
            <xs:element name="OutlookCommand" type="OutlookCommandType" />
        </xs:choice>
    </xs:complexType>
    <xs:complexType name="CommandDefinitionsType">
        <xs:sequence>
            <xs:element name="CommandDefinition"
type="CommandDefinitionType" minOccurs="0" maxOccurs="unbounded" />
        </xs:sequence>
    </xs:complexType>
    <xs:complexType name="CommandDefinitionType">
        <xs:sequence>
            <xs:element name="EnableRules" type="ReferenceEnableRulesType"
minOccurs="1" maxOccurs="1" />
            <xs:element name="DisplayRules" type="ReferenceDisplayRulesType"
minOccurs="1" maxOccurs="1" />
                <xs:element name="Actions" type="ActionsType" minOccurs="1"
maxOccurs="1" />
            </xs:sequence>
            <xs:attribute name="Id" type="xs:string" use="required" />
        </xs:complexType>
        <xs:complexType name="JavaScriptFunctionType">
            <xs:sequence>
```

```

        <xs:group ref="ParameterType" minOccurs="0"
maxOccurs="unbounded" />
    </xs:sequence>
    <xs:attribute name="FunctionName" type="JavaScriptIdentifier"
use="required" />
        <xs:attribute name="Library" type="xs:string" use="required" />
</xs:complexType>
<xs:complexType name="OutlookCommandType">
    <xs:attribute name="ActionType" type="Outlook ActionType"
use="required" />
        <xs:attribute name="Data" type="xs:string" use="optional" />
</xs:complexType>
<xs:complexType name="ReferenceEnableRulesType">
    <xs:sequence>
        <xs:element name="EnableRule" type="ReferenceEnableRuleType"
minOccurs="0" maxOccurs="unbounded" />
    </xs:sequence>
</xs:complexType>
<xs:complexType name="ReferenceDisplayRulesType">
    <xs:sequence>
        <xs:element name="DisplayRule" type="ReferenceDisplayRuleType"
minOccurs="0" maxOccurs="unbounded" />
    </xs:sequence>
</xs:complexType>
<xs:complexType name="ReferenceEnableRuleType">
    <xs:attribute name="Id" type="xs:string" use="required" />
</xs:complexType>
<xs:complexType name="ReferenceDisplayRuleType">
    <xs:attribute name="Id" type="xs:string" use="required" />
</xs:complexType>
<xs:complexType name="UrlType">
    <xs:sequence>
        <xs:group ref="NamedParameterType" minOccurs="0"
maxOccurs="unbounded" />
    </xs:sequence>
        <xs:attribute name="Address" type="xs:anyURI" use="required" />
        <xs:attribute name="WinMode" type="WinMode" use="optional" />
        <xs:attribute name="WinParams" type="xs:string" use="optional" />
        <xs:attribute name="PassParams" type="xs:boolean" use="optional" />
    </xs:complexType>

<!-- Command Value Restrictions -->
<xs:simpleType name="JavaScriptIdentifier">
    <xs:restriction base="xs:string">
        <xs:pattern value="[a-zA-Z$_][a-zA-Z$_0-9.]*" />
    </xs:restriction>
</xs:simpleType>
<xs:simpleType name="OutlookActionType">
    <xs:restriction base="xs:string">
        <xs:enumeration value="CheckForUpdates" />
        <xs:enumeration value="ConfigWizard" />
        <xs:enumeration value="GoTo" />
        <xs:enumeration value="GoOffline" />
        <xs:enumeration value="Help" />
        <xs:enumeration value="OpenOlkForm" />
    </xs:restriction>
</xs:simpleType>

```

```

<xs:enumeration value="Promote" />
<xs:enumeration value="SetRegarding" />
<xs:enumeration value="Settings" />
<xs:enumeration value="SignOut" />
<xs:enumeration value="SignOutForgetMe" />
<xs:enumeration value="OutlookImport" />
<xs:enumeration value="Sync" />
</xs:restriction>
</xs:simpleType>
<xs:simpleType name="WinMode">
<xs:restriction base="xs:integer">
<xs:minInclusive value="0" />
<xs:maxInclusive value="2" />
</xs:restriction>
</xs:simpleType>

<!-- Rule Container Types -->
<xs:complexType name="RuleDefinitionsEntityType">
<xs:sequence>
<xs:element name="TabDisplayRules"
type="TabDisplayRulesEntityType" minOccurs="1" maxOccurs="1" />
<xs:element name="DisplayRules" type="DisplayRulesType"
minOccurs="1" maxOccurs="1" />
<xs:element name="EnableRules" type="EnableRulesType"
minOccurs="1" maxOccurs="1" />
</xs:sequence>
</xs:complexType>
<xs:complexType name="RuleDefinitionsGlobalType">
<xs:sequence>
<xs:element name="TabDisplayRules"
type="TabDisplayRulesGlobalType" minOccurs="1" maxOccurs="1" />
<xs:element name="DisplayRules" type="DisplayRulesType"
minOccurs="1" maxOccurs="1" />
<xs:element name="EnableRules" type="EnableRulesType"
minOccurs="1" maxOccurs="1" />
</xs:sequence>
</xs:complexType>

<xs:complexType name="EnableRulesType">
<xs:sequence minOccurs="0" maxOccurs="unbounded">
<xs:element name="EnableRule">
<xs:complexType>
<xs:choice minOccurs="1" maxOccurs="unbounded">
<xs:group ref="EnableRulesGroup" />
<xs:element name="OrRule" type="OrEnableRuleType" />
</xs:choice>
<xs:attribute name="Id" type="xs:string" use="required"
/>
</xs:complexType>
</xs:element>
</xs:sequence>
</xs:complexType>

<xs:complexType name="OrEnableRuleType">
<xs:sequence minOccurs="2" maxOccurs="unbounded">

```

```

<xs:element name="Or">
    <xs:complexType>
        <xs:choice minOccurs="1" maxOccurs="unbounded">
            <xs:group ref="EnableRulesGroup" />
        </xs:choice>
    </xs:complexType>
</xs:element>
</xs:sequence>
</xs:complexType>

<xs:group name="EnableRulesGroup">
    <xs:choice>
        <xs:element name="CrmClientTypeRule"
type="CrmClientTypeRuleType" />
            <xs:element name="CrmOutlookClientTypeRule"
type="CrmOutlookClientTypeRuleType" />
                <xs:element name="CrmOfflineAccessStateRule"
type="CrmOfflineAccessStateRuleType" />
                    <xs:element name="CustomRule" type="CustomRuleType" />
                    <xs:element name="EntityRule" type="EntityRuleType" />
                    <xs:element name="FormStateRule" type="FormStateRuleType" />
                    <xs:element name="OutlookItemTrackingRule"
type="OutlookItemTrackingRuleType" />
                    <xs:element name="OutlookVersionRule"
type="OutlookVersionRuleType" />
                        <xs:element name="PageRule" type="PageRuleType" />
                        <xs:element name="RecordPrivilegeRule"
type="RecordPrivilegeRuleType" />
                            <xs:element name="SelectionCountRule"
type="SelectionCountRuleType" />
                                <xs:element name="SkuRule" type="SkuRuleType" />
                                <xs:element name="ValueRule" type="ValueRuleType" />
                                <xs:element name="CommandClientTypeRule"
type="CommandClientTypeRuleType" />
                            </xs:choice>
                        </xs:group>

<xs:complexType name="TabDisplayRulesEntityType">
    <xs:sequence minOccurs="0" maxOccurs="unbounded">
        <xs:element name="TabDisplayRule">
            <xs:complexType>
                <xs:choice minOccurs="1" maxOccurs="unbounded">
                    <xs:element name="EntityRule"
type="EntityTabRuleType" />
                </xs:choice>
                <xs:attribute name="TabCommand" type="xs:string"
use="required" />
            </xs:complexType>
        </xs:element>
    </xs:sequence>
</xs:complexType>

<xs:complexType name="TabDisplayRulesGlobalType">
    <xs:sequence minOccurs="0" maxOccurs="unbounded">
        <xs:element name="TabDisplayRule">

```

```

        <xs:complexType>
            <xs:choice minOccurs="1" maxOccurs="unbounded">
                <xs:element name="EntityRule"
type="EntityTabRuleType" />
                    <xs:element name="PageRule" type="PageTabRuleType"
/>
                </xs:choice>
                <xs:attribute name="TabCommand" type="xs:string"
use="required" />
            </xs:complexType>
        </xs:element>
    </xs:sequence>
</xs:complexType>

<xs:complexType name="DisplayRulesType">
    <xs:sequence minOccurs="0" maxOccurs="unbounded">
        <xs:element name="DisplayRule">
            <xs:complexType>
                <xs:choice minOccurs="1" maxOccurs="unbounded">
                    <xs:group ref="DisplayRulesGroup" />
                    <xs:element name="OrRule" type="OrDisplayRuleType"
/>
                </xs:choice>
                <xs:attribute name="Id" type="xs:string" use="required"
/>
            </xs:complexType>
        </xs:element>
    </xs:sequence>
</xs:complexType>

<xs:complexType name="OrDisplayRuleType">
    <xs:sequence minOccurs="2" maxOccurs="unbounded">
        <xs:element name="Or">
            <xs:complexType>
                <xs:choice minOccurs="1" maxOccurs="unbounded">
                    <xs:group ref="DisplayRulesGroup" />
                </xs:choice>
            </xs:complexType>
        </xs:element>
    </xs:sequence>
</xs:complexType>

<xs:group name="DisplayRulesGroup">
    <xs:choice>
        <xs:element name="CrmClientTypeRule"
type="CrmClientTypeRuleType" />
        <xs:element name="CommandClientTypeRule"
type="CommandClientTypeRuleType" />
        <xs:element name="DeviceTypeRule" type="DeviceTypeRuleType" />
        <xs:element name="CrmOutlookClientTypeRule"
type="CrmOutlookClientTypeRuleType" />
        <xs:element name="CrmOutlookClientVersionRule"
type="CrmOutlookClientVersionRuleType" />
        <xs:element name="CrmOfflineAccessStateRule"
type="CrmOfflineAccessStateRuleType" />
    </xs:choice>

```

```

        <xss:element name="EntityRule" type="EntityType" />
        <xss:element name="EntityPrivilegeRule"
type="EntityPrivilegeRuleType" />
            <xss:element name="EntityPropertyRule"
type="EntityPropertyRuleType" />
                <xss:element name="FormEntityContextRule"
type="FormEntityContextRuleType" />
                    <xss:element name="FormStateRule" type="FormStateRuleType" />
                    <xss:element name="MiscellaneousPrivilegeRule"
type="MiscellaneousPrivilegeRuleType" />
                <xss:element name="OrganizationSettingRule"
type="OrganizationSettingRuleType" />
                <xss:element name="OutlookRenderTypeRule"
type="OutlookRenderTypeRuleType" />
                    <xss:element name="OutlookVersionRule"
type="OutlookVersionRuleType" />
                    <xss:element name="PageRule" type="PageRuleType" />
                    <xss:element name="ReferencingAttributeRequiredRule"
type="ReferencingAttributeRequiredRuleType" />
                <xss:element name="RelationshipTypeRule"
type="RelationshipTypeRuleType" />
                    <xss:element name="SkuRule" type="SkuRuleType" />
                    <xss:element name="ValueRule" type="ValueRuleType" />
                    <xss:element name="OptionSetRule" type="OptionSetRuleType" />
                    <xss:element name="FormTypeRule" type="FormTypeRuleType" />
                    <xss:element name="HideForTabletExperienceRule" type
="HideForTabletExperienceRuleType" />
                    <xss:element name="HideIfNetBreezeNotAvailableRule" type
="HideIfNetBreezeNotAvailableRuleType" />
                    <xss:element name="HideIfServiceMetadataAvailableRule" type
="HideIfServiceMetadataAvailableRuleType" />
                    <xss:element name="HideIfSharepointS2SConfigurationEnabledRule"
type ="HideIfSharepointS2SConfigurationEnabledRuleType" />
                    <xss:element name="HideIfExportToExcelNotEnabledRule"
type="HideIfExportToExcelNotEnabledRuleType" />
                    <xss:element name="IsExportToExcelOnlineEnabledRule"
type="IsExportToExcelOnlineEnabledRuleType" />
                    <xss:element name="HideIfDisabledForMobileRule"
type="HideIfDisabledForMobileRuleType" />
                    <xss:element name="HideIfHybridSSSNotEnabledRule"
type="HideIfHybridSSSNotEnabledRuleType" />
                    <xss:element name="HideIfEmailSignatureNotEnabledRule"
type="HideIfEmailSignatureNotEnabledRuleType" />
                    <xss:element name="HideIfReverseHybridSSSNotEnabledRule"
type="HideIfReverseHybridSSSNotEnabledRuleType" />
                    <xss:element name="FeatureControlRule"
type="FeatureControlRuleType" />
                    <xss:element name="HideIfDelveNotAvailableRule" type
="HideIfDelveNotAvailableRuleType" />
                    <xss:element name="HideIfTestExchangeServerNotEnabledRule"
type="HideIfTestExchangeServerNotEnabledRuleType" />
                    <xss:element name="HideIfSSTroubleshootingNotEnabledRule"
type="HideIfSSTroubleshootingNotEnabledRuleType" />
                    <xss:element name="HideIfCurrentUserIsNotSystemAdministratorRule"
type="HideIfCurrentUserIsNotSystemAdministratorRuleType" />

```

```

        <xs:element name="HideIfPowerBITileNotAvailableRule" type
="HideIfPowerBITileNotAvailableRuleType" />
            <xs:element name="HideIfProcessActiveRule"
type="HideIfProcessActiveRuleType" />
                <xs:element name="HideIfProcessInactiveRule"
type="HideIfProcessInactiveRuleType" />
                    <xs:element name="HideIfProcessUnificationIsDisabledRule"
type="HideIfProcessUnificationIsDisabledRuleType" />
                        <xs:element
name="HideIf0365UserDoesNotHaveExchangeSubscriptionsRule" type
="HideIf0365UserDoesNotHaveExchangeSubscriptionsRuleType" />
                            <xs:element name="HideIfEmailIsApprovedByAdminRule" type
="HideIfEmailIsApprovedByAdminType" />
                                <xs:element name="HideIfUserIsNotTenantAdminRule" type
="HideIfUserIsNotTenantAdminType" />
                                    <xs:element name="HideIfEmailIsApprovedByAdminBasedOnESPRule"
type ="HideIfEmailIsApprovedByAdminBasedOnESPRuleType" />
                                        <xs:element name="HideIfProductRecommendationsNotEnabledRule"
type ="HideIfProductRecommendationsNotEnabledType" />
                                            </xs:choice>
</xs:group>

<!-- Rule Types -->
<xs:complexType name="CrmClientTypeRuleType">
    <xs:attribute name="Type" use="required">
        <xs:simpleType>
            <xs:restriction base="xs:string">
                <xs:enumeration value="Web" />
                <xs:enumeration value="Outlook" />
            </xs:restriction>
        </xs:simpleType>
    </xs:attribute>
    <xs:attributeGroup ref="StandardRuleAttributes" />
</xs:complexType>

<xs:complexType name="CommandClientTypeRuleType">
    <xs:attribute name="Type" use="required">
        <xs:simpleType>
            <xs:restriction base="xs:string">
                <xs:enumeration value="Modern" />
                <xs:enumeration value="Refresh" />
                <xs:enumeration value="Legacy" />
            </xs:restriction>
        </xs:simpleType>
    </xs:attribute>
    <xs:attributeGroup ref="StandardRuleAttributes" />
</xs:complexType>

<xs:complexType name="DeviceTypeRuleType">
    <xs:attribute name="Type" use="required">
        <xs:simpleType>
            <xs:restriction base="xs:string">
                <xs:enumeration value="None" />
                <xs:enumeration value="Phone" />
                <xs:enumeration value="Tablet" />
            </xs:restriction>
        </xs:simpleType>
    </xs:attribute>
</xs:complexType>

```

```

                <xs:enumeration value="Web" />
                <xs:enumeration value="Outlook" />
                <xs:enumeration value="InteractionCentric" />
            </xs:restriction>
        </xs:simpleType>
    </xs:attribute>
    <xs:attributeGroup ref="StandardRuleAttributes" />
</xs:complexType>

<xs:complexType name="CrmOfflineAccessStateRuleType">
    <xs:attribute name="State" use="required">
        <xs:simpleType>
            <xs:restriction base="xs:string">
                <xs:enumeration value="Online" />
                <xs:enumeration value="Offline" />
            </xs:restriction>
        </xs:simpleType>
    </xs:attribute>
    <xs:attributeGroup ref="StandardRuleAttributes" />
</xs:complexType>

<xs:complexType name="CrmOutlookClientTypeRuleType">
    <xs:attribute name="Type" use="required">
        <xs:simpleType>
            <xs:restriction base="xs:string">
                <xs:enumeration value="CrmForOutlook" />
                <xs:enumeration value="CrmForOutlookOfflineAccess" />
            </xs:restriction>
        </xs:simpleType>
    </xs:attribute>
    <xs:attributeGroup ref="StandardRuleAttributes" />
</xs:complexType>

<xs:complexType name="OptionSetRuleType">
    <xs:attribute name="OptionSet" type="xs:string" use="required"/>
    <xs:attribute name="StateCode" type="xs:string" use="required"/>
    <xs:attribute name="ObjectTypeCode" type="xs:string"
use="required"/>
    <xs:attributeGroup ref="StandardRuleAttributes" />
</xs:complexType>

<xs:complexType name="CustomRuleType">
    <xs:sequence>
        <xs:group ref="ParameterType" minOccurs="0"
maxOccurs="unbounded" />
    </xs:sequence>
    <xs:attribute name="FunctionName" type="JavaScriptIdentifier"
use="required" />
    <xs:attribute name="Library" type="xs:string" use="required" />
    <xs:attributeGroup ref="StandardRuleAttributes" />
</xs:complexType>

<xs:complexType name="EntityRuleType">
    <xs:attributeGroup ref="EntityRuleTypeAttributes" />
    <xs:attributeGroup ref="StandardRuleAttributes" />

```

```

    </xs:complexType>
    <xs:complexType name="EntityTabRuleType">
        <xs:attributeGroup ref="EntityRuleTypeAttributes" />
    </xs:complexType>
    <xs:attributeGroup name="EntityRuleTypeAttributes">
        <xs:attribute name="EntityName" type="xs:string" use="optional" />
        <xs:attribute name="AppliesTo" type="AppliesToType" use="optional" />
    </>
        <xs:attribute name="Context" use="optional">
            <xs:simpleType>
                <xs:restriction base="xs:string">
                    <xs:pattern value="[a-zA-Z_][a-zA-Z_0-9]*" />
                </xs:restriction>
            </xs:simpleType>
        </xs:attribute>
    </xs:attributeGroup>

    <xs:complexType name="EntityPropertyRuleType">
        <xs:attribute name="EntityName" type="xs:string" use="optional" />
        <xs:attribute name="AppliesTo" type="AppliesToType" use="optional" />
    </>
        <xs:attribute name="PropertyName" use="required">
            <xs:simpleType>
                <xs:restriction base="xs:string">
                    <xs:enumeration value="DuplicateDetectionEnabled" />
                    <xs:enumeration value="GridFiltersEnabled" />
                    <xs:enumeration value="HasStateCode" />
                    <xs:enumeration value="IsConnectionsEnabled" />
                    <xs:enumeration value="MailMergeEnabled" />
                    <xs:enumeration value="WorksWithQueue" />
                    <xs:enumeration value="HasActivities" />
                    <xs:enumeration value="IsActivity" />
                    <xs:enumeration value="HasNotes" />
                    <xs:enumeration value="IsCustomizable" />
                    <xs:enumeration value="IsActivityParty" />
                    <xs:enumeration value="HasEmailAddresses" />
                    <xs:enumeration value="IsChildEntity" />
                    <xs:enumeration value="IsImportable" />
                    <xs:enumeration value=".IsEnabledForCharts" />
                    <xs:enumeration value="IsBusinessProcessEnabled" />
                    <xs:enumeration value="HasFeedback" />
                    <xs:enumeration value="IsBPFEentity" />
                </xs:restriction>
            </xs:simpleType>
        </xs:attribute>
        <xs:attribute name="PropertyValue" type="xs:boolean" use="required" />
    </>
        <xs:attributeGroup ref="StandardRuleAttributes" />
    </xs:complexType>

    <xs:complexType name="FormEntityContextRuleType">
        <xs:attribute name="EntityName" type="xs:string" use="required" />
        <xs:attributeGroup ref="StandardRuleAttributes" />
    </xs:complexType>

```

```

<xs:complexType name="FormStateRuleType">
    <xs:attribute name="State" use="required">
        <xs:simpleType>
            <xs:restriction base="xs:string">
                <xs:enumeration value="Create" />
                <xs:enumeration value="Existing" />
                <xs:enumeration value="ReadOnly" />
                <xs:enumeration value="Disabled" />
                <xs:enumeration value="BulkEdit" />
            </xs:restriction>
        </xs:simpleType>
    </xs:attribute>
    <xs:attributeGroup ref="StandardRuleAttributes" />
</xs:complexType>

<xs:complexType name="OrganizationSettingRuleType">
    <xs:attribute name="Setting" use="required">
        <xs:simpleType>
            <xs:restriction base="xs:string">
                <xs:enumeration value="IsSharepointEnabled" />
                <xs:enumeration value="IsSOPIntegrationEnabled" />
                <xs:enumeration value="IsFiscalCalendarDefined" />
                <xs:enumeration value="IsReadFormModeDefined" />
                <xs:enumeration
value="IsBPFFeatureCustomizationEnabled" />
            </xs:restriction>
        </xs:simpleType>
    </xs:attribute>
    <xs:attributeGroup ref="StandardRuleAttributes" />
</xs:complexType>

<xs:complexType name="FormTypeRuleType">
    <xs:attribute name="Type" use="required">
        <xs:simpleType>
            <xs:restriction base="xs:string">
                <xs:enumeration value="Main" />
                <xs:enumeration value="Preview" />
                <xs:enumeration value="AppointmentBook" />
                <xs:enumeration value="Dashboard" />
                <xs:enumeration value="Quick" />
                <xs:enumeration value="QuickCreate" />
                <xs:enumeration value="Card" />
                <xs:enumeration value="MainInteractionCentric" />
            </xs:restriction>
        </xs:simpleType>
    </xs:attribute>
    <xs:attributeGroup ref="StandardRuleAttributes" />
</xs:complexType>

<xs:complexType name="HideForTabletExperienceRuleType">
    <xs:attributeGroup ref="StandardRuleAttributes" />
</xs:complexType>

<xs:complexType name="HideIfNetBreezeNotAvailableRuleType">
    <xs:attributeGroup ref="StandardRuleAttributes" />

```

```
</xs:complexType>

<xs:complexType name="HideIfDisabledForMobileRuleType">
    <xs:attributeGroup ref="StandardRuleAttributes" />
</xs:complexType>

<xs:complexType name="HideIfServiceMetadataAvailableRuleType">
    <xs:attributeGroup ref="StandardRuleAttributes" />
</xs:complexType>

<xs:complexType name="HideIfSharepointS2SConfigurationEnabledRuleType">
    <xs:attributeGroup ref="StandardRuleAttributes" />
</xs:complexType>

<xs:complexType name="HideIfExportToExcelNotEnabledRuleType">
    <xs:attributeGroup ref="StandardRuleAttributes" />
</xs:complexType>

<xs:complexType name="IsExportToExcelOnlineEnabledRuleType">
    <xs:attributeGroup ref="StandardRuleAttributes" />
</xs:complexType>

<xs:complexType name="HideIfDelveNotAvailableRuleType">
    <xs:attributeGroup ref="StandardRuleAttributes" />
</xs:complexType>

<xs:complexType name="HideIfPowerBITileNotAvailableRuleType">
    <xs:attributeGroup ref="StandardRuleAttributes" />
</xs:complexType>

<xs:complexType name="HideIfProcessActiveRuleType">
    <xs:attributeGroup ref="StandardRuleAttributes" />
</xs:complexType>

<xs:complexType name="HideIfProcessInactiveRuleType">
    <xs:attributeGroup ref="StandardRuleAttributes" />
</xs:complexType>

<xs:complexType name="HideIfProcessUnificationIsDisabledRuleType">
    <xs:attributeGroup ref="StandardRuleAttributes" />
</xs:complexType>

<xs:complexType
name="HideIfO365UserDoesNotHaveExchangeSubscriptionsRuleType">
    <xs:attributeGroup ref="StandardRuleAttributes" />
</xs:complexType>

<xs:complexType name="HideIfEmailIsApprovedByAdminBasedOnESPRuleType">
    <xs:attributeGroup ref="StandardRuleAttributes" />
</xs:complexType>

<xs:complexType name="HideIfSSSTroubleshootingNotEnabledRuleType">
    <xs:attributeGroup ref="StandardRuleAttributes" />
</xs:complexType>
```

```

<xs:complexType
  name="HideIfCurrentUserIsNotSystemAdministratorRuleType">
  <xs:attributeGroup ref="StandardRuleAttributes" />
</xs:complexType>

<xs:complexType name="HideIfHybridSSSNotEnabledRuleType">
  <xs:attributeGroup ref="StandardRuleAttributes" />
</xs:complexType>

<xs:complexType name="HideIfEmailSignatureNotEnabledRuleType">
  <xs:attributeGroup ref="StandardRuleAttributes" />
</xs:complexType>

<xs:complexType name="HideIfReverseHybridSSSNotEnabledRuleType">
  <xs:attributeGroup ref="StandardRuleAttributes" />
</xs:complexType>

<xs:complexType name="HideIfEmailIsApprovedByAdminType">
  <xs:attributeGroup ref="StandardRuleAttributes" />
</xs:complexType>

<xs:complexType name="HideIfUserIsNotTenantAdminType">
  <xs:attributeGroup ref="StandardRuleAttributes" />
</xs:complexType>

<xs:complexType name="HideIfProductRecommendationsNotEnabledType">
  <xs:attributeGroup ref="StandardRuleAttributes" />
</xs:complexType>

<xs:complexType name="HideIfTestExchangeServerNotEnabledRuleType">
  <xs:attributeGroup ref="StandardRuleAttributes" />
</xs:complexType>

<xs:complexType name="OutlookItemTrackingRuleType">
  <xs:attribute name="TrackedInCrm" type="xs:boolean" use="required"
/>
  <xs:attribute name="AppliesTo" type="AppliesToPrimaryType"
use="optional" />
  <xs:attributeGroup ref="StandardRuleAttributes" />
</xs:complexType>

<xs:complexType name="OutlookRenderTypeRuleType">
  <xs:attribute name="Type" use="required">
    <xs:simpleType>
      <xs:restriction base="xs:string">
        <xs:enumeration value="Web" />
        <xs:enumeration value="Outlook" />
      </xs:restriction>
    </xs:simpleType>
  </xs:attribute>
  <xs:attributeGroup ref="StandardRuleAttributes" />
</xs:complexType>

<xs:complexType name="OutlookVersionRuleType">
  <xs:attribute name="Version" use="required">

```

```

<xs:simpleType>
    <xs:restriction base="xs:string">
        <xs:enumeration value="2003" />
        <xs:enumeration value="2007" />
        <xs:enumeration value="2010" />
    </xs:restriction>
</xs:simpleType>
</xs:attribute>
<xs:attributeGroup ref="StandardRuleAttributes" />
</xs:complexType>

<xs:complexType name="CrmOutlookClientVersionRuleType">
    <xs:attribute name="Major" type="xs:integer" use="required"/>
    <xs:attribute name="Minor" type="xs:integer" use="optional"/>
    <xs:attribute name="Build" type="xs:integer" use="optional"/>
    <xs:attribute name="Revision" type="xs:integer" use="optional"/>
    <xs:attributeGroup ref="StandardRuleAttributes" />
</xs:complexType>

<xs:complexType name="PageRuleType">
    <xs:attribute name="Address" type="xs:anyURI" use="required" />
    <xs:attributeGroup ref="StandardRuleAttributes" />
</xs:complexType>
<xs:complexType name="PageTabRuleType">
    <xs:attribute name="Address" type="xs:anyURI" use="required" />
</xs:complexType>

<xs:complexType name="RecordPrivilegeRuleType">
    <xs:attribute name="PrivilegeType" type="PrivilegeTypeType"
use="required" />
    <xs:attribute name="AppliesTo" type="AppliesToPrimaryType"
use="optional" />
    <xs:attributeGroup ref="StandardRuleAttributes" />
</xs:complexType>
<xs:complexType name="EntityPrivilegeRuleType">
    <xs:attribute name="PrivilegeType" type="PrivilegeTypeType"
use="required" />
    <xs:attribute name="PrivilegeDepth" type="PrivilegeDepthType"
use="required" />
    <xs:attribute name="AppliesTo" type="AppliesToType" use="optional"
/>
    <xs:attribute name="EntityName" type="xs:string" use="optional" />
    <xs:attributeGroup ref="StandardRuleAttributes" />
</xs:complexType>

<xs:complexType name="MiscellaneousPrivilegeRuleType">
    <xs:attribute name="PrivilegeName" type="xs:string" use="required"
/>
    <xs:attribute name="PrivilegeDepth" type="PrivilegeDepthType"
use="optional" />
    <xs:attributeGroup ref="StandardRuleAttributes" />
</xs:complexType>

<xs:complexType name="ReferencingAttributeRequiredRuleType">
    <xs:attributeGroup ref="StandardRuleAttributes" />

```

```

    </xs:complexType>

    <xs:complexType name="RelationshipTypeRuleType">
        <xs:attribute name="AppliesTo" type="AppliesToSelectedType"
use="required" />
        <xs:attribute name="RelationshipType" use="optional">
            <xs:simpleType>
                <xs:restriction base="xs:string">
                    <xs:enumeration value="OneToMany" />
                    <xs:enumeration value="ManyToMany" />
                    <xs:enumeration value="NoRelationship" />
                </xs:restriction>
            </xs:simpleType>
        </xs:attribute>
        <xs:attribute name="AllowCustomRelationship" use="optional"
type="xs:boolean" default="true" />
        <xs:attribute name="AllowSystemRelationship" use="optional"
type="xs:boolean" default="true" />
        <xs:attributeGroup ref="StandardRuleAttributes" />
    </xs:complexType>

    <xs:complexType name="SelectionCountRuleType">
        <xs:attribute name="AppliesTo" type="AppliesToType" use="optional"
/>
        <xs:attribute name="Minimum" type="xs:integer" use="optional" />
        <xs:attribute name="Maximum" type="xs:integer" use="optional" />
        <xs:attributeGroup ref="StandardRuleAttributes" />
    </xs:complexType>

    <xs:complexType name="SkuRuleType">
        <xs:attribute name="Sku" use="required">
            <xs:simpleType>
                <xs:restriction base="xs:string">
                    <xs:enumeration value="OnPremise" />
                    <xs:enumeration value="Online" />
                    <xs:enumeration value="Spla" />
                </xs:restriction>
            </xs:simpleType>
        </xs:attribute>
        <xs:attributeGroup ref="StandardRuleAttributes" />
    </xs:complexType>

    <xs:complexType name="ValueRuleType">
        <xs:attribute name="Field" type="xs:string" use="required" />
        <xs:attribute name="Value" type="xs:string" use="required" />
        <xs:attributeGroup ref="StandardRuleAttributes" />
    </xs:complexType>

    <xs:complexType name="FeatureControlRuleType">
        <xs:attribute name="FeatureControlBit" type="xs:string"
use="required" />
        <xs:attribute name="ExpectedValue" type="xs:boolean" use="required"
/>
        <xs:attributeGroup ref="StandardRuleAttributes" />
    </xs:complexType>

```

```

<!-- Rule Attributes -->
<xs:attributeGroup name="StandardRuleAttributes">
    <xs:attribute name="Default" type="xs:boolean" use="optional" />
    <xs:attribute name="InvertResult" type="xs:boolean" use="optional" />
</xs:attributeGroup>

<!-- Rule Value Restrictions -->
<xs:simpleType name="AppliesToType">
    <xs:restriction base="xs:string">
        <xs:enumeration value="PrimaryEntity" />
        <xs:enumeration value="SelectedEntity" />
    </xs:restriction>
</xs:simpleType>
<xs:simpleType name="AppliesToPrimaryType">
    <xs:restriction base="xs:string">
        <xs:enumeration value="PrimaryEntity" />
    </xs:restriction>
</xs:simpleType>
<xs:simpleType name="AppliesToSelectedType">
    <xs:restriction base="xs:string">
        <xs:enumeration value="SelectedEntity" />
    </xs:restriction>
</xs:simpleType>

<xs:simpleType name="PrivilegeDepthType">
    <xs:restriction base="xs:string">
        <xs:enumeration value="None" />
        <xs:enumeration value="Basic" />
        <xs:enumeration value="Local" />
        <xs:enumeration value="Deep" />
        <xs:enumeration value="Global" />
    </xs:restriction>
</xs:simpleType>
<xs:simpleType name="PrivilegeTypeType">
    <xs:restriction base="xs:string">
        <xs:enumeration value="Create" />
        <xs:enumeration value="Read" />
        <xs:enumeration value="Write" />
        <xs:enumeration value="Delete" />
        <xs:enumeration value="Assign" />
        <xs:enumeration value="Share" />
        <xs:enumeration value="Append" />
        <xs:enumeration value="AppendTo" />
    </xs:restriction>
</xs:simpleType>

<!-- Parameter Types -->
<xs:group name="ParameterType">
    <xs:choice>
        <xs:element name="BoolParameter" type="BoolParameterType" />
        <xs:element name="CrmParameter" type="CrmParameterType" />
        <xs:element name="DecimalParameter" type="DecimalParameterType" />
    </xs:choice>
</xs:group>

```

```

        <xs:element name="IntParameter" type="IntParameterType" />
        <xs:element name="StringParameter" type="StringParameterType" />
    </xs:choice>
</xs:group>
<xs:group name="NamedParameterType">
    <xs:choice>
        <xs:element name="BoolParameter" type="BoolNamedParameterType" />
    <xs:element name="CrmParameter" type="CrmNamedParameterType" />
    <xs:element name="DecimalParameter" type="DecimalNamedParameterType" />
        <xs:element name="IntParameter" type="IntNamedParameterType" />
        <xs:element name="StringParameter" type="StringNamedParameterType" />
    </xs:choice>
</xs:group>
<xs:complexType name="BoolParameterType">
    <xs:attribute name="Value" type="xs:boolean" use="required" />
</xs:complexType>
<xs:complexType name="BoolNamedParameterType">
    <xs:attribute name="Value" type="xs:boolean" use="required" />
    <xs:attribute name="Name" type="xs:string" use="required" />
</xs:complexType>
<xs:complexType name="CrmParameterType">
    <xs:attribute name="Value" type="CrmParameterValue" use="required" />
/>
</xs:complexType>
<xs:complexType name="CrmNamedParameterType">
    <xs:attribute name="Value" type="CrmNamedParameterValue" use="required" />
    <xs:attribute name="Name" type="xs:string" use="required" />
</xs:complexType>
<xs:complexType name="DecimalParameterType">
    <xs:attribute name="Value" type="xs:decimal" use="required" />
</xs:complexType>
<xs:complexType name="DecimalNamedParameterType">
    <xs:attribute name="Value" type="xs:decimal" use="required" />
    <xs:attribute name="Name" type="xs:string" use="required" />
</xs:complexType>
<xs:complexType name="IntParameterType">
    <xs:attribute name="Value" type="xs:integer" use="required" />
</xs:complexType>
<xs:complexType name="IntNamedParameterType">
    <xs:attribute name="Value" type="xs:integer" use="required" />
    <xs:attribute name="Name" type="xs:string" use="required" />
</xs:complexType>
<xs:complexType name="StringParameterType">
    <xs:attribute name="Value" type="xs:string" use="required" />
</xs:complexType>
<xs:complexType name="StringNamedParameterType">
    <xs:attribute name="Value" type="xs:string" use="required" />
    <xs:attribute name="Name" type="xs:string" use="required" />
</xs:complexType>

<xs:simpleType name="CrmParameterValue">

```

```

<xs:restriction base="xs:string">
    <xs:enumeration value="PrimaryEntityTypeCode" />
    <xs:enumeration value="PrimaryEntityTypeName" />
    <xs:enumeration value="PrimaryItemIds" />
    <xs:enumeration value="FirstPrimaryItemId" />
    <xs:enumeration value="PrimaryControl" />
    <xs:enumeration value="PrimaryControlId" />
    <xs:enumeration value="SelectedEntityTypeCode" />
    <xs:enumeration value="SelectedEntityTypeName" />
    <xs:enumeration value="FirstSelectedItemId" />
    <xs:enumeration value="SelectedControl" />
    <xs:enumeration value="SelectedControlSelectedItemCount" />
    <xs:enumeration value="SelectedControlSelectedItemIds" />
    <xs:enumeration value="SelectedControlSelectedItemReferences" />
    <xs:enumeration value="SelectedControlAllItemCount" />
    <xs:enumeration value="SelectedControlAllItemIds" />
    <xs:enumeration value="SelectedControlAllItemReferences" />
    <xs:enumeration value="SelectedControlUnselectedItemCount" />
    <xs:enumeration value="SelectedControlUnselectedItemIds" />
    <xs:enumeration value="SelectedControlUnselectedItemReferences" />
/>
    <xs:enumeration value="OrgName" />
    <xs:enumeration value="OrgLcid" />
    <xs:enumeration value="UserLcid" />
    <xs:enumeration value="CommandProperties" />
</xs:restriction>
</xs:simpleType>

<xs:simpleType name="CrmNamedParameterValue">
    <xs:restriction base="xs:string">
        <xs:enumeration value="PrimaryEntityTypeCode" />
        <xs:enumeration value="PrimaryEntityTypeName" />
        <xs:enumeration value="PrimaryItemIds" />
        <xs:enumeration value="FirstPrimaryItemId" />
        <xs:enumeration value="PrimaryControl" />
        <xs:enumeration value="PrimaryControlId" />
        <xs:enumeration value="SelectedEntityTypeCode" />
        <xs:enumeration value="SelectedEntityTypeName" />
        <xs:enumeration value="FirstSelectedItemId" />
        <xs:enumeration value="SelectedControl" />
        <xs:enumeration value="SelectedControlSelectedItemCount" />
        <xs:enumeration value="SelectedControlSelectedItemIds" />
        <xs:enumeration value="SelectedControlAllItemCount" />
        <xs:enumeration value="SelectedControlAllItemIds" />
        <xs:enumeration value="SelectedControlUnselectedItemCount" />
        <xs:enumeration value="SelectedControlUnselectedItemIds" />
        <xs:enumeration value="OrgName" />
        <xs:enumeration value="OrgLcid" />
        <xs:enumeration value="UserLcid" />
        <xs:enumeration value="CommandProperties" />
    </xs:restriction>
</xs:simpleType>

<!-- LocLabels Types -->
<xss:complexType name="RibbonLocLabelsType">

```

```
<xs:sequence minOccurs="1" maxOccurs="1">
    <xs:element name="LocLabel" type="RibbonLocLabelType"
minOccurs="0" maxOccurs="unbounded" />
</xs:sequence>
</xs:complexType>
<xs:complexType name="RibbonLocLabelType">
    <xs:sequence minOccurs="1" maxOccurs="1">
        <xs:element name="Titles" type="RibbonTitlesType" minOccurs="1"
maxOccurs="1" />
    </xs:sequence>
    <xs:attribute name="Id" type="xs:string" use="required" />
</xs:complexType>
<xs:complexType name="RibbonTitlesType">
    <xs:sequence minOccurs="1" maxOccurs="1">
        <xs:element name="Title" type="RibbonTitleType" minOccurs="1"
maxOccurs="unbounded" />
    </xs:sequence>
</xs:complexType>
<xs:complexType name="RibbonTitleType">
    <xs:attribute name="description" type="xs:string" use="required" />
    <xs:attribute name="languagecode" type="xs:int" use="required" />
</xs:complexType>
</xs:schema>
```

See also

[Customize commands and the ribbon](#)

[Ribbon core schema](#)

[Ribbon WSS schema](#)

[Customization XML reference](#)

Ribbon WSS schema

Article • 11/30/2022

The following is the schema definition for the ribbon types WSS of an import/export customization file. Ribbon WSS is included from the [Ribbon Core Schema](#). You can find schema in the `Schemas\9.0.0.2090\RibbonWSS.xsd` folder when you download the Schemas zip file.

Download the [Schemas](#).

For more information, see [Package and Distribute Extensions with Solutions](#).

Schema

XML

```
<?xml version="1.0" encoding="utf-8"?>
<xs:schema id="CrmRibbonWss" xmlns:xs="https://www.w3.org/2001/XMLSchema">

  <xs:simpleType name="AltType">
    <xs:restriction base="xs:string" />
  </xs:simpleType>

  <xs:simpleType name="ClassNameType">
    <xs:restriction base="xs:string" />
  </xs:simpleType>

  <xs:simpleType name="ContextualColorType">
    <xs:restriction base="xs:string">
      <xs:enumeration value="None" />
      <xs:enumeration value="DarkBlue" />
      <xs:enumeration value="LightBlue" />
      <xs:enumeration value="Teal" />
      <xs:enumeration value="Orange" />
      <xs:enumeration value="Green" />
      <xs:enumeration value="Magenta" />
      <xs:enumeration value="Yellow" />
      <xs:enumeration value="Purple" />
    </xs:restriction>
  </xs:simpleType>

  <xs:simpleType name="CommandType">
    <xs:restriction base="xs:string" />
  </xs:simpleType>

  <xs:simpleType name="CommandTypeType">
    <xs:restriction base="xs:string" >
      <xs:enumeration value="General" />
    </xs:restriction>
  </xs:simpleType>
```

```
<xs:enumeration value="OptionSelection" />
<xs:enumeration value="IgnoredByMenu" />
</xs:restriction>
</xs:simpleType>

<xs:simpleType name="DescriptionType">
  <xs:restriction base="xs:string" />
</xs:simpleType>

<xs:simpleType name="DisplayModeType">
  <xs:restriction base="xs:string">
    <xs:enumeration value="Default" />
    <xs:enumeration value="Small" />
    <xs:enumeration value="Medium" />
    <xs:enumeration value="Large" />
    <xs:enumeration value="Text" />
    <xs:enumeration value="Menu" />
    <xs:enumeration value="Menu16" />
    <xs:enumeration value="Menu32" />
    <xs:enumeration value="Thin" />
  </xs:restriction>
</xs:simpleType>

<xs:simpleType name="ElementDimensionsType">
  <xs:restriction base="xs:string">
    <xs:enumeration value="Size16by16" />
    <xs:enumeration value="Size32by32" />
    <xs:enumeration value="Size48by48" />
    <xs:enumeration value="Size64by48" />
    <xs:enumeration value="Size72by96" />
    <xs:enumeration value="Size96by72" />
    <xs:enumeration value="Size96by96" />
    <xs:enumeration value="Size128by128" />
    <xs:enumeration value="Size190by30" />
    <xs:enumeration value="Size190by40" />
    <xs:enumeration value="Size190by50" />
    <xs:enumeration value="Size190by60" />
  </xs:restriction>
</xs:simpleType>

<xs:simpleType name="HTMLType">
  <xs:restriction base="xs:string" />
</xs:simpleType>

<xs:simpleType name="IdType">
  <xs:restriction base="xs:string" />
</xs:simpleType>

<xs:simpleType name="ImageClassType">
  <xs:restriction base="xs:string" />
</xs:simpleType>

<xs:simpleType name="ImagePositionType">
  <xs:restriction base="xs:nonPositiveInteger" />
</xs:simpleType>
```

```
<xs:simpleType name="ImageUrlType">
  <xs:restriction base="xs:string" />
</xs:simpleType>

<xs:simpleType name="LabelCssType">
  <xs:restriction base="xs:string" />
</xs:simpleType>

<xs:simpleType name="LabelTextType">
  <xs:restriction base="xs:string" />
</xs:simpleType>

<xs:simpleType name="MenuItemIdType">
  <xs:restriction base="xs:string" />
</xs:simpleType>

<xs:simpleType name="ModernCommandTypeType">
  <xs:restriction base="xs:string">
    <xs:enumeration value="ControlCommand" />
    <xs:enumeration value="System" />
  </xs:restriction>
</xs:simpleType>

<xs:simpleType name="PixelLengthType">
  <xs:restriction base="xs:string" />
</xs:simpleType>

<xs:simpleType name="SectionTypeType">
  <xs:restriction base="xs:string">
    <xs:enumeration value="Divider" />
    <xs:enumeration value="OneRow" />
    <xs:enumeration value="TwoRow" />
    <xs:enumeration value="ThreeRow" />
  </xs:restriction>
</xs:simpleType>

<xs:simpleType name="SectionAlignmentType">
  <xs:restriction base="xs:string">
    <xs:enumeration value="Top" />
    <xs:enumeration value="Middle" />
  </xs:restriction>
</xs:simpleType>

<xs:simpleType name="SequenceType">
  <xs:restriction base="xs:integer" />
</xs:simpleType>

<xs:simpleType name="PriorityType">
  <xs:restriction base="xs:integer" />
</xs:simpleType>

<xs:simpleType name="SizeType">
  <xs:restriction base="xs:string" />
</xs:simpleType>
```

```
<xs:simpleType name="TemplateType">
  <xs:restriction base="xs:string" />
</xs:simpleType>

<xs:simpleType name="TemplateAliasType">
  <xs:restriction base="xs:string" />
</xs:simpleType>

<xs:simpleType name="TextDirectionType">
  <xs:restriction base="xs:string" />
</xs:simpleType>

<xs:simpleType name="TitleType">
  <xs:restriction base="xs:string" />
</xs:simpleType>

<xs:simpleType name="UnitNameType">
  <xs:restriction base="xs:string" />
</xs:simpleType>

<xs:simpleType name="ValueType">
  <xs:restriction base="xs:decimal" />
</xs:simpleType>

<xs:complexType name="ButtonType">
  <xs:attribute name="Alt" type="AltType" />
  <xs:attribute name="Command" type="CommandType" />
  <xs:attribute name="CommandType" type="CommandTypeType" />
  <xs:attribute name="CommandValueId" type="xs:string" />
  <xs:attribute name="Description" type="xs:string" />
  <xs:attribute name="Id" type="IdType" use="required" />
  <xs:attribute name="Image32by32" type="ImageUrlType" />
  <xs:attribute name="Image32by32Class" type="ImageClassType" />
  <xs:attribute name="Image32by32Left" type="ImagePositionType" />
  <xs:attribute name="Image32by32Top" type="ImagePositionType" />
  <xs:attribute name="Image16by16" type="ImageUrlType" />
  <xs:attribute name="Image16by16Class" type="ImageClassType" />
  <xs:attribute name="Image16by16Left" type="ImagePositionType" />
  <xs:attribute name="Image16by16Top" type="ImagePositionType" />
  <xs:attribute name="LabelCss" type="LabelCssType" />
  <xs:attribute name="LabelText" type="LabelTextType" />
  <xs:attribute name="MenuItemId" type="MenuItemIdType" />
  <xs:attribute name="ModernCommandType" type="ModernCommandTypeType" />
  <xs:attribute name="Sequence" type="SequenceType" />
  <xs:attribute name="Priority" type="PriorityType" />
  <xs:attribute name="TemplateAlias" type="TemplateAliasType" />
  <xs:attribute name="ToolTipImage32by32" type="ImageUrlType" />
  <xs:attribute name="ToolTipImage32by32Class" type="ImageClassType" />
  <xs:attribute name="ToolTipImage32by32Left" type="ImagePositionType" />
  <xs:attribute name="ToolTipImage32by32Top" type="ImagePositionType" />
  <xs:attribute name="ToolTipTitle" type="xs:string" />
  <xs:attribute name="ToolTipDescription" type="xs:string" />
  <xs:attribute name="ToolTipHelpKeyWord" type="xs:string" />
  <xs:attribute name="ToolTipShortcutKey" type="xs:string" />
```

```

<xs:attribute name="ModernImage" type="xs:string" />
</xs:complexType>

<xs:complexType name="CheckBoxType">
<xs:attribute name="Alt" type="AltType" />
<xs:attribute name="Command" type="CommandType" />
<xs:attribute name="Id" type="IdType" use="required" />
<xs:attribute name="LabelText" type="LabelTextType" />
<xs:attribute name="QueryCommand" type="CommandType" />
<xs:attribute name="TemplateAlias" type="TemplateAliasType" />
<xs:attribute name="MenuItemId" type="MenuItemIdType" />
<xs:attribute name="Sequence" type="SequenceType" />
<xs:attribute name="ToolTipImage32by32" type="ImageUrlType" />
<xs:attribute name="ToolTipImage32by32Class" type="ImageClassType" />
<xs:attribute name="ToolTipImage32by32Left" type="ImagePositionType" />
<xs:attribute name="ToolTipImage32by32Top" type="ImagePositionType" />
<xs:attribute name="ToolTipTitle" type="xs:string" />
<xs:attribute name="ToolTipDescription" type="xs:string" />
<xs:attribute name="ToolTipHelpKeyWord" type="xs:string" />
<xs:attribute name="ToolTipShortcutKey" type="xs:string" />
</xs:complexType>

<xs:complexType name="ColorPickerType">
<xs:sequence>
<xs:element name="Colors" type="ColorStylesType" minOccurs="0"
maxOccurs="1" />
</xs:sequence>
<xs:attribute name="Command" type="CommandType" />
<xs:attribute name="CommandPreview" type="CommandType" />
<xs:attribute name="CommandRevert" type="CommandType" />
<xs:attribute name="Id" type="IdType" use="required" />
<xs:attribute name="QueryCommand" type="CommandType" />
<xs:attribute name="Sequence" type="SequenceType" />
<xs:attribute name="TemplateAlias" type="TemplateAliasType" />
</xs:complexType>

<xs:complexType name="ColorStylesType">
<xs:sequence>
<xs:element name="Color" type="ColorStyleType" minOccurs="0"
maxOccurs="unbounded" />
</xs:sequence>
</xs:complexType>

<xs:complexType name="ColorStyleType">
<xs:attribute name="Sequence" type="SequenceType" />
<xs:attribute name="Title" type="AltType" />
<xs:attribute name="Style" type="xs:string" />
<xs:attribute name="Color" type="xs:string" />
<xs:attribute name="DisplayColor" type="xs:string" />
</xs:complexType>

<xs:complexType name="ComboBoxType">
<xs:sequence>
<xs:element name="Menu" type="MenuType" minOccurs="0" maxOccurs="1"/>
</xs:sequence>

```

```

<xs:attribute name="AllowFreeForm" type="xs:boolean" default="false" />
<xs:attribute name="AltArrow" type="AltType" />
<xs:attribute name="Alt" type="AltType" />
<xs:attribute name="AutoComplete" type="xs:boolean" default="true" />
<xs:attribute name="AutoCompleteDelay" type="xs:integer" default="100"
/>
<xs:attribute name="CacheMenuVersions" type="xs:boolean" default="false"
/>
<xs:attribute name="Command" type="CommandType" />
<xs:attribute name="CommandMenuOpen" type="CommandType" />
<xs:attribute name="CommandMenuClose" type="CommandType" />
<xs:attribute name="CommandPreview" type="CommandType" />
<xs:attribute name="CommandPreviewRevert" type="CommandType" />
<xs:attribute name="ImeEnabled" type="xs:boolean" />
<xs:attribute name="InitialItem" type="MenuItemIdType"/>
<xs:attribute name="Id" type="IdType" use="required" />
<xs:attribute name="QueryCommand" type="CommandType" />
<xs:attribute name="PopulateDynamically" type="xs:boolean"
default="false" />
<xs:attribute name="PopulateQueryCommand" type="CommandType" />
<xs:attribute name="PopulateOnlyOnce" type="xs:boolean" default="false"
/>
<xs:attribute name="Sequence" type="SequenceType" />
<xs:attribute name="TemplateAlias" type="TemplateAliasType" />
<xs:attribute name="ToolTipImage32by32" type="ImageUrlType" />
<xs:attribute name="ToolTipImage32by32Class" type="ImageClassType" />
<xs:attribute name="ToolTipImage32by32Left" type="ImagePositionType" />
<xs:attribute name="ToolTipImage32by32Top" type="ImagePositionType" />
<xs:attribute name="ToolTipTitle" type="xs:string" />
<xs:attribute name="ToolTipDescription" type="xs:string" />
<xs:attribute name="ToolTipHelpKeyWord" type="xs:string" />
<xs:attribute name="ToolTipSelectedItemTitle" type="xs:string" />
<xs:attribute name="ToolTipShortcutKey" type="xs:string" />
<xs:attribute name="Width" type="PixelLengthType" />
</xs:complexType>

<xs:complexType name="CommandUIType">
<xs:sequence>
<xs:choice minOccurs="1" maxOccurs="unbounded">
<xs:element name="Ribbon" type="RibbonType" minOccurs="0"
maxOccurs="1" />
<xs:element name="QAT" type="QATTType" minOccurs="0" maxOccurs="1" />
<xs:element name="Jewel" type="JewelType" minOccurs="0"
maxOccurs="1" />
<xs:element name="Templates" type="TemplatesType" minOccurs="0"
maxOccurs="1" />
</xs:choice>
</xs:sequence>
</xs:complexType>

<xs:element name="CommandUI" type="CommandUIType">
</xs:element>

<xs:complexType name="ContextualGroupType">
<xs:sequence>

```

```

        <xs:element name="Tab" type="TabType" minOccurs="0"
maxOccurs="unbounded" />
    </xs:sequence>
    <xs:attribute name="Color" type="ContextualColorType" />
    <xs:attribute name="Command" type="CommandType" />
    <xs:attribute name="ContextualGroupId" type="xs:string" />
    <xs:attribute name="Id" type="IdType" use="required" />
    <xs:attribute name="Sequence" type="SequenceType" />
    <xs:attribute name="Title" type="TitleType" />
</xs:complexType>

<xs:complexType name="ContextualTabsType">
    <xs:sequence>
        <xs:element name="ContextualGroup" type="ContextualGroupType"
minOccurs="0" maxOccurs="unbounded"/>
    </xs:sequence>
    <xs:attribute name="Id" type="IdType" use="required" />
</xs:complexType>

<xs:complexType name="ControlRefType">
    <xs:attribute name="DisplayMode" type="DisplayModeType" />
    <xs:attribute name="TemplateAlias" type="TemplateAliasType" />
</xs:complexType>

<xs:complexType name="ControlsType">
    <xs:sequence>
        <xs:choice minOccurs="0" maxOccurs="unbounded">
            <xs:element name="Button" type="ButtonType" minOccurs="0"
maxOccurs="unbounded" />
            <xs:element name="CheckBox" type="CheckBoxType" minOccurs="0"
maxOccurs="unbounded" />
            <xs:element name="ComboBox" type="ComboBoxType" minOccurs="0"
maxOccurs="unbounded" />
            <xs:element name="DropDown" type="DropDownType" minOccurs="0"
maxOccurs="unbounded" />
            <xs:element name="FlyoutAnchor" type="FlyoutAnchorType"
minOccurs="0" maxOccurs="unbounded" />
            <xs:element name="GalleryButton" type="GalleryButtonType"
minOccurs="0" maxOccurs="unbounded" />
            <xs:element name="Label" type="LabelType" minOccurs="0"
maxOccurs="unbounded" />
            <xs:element name="MRUSplitButton" type="MRUSplitButtonType"
minOccurs="0" maxOccurs="unbounded" />
            <xs:element name="Spinner" type="SpinnerType" minOccurs="0"
maxOccurs="unbounded" />
            <xs:element name="SplitButton" type="SplitButtonType" minOccurs="0"
maxOccurs="unbounded" />
            <xs:element name="TextBox" type="TextBoxType" minOccurs="0"
maxOccurs="unbounded" />
            <xs:element name="ToggleButton" type="ToggleButtonType"
minOccurs="0" maxOccurs="unbounded" />
        </xs:choice>
    </xs:sequence>
    <xs:attribute name="Id" type="IdType" use="required" />
    <xs:attribute name="ToolTipImage32by32" type="ImageUrlType" />

```

```

<xs:attribute name="ToolTipImage32by32Class" type="ImageClassType" />
<xs:attribute name="ToolTipImage32by32Left" type="ImagePositionType" />
<xs:attribute name="ToolTipImage32by32Top" type="ImagePositionType" />
<xs:attribute name="ToolTipTitle" type="xs:string" />
<xs:attribute name="ToolTipDescription" type="xs:string" />
<xs:attribute name="ToolTipHelpKeyWord" type="xs:string" />
<xs:attribute name="ToolTipShortcutKey" type="xs:string" />
<xs:attribute name="ToolTipSelectedItemTitle" type="xs:string" />
</xs:complexType>

<xs:complexType name="DropDownType">
  <xs:sequence>
    <xs:element name="Menu" type="MenuType" minOccurs="0" maxOccurs="1"/>
  </xs:sequence>
  <xs:attribute name="AltArrow" type="AltType" />
  <xs:attribute name="Alt" type="AltType" />
  <xs:attribute name="CacheMenuVersions" type="xs:boolean" default="false" />
  <xs:attribute name="Command" type="CommandType" />
  <xs:attribute name="CommandMenuOpen" type="CommandType" />
  <xs:attribute name="CommandMenuClose" type="CommandType" />
  <xs:attribute name="CommandPreview" type="CommandType" />
  <xs:attribute name="CommandPreviewRevert" type="CommandType" />
  <xs:attribute name="InitialItem" type="MenuItemIdType" />
  <xs:attribute name="Id" type="IdType" use="required" />
  <xs:attribute name="PopulateDynamically" type="xs:boolean" default="false" />
  <xs:attribute name="PopulateQueryCommand" type="CommandType" />
  <xs:attribute name="PopulateOnlyOnce" type="xs:boolean" default="false" />
  <xs:attribute name="QueryCommand" type="CommandType" />
  <xs:attribute name="Sequence" type="SequenceType" />
  <xs:attribute name="TemplateAlias" type="TemplateAliasType" />
  <xs:attribute name="ToolTipImage32by32" type="ImageUrlType" />
  <xs:attribute name="ToolTipImage32by32Class" type="ImageClassType" />
  <xs:attribute name="ToolTipImage32by32Left" type="ImagePositionType" />
  <xs:attribute name="ToolTipImage32by32Top" type="ImagePositionType" />
  <xs:attribute name="ToolTipTitle" type="xs:string" />
  <xs:attribute name="ToolTipDescription" type="xs:string" />
  <xs:attribute name="ToolTipHelpKeyWord" type="xs:string" />
  <xs:attribute name="ToolTipShortcutKey" type="xs:string" />
  <xs:attribute name="ToolTipSelectedItemTitle" type="xs:string" />
  <xs:attribute name="Width" type="PixelLengthType" />
  <xs:attribute name="SelectedItemDisplayStyle" type="DisplayStyleType" />
</xs:complexType>

<xs:complexType name="FlyoutAnchorType">
  <xs:sequence>
    <xs:element name="Menu" type="MenuType" minOccurs="0" maxOccurs="1"/>
  </xs:sequence>
  <xs:attribute name="Alt" type="AltType" />
  <xs:attribute name="CacheMenuVersions" type="xs:boolean" default="false" />
  <xs:attribute name="Command" type="CommandType" />
  <xs:attribute name="CommandType" type="CommandTypeType" />

```

```

<xs:attribute name="CommandMenuClose" type="CommandType" />
<xs:attribute name="Id" type="IdType" use="required" />
<xs:attribute name="Image16by16" type="ImageUrlType" />
<xs:attribute name="Image16by16Class" type="ImageClassType" />
<xs:attribute name="Image16by16Left" type="ImagePositionType" />
<xs:attribute name="Image16by16Top" type="ImagePositionType" />
<xs:attribute name="Image32by32" type="ImageUrlType" />
<xs:attribute name="Image32by32Class" type="ImageClassType" />
<xs:attribute name="Image32by32Left" type="ImagePositionType" />
<xs:attribute name="Image32by32Top" type="ImagePositionType" />
<xs:attribute name="LabelText" type="LabelTextType" />
<xs:attribute name="ModernCommandType" type="ModernCommandTypeType" />
<xs:attribute name="PopulateDynamically" type="xs:boolean"
default="false" />
<xs:attribute name="PopulateQueryCommand" type="CommandType" />
<xs:attribute name="PopulateOnlyOnce" type="xs:boolean" default="false"
/>
<xs:attribute name="Sequence" type="SequenceType" />
<xs:attribute name="TemplateAlias" type="TemplateAliasType" />
<xs:attribute name="ToolTipImage32by32" type="ImageUrlType" />
<xs:attribute name="ToolTipImage32by32Class" type="ImageClassType" />
<xs:attribute name="ToolTipImage32by32Left" type="ImagePositionType" />
<xs:attribute name="ToolTipImage32by32Top" type="ImagePositionType" />
<xs:attribute name="ToolTipTitle" type="xs:string" />
<xs:attribute name="ToolTipDescription" type="xs:string" />
<xs:attribute name="ToolTipHelpKeyWord" type="xs:string" />
<xs:attribute name="ToolTipSelectedItemTitle" type="xs:string" />
<xs:attribute name="ToolTipShortcutKey" type="xs:string" />
<xs:attribute name="ModernImage" type="xs:string" />
</xs:complexType>

<xs:complexType name="GalleryType">
<xs:sequence>
<xs:element name="GalleryButton" type="GalleryButtonType"
minOccurs="1" maxOccurs="unbounded" />
</xs:sequence>
<xs:attribute name="Command" type="CommandType" />
<xs:attribute name="CommandPreview" type="CommandType" />
<xs:attribute name="CommandRevert" type="CommandType" />
<xs:attribute name="ElementDimensions" type="ElementDimensionsType"
use="required" />
<xs:attribute name="Id" type="IdType" use="required" />
<xs:attribute name="QueryCommand" type="CommandType" />
<xs:attribute name="Sequence" type="SequenceType" />
<xs:attribute name="Width" type="xs:integer" use="required" />
</xs:complexType>

<xs:complexType name="GalleryButtonType">
<xs:attribute name="Alt" type="AltType" />
<xs:attribute name="Command" type="CommandType" />
<xs:attribute name="CommandPreview" type="CommandType" />
<xs:attribute name="CommandRevert" type="CommandType" />
<xs:attribute name="CommandType" type="CommandTypeType" />
<xs:attribute name="CommandValueId" type="xs:string" />
<xs:attribute name="ElementDimensions" type="ElementDimensionsType" />

```

```

<xs:attribute name="Id" type="IdType" use="required" />
<xs:attribute name="Image" type="ImageUrlType" />
<xs:attribute name="ImageClass" type="ImageClassType" />
<xs:attribute name="ImageLeft" type="ImagePositionType" />
<xs:attribute name="ImageTop" type="ImagePositionType" />
<xs:attribute name="InnerHTML" type="HTMLType" />
<xs:attribute name="MenuItemId" type="MenuItemIdType" />
<xs:attribute name="QueryCommand" type="CommandType" />
<xs:attribute name="Sequence" type="SequenceType" />
<xs:attribute name="TemplateAlias" type="TemplateAliasType" />
<xs:attribute name="ToolTipImage32by32" type="ImageUrlType" />
<xs:attribute name="ToolTipImage32by32Class" type="ImageClassType" />
<xs:attribute name="ToolTipImage32by32Left" type="ImagePositionType" />
<xs:attribute name="ToolTipImage32by32Top" type="ImagePositionType" />
<xs:attribute name="ToolTipTitle" type="xs:string" />
<xs:attribute name="ToolTipDescription" type="xs:string" />
<xs:attribute name="ToolTipHelpKeyWord" type="xs:string" />
<xs:attribute name="ToolTipShortcutKey" type="xs:string" />
</xs:complexType>

<xs:complexType name="GroupTemplateType">
  <xs:sequence>
    <xs:element name="Layout" type="LayoutType" minOccurs="0"
maxOccurs="unbounded" />
  </xs:sequence>
  <xs:attribute name="Id" type="IdType" use="required" />
  <xs:attribute name="ClassName" type="ClassNameType" />
</xs:complexType>

<xs:complexType name="GroupsType">
  <xs:sequence>
    <xs:element name="Group" type="GroupType" minOccurs="0"
maxOccurs="unbounded" />
  </xs:sequence>
  <xs:attribute name="Id" type="IdType" use="required" />
</xs:complexType>

<xs:complexType name="GroupType">
  <xs:all>
    <xs:element name="Controls" type="ControlsType" minOccurs="1"
maxOccurs="1" />
  </xs:all>
  <xs:attribute name="Command" type="CommandType" />
  <xs:attribute name="Description" type="DescriptionType" />
  <xs:attribute name="Id" type="IdType" use="required" />
  <xs:attribute name="Image32by32Popup" type="ImageUrlType" />
  <xs:attribute name="Image32by32PopupClass" type="ImageClassType" />
  <xs:attribute name="Image32by32PopupLeft" type="ImagePositionType" />
  <xs:attribute name="Image32by32PopupTop" type="ImagePositionType" />
  <xs:attribute name="PopupWidth" type="PixelLengthType" />
  <xs:attribute name="Sequence" type="SequenceType" />
  <xs:attribute name="Template" type="TemplateType" />
  <xs:attribute name="Title" type="TitleType" />
</xs:complexType>

```

```

<xs:complexType name="InsertTableType">
  <xs:attribute name="Alt" type="AltType" />
  <xs:attribute name="Command" type="CommandType" />
  <xs:attribute name="CommandType" type="CommandTypeType" />
  <xs:attribute name="CommandPreview" type="CommandType" />
  <xs:attribute name="CommandRevert" type="CommandType" />
  <xs:attribute name="Id" type="IdType" use="required" />
  <xs:attribute name="MenuSectionInitialTitle" type="xs:string" />
  <xs:attribute name="MenuSectionTitle" type="xs:string" />
  <xs:attribute name="Sequence" type="SequenceType" />
</xs:complexType>

<xs:complexType name="JewelType">
  <xs:sequence>
    <xs:element name="Menu" type="MenuType" minOccurs="0" maxOccurs="1"/>
  </xs:sequence>
  <xs:attribute name="Alt" type="AltType" />
  <xs:attribute name="CacheMenuVersions" type="xs:boolean" default="false" />
  <xs:attribute name="Command" type="CommandType" />
  <xs:attribute name="CommandMenuClose" type="CommandType" />
  <xs:attribute name="CommandMenuOpen" type="CommandType" />
  <xs:attribute name="Height" type="xs:integer" />
  <xs:attribute name="Id" type="IdType" use="required" />
  <xs:attribute name="ImageDownArrow" type="ImageUrlType" />
  <xs:attribute name="ImageDownArrowClass" type="ImageClassType" />
  <xs:attribute name="ImageDownArrowLeft" type="ImagePositionType" />
  <xs:attribute name="ImageDownArrowTop" type="ImagePositionType" />
  <xs:attribute name="ImageSideArrow" type="ImageUrlType" />
  <xs:attribute name="ImageSideArrowClass" type="ImageClassType" />
  <xs:attribute name="ImageSideArrowLeft" type="ImagePositionType" />
  <xs:attribute name="ImageSideArrowTop" type="ImagePositionType" />
  <xs:attribute name="ImageUpArrow" type="ImageUrlType" />
  <xs:attribute name="ImageUpArrowClass" type="ImageClassType" />
  <xs:attribute name="ImageUpArrowLeft" type="ImagePositionType" />
  <xs:attribute name="ImageUpArrowTop" type="ImagePositionType" />
  <xs:attribute name="Image" type="ImageUrlType" />
  <xs:attribute name="ImageClass" type="ImageClassType" />
  <xs:attribute name="ImageLeft" type="ImagePositionType" />
  <xs:attribute name="ImageTop" type="ImagePositionType" />
  <xs:attribute name="ImageHover" type="ImageUrlType" />
  <xs:attribute name="ImageHoverClass" type="ImageClassType" />
  <xs:attribute name="ImageHoverLeft" type="ImagePositionType" />
  <xs:attribute name="ImageHoverTop" type="ImagePositionType" />
  <xs:attribute name="ImageDown" type="ImageUrlType" />
  <xs:attribute name="ImageDownClass" type="ImageClassType" />
  <xs:attribute name="ImageDownLeft" type="ImagePositionType" />
  <xs:attribute name="ImageDownTop" type="ImagePositionType" />
  <xs:attribute name="ImageLeftSide" type="ImageUrlType" />
  <xs:attribute name="ImageLeftSideClass" type="ImageClassType" />
  <xs:attribute name="ImageLeftSideLeft" type="ImagePositionType" />
  <xs:attribute name="ImageLeftSideTop" type="ImagePositionType" />
  <xs:attribute name="ImageLeftSideWidth" type="xs:integer" />
  <xs:attribute name="ImageLeftSideHover" type="ImageUrlType" />
  <xs:attribute name="ImageLeftSideHoverClass" type="ImageClassType" />

```

```

<xs:attribute name="ImageLeftSideHoverLeft" type="ImagePositionType" />
<xs:attribute name="ImageLeftSideHoverTop" type="ImagePositionType" />
<xs:attribute name="ImageLeftSideDown" type="ImageUrlType" />
<xs:attribute name="ImageLeftSideDownClass" type="ImageClassType" />
<xs:attribute name="ImageLeftSideDownLeft" type="ImagePositionType" />
<xs:attribute name="ImageLeftSideDownTop" type="ImagePositionType" />
<xs:attribute name="ImageRightSide" type="ImageUrlType" />
<xs:attribute name="ImageRightSideClass" type="ImageClassType" />
<xs:attribute name="ImageRightSideLeft" type="ImagePositionType" />
<xs:attribute name="ImageRightSideTop" type="ImagePositionType" />
<xs:attribute name="ImageRightSideWidth" type="xs:integer" />
<xs:attribute name="ImageRightSideHover" type="ImageUrlType" />
<xs:attribute name="ImageRightSideHoverClass" type="ImageClassType" />
<xs:attribute name="ImageRightSideHoverLeft" type="ImagePositionType" />
<xs:attribute name="ImageRightSideHoverTop" type="ImagePositionType" />
<xs:attribute name="ImageRightSideDown" type="ImageUrlType" />
<xs:attribute name="ImageRightSideDownClass" type="ImageClassType" />
<xs:attribute name="ImageRightSideDownLeft" type="ImagePositionType" />
<xs:attribute name="ImageRightSideDownTop" type="ImagePositionType" />
<xs:attribute name="LabelCss" type="LabelCssType" />
<xs:attribute name="LabelText" type="LabelTextType" />
<xs:attribute name="PopulateDynamically" type="xs:boolean"
default="false" />
<xs:attribute name="PopulateQueryCommand" type="CommandType" />
<xs:attribute name="PopulateOnlyOnce" type="xs:boolean" default="false"
/>
</xs:complexType>

<xs:complexType name="LabelType">
<xs:attribute name="ForId" type="xs:string" />
<xs:attribute name="Id" type="IdType" use="required" />
<xs:attribute name="LabelText" type="LabelTextType" />
<xs:attribute name="Image16by16" type="ImageUrlType" />
<xs:attribute name="Image16by16Class" type="ImageClassType" />
<xs:attribute name="Image16by16Left" type="ImagePositionType" />
<xs:attribute name="Image16by16Top" type="ImagePositionType" />
<xs:attribute name="Sequence" type="SequenceType" />
<xs:attribute name="TemplateAlias" type="TemplateAliasType" />
<xs:attribute name="QueryCommand" type="CommandType" />
<xs:attribute name="Command" type="CommandType" />
<xs:attribute name="ModernImage" type="xs:string" />
</xs:complexType>

<xs:complexType name="LayoutType">
<xs:sequence>
<xs:choice minOccurs="0" maxOccurs="unbounded">
<xs:element name="Section" type="SectionType" minOccurs="0"
maxOccurs="unbounded" />
<xs:element name="OverflowSection" type="OverflowSectionType"
minOccurs="0" maxOccurs="unbounded" />
</xs:choice>
</xs:sequence>
<xs:attribute name="Title" type="TitleType" use="required" />
<xs:attribute name="LayoutTitle" type="TitleType" />
</xs:complexType>

```

```

<xs:complexType name="MaxSizeType">
  <xs:attribute name="GroupId" type="IdType" use="required" />
  <xs:attribute name="Id" type="IdType" use="required" />
  <xs:attribute name="Sequence" type="SequenceType" />
  <xs:attribute name="Size" type="SizeType" use="required" />
</xs:complexType>

<xs:complexType name="MenuType">
  <xs:sequence>
    <xs:element name="MenuSection" type="MenuSectionType" minOccurs="1"
maxOccurs="unbounded" />
  </xs:sequence>
  <xs:attribute name="Id" type="IdType" use="required" />
  <xs:attribute name="MaxWidth" type="PixelLengthType" />
</xs:complexType>

<xs:complexType name="MenuSectionType">
  <xs:choice minOccurs="1" maxOccurs="1">
    <xs:element name="Controls" type="MenuSectionControlsType"
minOccurs="1" maxOccurs="1" />
    <xs:element name="Gallery" type="GalleryType" minOccurs="1"
maxOccurs="1" />
  </xs:choice>
  <xs:attribute name="Id" type="IdType" use="required" />
  <xs:attribute name="Title" type="TitleType"/>
  <xs:attribute name="Scrollable" type="xs:boolean" default="false"/>
  <xs:attribute name="Sequence" type="SequenceType" />
  <xs:attribute name="MaxHeight" type="PixelLengthType" />
  <xs:attribute name="DisplayMode" type="DisplayModeType" default="Menu"
/>
</xs:complexType>

<xs:complexType name="MenuSectionControlsType">
  <xs:sequence>
    <xs:choice minOccurs="0" maxOccurs="unbounded">
      <xs:element name="ToggleButton" type="ToggleButtonType"
minOccurs="0" maxOccurs="unbounded" />
      <xs:element name="Button" type="ButtonType" minOccurs="0"
maxOccurs="unbounded" />
      <xs:element name="ColorPicker" type="ColorPickerType" minOccurs="0"
maxOccurs="unbounded" />
      <xs:element name="FlyoutAnchor" type="FlyoutAnchorType"
minOccurs="0" maxOccurs="unbounded" />
      <xs:element name="InsertTable" type="InsertTableType" minOccurs="0"
maxOccurs="unbounded" />
    </xs:choice>
  </xs:sequence>
  <xs:attribute name="Id" type="IdType" use="required" />
</xs:complexType>

<xs:complexType name="MRUSplitButtonType">
  <xs:sequence>
    <xs:element name="Menu" type="MenuType" minOccurs="0" maxOccurs="1"/>
  </xs:sequence>

```

```

<xs:attribute name="Alt" type="AltType" />
<xs:attribute name="CacheMenuVersions" type="xs:boolean" default="false"
/>
<xs:attribute name="Command" type="CommandType" />
<xs:attribute name="CommandMenuOpen" type="CommandType" />
<xs:attribute name="CommandMenuClose" type="CommandType" />
<xs:attribute name="CommandPreview" type="CommandType" />
<xs:attribute name="CommandPreviewRevert" type="CommandType" />
<xs:attribute name="Id" type="IdType" use="required" />
<xs:attribute name="InitialItem" type="MenuItemIdType" use="required" />
<xs:attribute name="MenuAlt" type="AltType" />
<xs:attribute name="MenuCommand" type="CommandType" />
<xs:attribute name="PopulateDynamically" type="xs:boolean"
default="false" />
<xs:attribute name="PopulateQueryCommand" type="CommandType" />
<xs:attribute name="PopulateOnlyOnce" type="xs:boolean" default="false"
/>
<xs:attribute name="Sequence" type="SequenceType" />
<xs:attribute name="QueryCommand" type="CommandType" />
<xs:attribute name="TemplateAlias" type="TemplateAliasType" />
<xs:attribute name="ToolTipImage32by32" type="ImageUrlType" />
<xs:attribute name="ToolTipImage32by32Class" type="ImageClassType" />
<xs:attribute name="ToolTipImage32by32Left" type="ImagePositionType" />
<xs:attribute name="ToolTipImage32by32Top" type="ImagePositionType" />
<xs:attribute name="ToolTipTitle" type="xs:string" />
<xs:attribute name="ToolTipDescription" type="xs:string" />
<xs:attribute name="ToolTipHelpKeyWord" type="xs:string" />
<xs:attribute name="ToolTipSelectedItemTitle" type="xs:string" />
<xs:attribute name="ToolTipShortcutKey" type="xs:string" />
</xs:complexType>

<xs:complexType name="LowScaleWarningType">
<xs:attribute name="Id" type="IdType" use="required" />
<xs:attribute name="Message" type="xs:string" />
<xs:attribute name="Sequence" type="SequenceType" />
</xs:complexType>

<xs:complexType name="OverflowAreaType">
<xs:attribute name="DisplayMode" type="DisplayModeType" />
<xs:attribute name="TemplateAlias" type="TemplateAliasType" />
</xs:complexType>

<xs:complexType name="OverflowSectionType">
<xs:attribute name="DisplayMode" type="DisplayModeType" />
<xs:attribute name="DividerAfter" type="xs:boolean" />
<xs:attribute name="DividerBefore" type="xs:boolean" />
<xs:attribute name="TemplateAlias" type="TemplateAliasType" />
<xs:attribute name="Type" type="SectionTypeType" />
</xs:complexType>

<xs:complexType name="QATTType">
<xs:sequence>
<xs:element name="Controls" type="ControlsType" minOccurs="1"
maxOccurs="1" />
</xs:sequence>

```

```

<xs:attribute name="Id" type="IdType" use="required" />
<xs:attribute name="ImageDownArrow" type="ImageUrlType" />
<xs:attribute name="ImageDownArrowClass" type="ImageClassType" />
<xs:attribute name="ImageDownArrowLeft" type="ImagePositionType" />
<xs:attribute name="ImageDownArrowTop" type="ImagePositionType" />
<xs:attribute name="ImageSideArrow" type="ImageUrlType" />
<xs:attribute name="ImageSideArrowClass" type="ImageClassType" />
<xs:attribute name="ImageSideArrowLeft" type="ImagePositionType" />
<xs:attribute name="ImageSideArrowTop" type="ImagePositionType" />
<xs:attribute name="ImageUpArrow" type="ImageUrlType" />
<xs:attribute name="ImageUpArrowClass" type="ImageClassType" />
<xs:attribute name="ImageUpArrowLeft" type="ImagePositionType" />
<xs:attribute name="ImageUpArrowTop" type="ImagePositionType" />
</xs:complexType>

<xs:complexType name="RibbonTemplatesType">
  <xs:sequence>
    <xs:element name="GroupTemplate" type="GroupTemplateType"
minOccurs="0" maxOccurs="unbounded" />
  </xs:sequence>
  <xs:attribute name="Id" type="IdType" use="required" />
</xs:complexType>

<xs:complexType name="RibbonType">
  <xs:sequence>
    <xs:element name="Tabs" type="TabsType" minOccurs="1" maxOccurs="1" />
    <xs:element name="ContextualTabs" type="ContextualTabsType"
minOccurs="1" maxOccurs="1" />
  </xs:sequence>
  <xs:attribute name="Id" type="IdType" use="required" />
  <xs:attribute name="Image32by32GroupPopupDefault" type="ImageUrlType" />
  <xs:attribute name="Image32by32GroupPopupDefaultClass"
type="ImageClassType" />
  <xs:attribute name="Image32by32GroupPopupDefaultLeft"
type="ImagePositionType" />
  <xs:attribute name="Image32by32GroupPopupDefaultTop"
type="ImagePositionType" />
  <xs:attribute name="ImageDownArrow" type="ImageUrlType" />
  <xs:attribute name="ImageDownArrowClass" type="ImageClassType" />
  <xs:attribute name="ImageDownArrowLeft" type="ImagePositionType" />
  <xs:attribute name="ImageDownArrowTop" type="ImagePositionType" />
  <xs:attribute name="ImageSideArrow" type="ImageUrlType" />
  <xs:attribute name="ImageSideArrowClass" type="ImageClassType" />
  <xs:attribute name="ImageSideArrowLeft" type="ImagePositionType" />
  <xs:attribute name="ImageSideArrowTop" type="ImagePositionType" />
  <xs:attribute name="ImageUpArrow" type="ImageUrlType" />
  <xs:attribute name="ImageUpArrowClass" type="ImageClassType" />
  <xs:attribute name="ImageUpArrowLeft" type="ImagePositionType" />
  <xs:attribute name="ImageUpArrowTop" type="ImagePositionType" />
  <xs:attribute name="RootEventCommand" type="CommandType" />
  <xs:attribute name="TabSwitchCommand" type="CommandType" />
  <xs:attribute name="ScaleCommand" type="CommandType" />
  <xs:attribute name="TextDirection" type="TextDirectionType" />
  <xs:attribute name="ToolTipFooterText" type="xs:string" />
  <xs:attribute name="ToolTipFooterImage16by16" type="ImageUrlType" />

```

```

        <xs:attribute name="ToolTipFooterImage16by16Class" type="ImageClassType"
/>
        <xs:attribute name="ToolTipFooterImage16by16Left"
type="ImagePositionType" />
        <xs:attribute name="ToolTipFooterImage16by16Top"
type="ImagePositionType" />
        <xs:attribute name="ToolTipDisabledCommandImage16by16"
type="ImageUrlType" />
        <xs:attribute name="ToolTipDisabledCommandImage16by16Class"
type="ImageClassType" />
        <xs:attribute name="ToolTipDisabledCommandImage16by16Left"
type="ImagePositionType" />
        <xs:attribute name="ToolTipDisabledCommandImage16by16Top"
type="ImagePositionType" />
        <xs:attribute name="ToolTipDisabledCommandDescription" type="xs:string"
/>
<xs:attribute name="ToolTipDisabledCommandTitle" type="xs:string" />
<xs:attribute name="ToolTipDisabledCommandHelpKey" type="xs:string" />
<xs:attribute name="ToolTipHelpCommand" type="xs:string" />
<xs:attribute name="ToolTipSelectedItemTitlePrefix" type="xs:string" />
<xs:attribute name="ShortcutKeyJumpToRibbon_Ctrl" type="xs:string" />
<xs:attribute name="ShortcutKeyJumpToRibbon_Alt" type="xs:string" />
<xs:attribute name="ShortcutKeyJumpToRibbon_Shift" type="xs:string" />
<xs:attribute name="ShortcutKeyJumpToRibbon_AccessKey" type="xs:string"
/>
<xs:attribute name="ShortcutKeyJumpToFirstControl_Ctrl" type="xs:string"
/>
<xs:attribute name="ShortcutKeyJumpToFirstControl_Alt" type="xs:string"
/>
<xs:attribute name="ShortcutKeyJumpToFirstControl_Shift"
type="xs:string" />
<xs:attribute name="ShortcutKeyJumpToFirstControl_AccessKey"
type="xs:string" />
<xs:attribute name="ATContextualTabText" type="xs:string" />
<xs:attribute name="ATTabPositionText" type="xs:string" />
<xs:attribute name="NavigationHelpText" type="xs:string" />
</xs:complexType>

<xs:complexType name="RowType">
<xs:sequence>
<xs:choice minOccurs="0" maxOccurs="unbounded">
<xs:element name="ControlRef" type="ControlRefType" minOccurs="0"
maxOccurs="unbounded" />
<xs:element name="Strip" type="StripType" minOccurs="0"
maxOccurs="unbounded" />
<xs:element name="OverflowArea" type="OverflowAreaType"
minOccurs="0" maxOccurs="unbounded" />
</xs:choice>
</xs:sequence>
</xs:complexType>

<xs:complexType name="ScaleType">
<xs:attribute name="GroupId" type="IdType" use="required" />
<xs:attribute name="Id" type="IdType" use="required" />
<xs:attribute name="Sequence" type="SequenceType" />

```

```

<xs:attribute name="Size" type="SizeType" use="required" />
<xs:attribute name="PopupSize" type="SizeType" />
</xs:complexType>

<xs:complexType name="ScalingType">
  <xs:sequence>
    <xs:element name="MaxSize" type="MaxSizeType" minOccurs="0"
maxOccurs="unbounded"/>
    <xs:choice minOccurs="0" maxOccurs="unbounded">
      <xs:element name="Scale" type="ScaleType" minOccurs="0"
maxOccurs="1"/>
      <xs:element name="LowScaleWarning" type="LowScaleWarningType"
minOccurs="0" maxOccurs="1"/>
    </xs:choice>
  </xs:sequence>
  <xs:attribute name="Id" type="IdType" use="required" />
</xs:complexType>

<xs:complexType name="SectionType">
  <xs:sequence>
    <xs:element name="Row" type="RowType" minOccurs="0" maxOccurs="3" />
  </xs:sequence>
  <xs:attribute name="Type" type="SectionTypeType" />
  <xs:attribute name="Alignment" type="SectionAlignmentType" />
</xs:complexType>

<xs:complexType name="SpinnerType">
  <xs:sequence>
    <xs:element name="Unit" type="UnitType" minOccurs="1"
maxOccurs="unbounded" />
  </xs:sequence>
  <xs:attribute name="AccelerationInterval" type="xs:integer" />
  <xs:attribute name="AltDownArrow" type="AltType" />
  <xs:attribute name="AltUpArrow" type="AltType" />
  <xs:attribute name="Command" type="CommandType" />
  <xs:attribute name="DefaultUnit" type="UnitNameType" />
  <xs:attribute name="DefaultValue" type="ValueType" />
  <xs:attribute name="Id" type="IdType" use="required" />
  <xs:attribute name="ImeEnabled" type="xs:boolean" />
  <xs:attribute name="MultiplierInterval" type="xs:integer" />
  <xs:attribute name="QueryCommand" type="CommandType" />
  <xs:attribute name="Sequence" type="SequenceType" />
  <xs:attribute name="TemplateAlias" type="TemplateAliasType" />
  <xs:attribute name="ToolTipImage32by32" type="ImageUrlType" />
  <xs:attribute name="ToolTipImage32by32Class" type="ImageClassType" />
  <xs:attribute name="ToolTipImage32by32Left" type="ImagePositionType" />
  <xs:attribute name="ToolTipImage32by32Top" type="ImagePositionType" />
  <xs:attribute name="ToolTipTitle" type="xs:string" />
  <xs:attribute name="ToolTipDescription" type="xs:string" />
  <xs:attribute name="ToolTipHelpKeyWord" type="xs:string" />
  <xs:attribute name="ToolTipShortcutKey" type="xs:string" />
</xs:complexType>

<xs:complexType name="SplitButtonType">
  <xs:sequence>

```

```

        <xs:element name="Menu" type="MenuType" minOccurs="0" maxOccurs="1"/>
    </xs:sequence>
    <xs:attribute name="Alt" type="AltType" />
    <xs:attribute name="CacheMenuVersions" type="xs:boolean" default="false"
/>
    <xs:attribute name="Command" type="CommandType" />
    <xs:attribute name="CommandMenuOpen" type="CommandType" />
    <xs:attribute name="CommandMenuClose" type="CommandType" />
    <xs:attribute name="CommandType" type="CommandTypeType" />
    <xs:attribute name="Id" type="IdType" use="required" />
    <xs:attribute name="Image32by32" type="ImageUrlType" />
    <xs:attribute name="Image32by32Class" type="ImageClassType" />
    <xs:attribute name="Image32by32Left" type="ImagePositionType" />
    <xs:attribute name="Image32by32Top" type="ImagePositionType" />
    <xs:attribute name="Image16by16" type="ImageUrlType" />
    <xs:attribute name="Image16by16Class" type="ImageClassType" />
    <xs:attribute name="Image16by16Left" type="ImagePositionType" />
    <xs:attribute name="Image16by16Top" type="ImagePositionType" />
    <xs:attribute name="LabelText" type="LabelTextType" />
    <xs:attribute name="MenuAlt" type="AltType" />
    <xs:attribute name="MenuCommand" type="CommandType" />
    <xs:attribute name="PopulateDynamically" type="xs:boolean"
default="false" />
    <xs:attribute name="PopulateQueryCommand" type="CommandType" />
    <xs:attribute name="PopulateOnlyOnce" type="xs:boolean" default="false"
/>
    <xs:attribute name="Sequence" type="SequenceType" />
    <xs:attribute name="TemplateAlias" type="TemplateAliasType" />
    <xs:attribute name="ToolTipImage32by32" type="ImageUrlType" />
    <xs:attribute name="ToolTipImage32by32Class" type="ImageClassType" />
    <xs:attribute name="ToolTipImage32by32Left" type="ImagePositionType" />
    <xs:attribute name="ToolTipImage32by32Top" type="ImagePositionType" />
    <xs:attribute name="ToolTipTitle" type="xs:string" />
    <xs:attribute name="ToolTipDescription" type="xs:string" />
    <xs:attribute name="ToolTipHelpKeyWord" type="xs:string" />
    <xs:attribute name="ToolTipSelectedItemTitle" type="xs:string" />
    <xs:attribute name="ToolTipShortcutKey" type="xs:string" />
    <xs:attribute name="ModernImage" type="xs:string" />
</xs:complexType>

<xs:complexType name="StripType">
    <xs:sequence>
        <xs:element name="ControlRef" type="ControlRefType" minOccurs="0"
maxOccurs="unbounded" />
    </xs:sequence>
</xs:complexType>

<xs:complexType name="TabType">
    <xs:sequence>
        <xs:element name="Scaling" type="ScalingType" minOccurs="1"
maxOccurs="1" />
        <xs:element name="Groups" type="GroupsType" minOccurs="1"
maxOccurs="1" />
    </xs:sequence>
    <xs:attribute name="Command" type="CommandType" />

```

```

<xs:attribute name="CssClass" type="ClassNameType" />
<xs:attribute name="Description" type="DescriptionType" />
<xs:attribute name="Id" type="IdType" use="required" />
<xs:attribute name="Sequence" type="SequenceType" />
<xs:attribute name="Title" type="TitleType" />
</xs:complexType>

<xs:complexType name="TabsType">
  <xs:sequence>
    <xs:element name="Tab" type="TabType" minOccurs="0"
maxOccurs="unbounded" />
  </xs:sequence>
  <xs:attribute name="Id" type="IdType" use="required" />
</xs:complexType>

<xs:complexType name="TemplatesType">
  <xs:all>
    <xs:element name="RibbonTemplates" type="RibbonTemplatesType" />
  </xs:all>
</xs:complexType>

<xs:complexType name="TextBoxType">
  <xs:attribute name="Command" type="CommandType" />
  <xs:attribute name="Id" type="IdType" use="required" />
  <xs:attribute name="IMeEnabled" type="xs:boolean" />
  <xs:attribute name="MaxLength" type="xs:integer" />
  <xs:attribute name="QueryCommand" type="CommandType" />
  <xs:attribute name="Sequence" type="SequenceType" />
  <xs:attribute name="ShowAsLabel" type="xs:boolean" />
  <xs:attribute name="TemplateAlias" type="TemplateAliasType" />
  <xs:attribute name="ToolTipImage32by32" type="ImageUrlType" />
  <xs:attribute name="ToolTipImage32by32Class" type="ImageClassType" />
  <xs:attribute name="ToolTipImage32by32Left" type="ImagePositionType" />
  <xs:attribute name="ToolTipImage32by32Top" type="ImagePositionType" />
  <xs:attribute name="ToolTipTitle" type="xs:string" />
  <xs:attribute name="ToolTipDescription" type="xs:string" />
  <xs:attribute name="ToolTipHelpKeyWord" type="xs:string" />
  <xs:attribute name="ToolTipShortcutKey" type="xs:string" />
  <xs:attribute name="Width" type="PixelLengthType" />
</xs:complexType>

<xs:complexType name="ToggleButtonType">
  <xs:attribute name="Alt" type="AltType" />
  <xs:attribute name="Command" type="CommandType" />
  <xs:attribute name="CommandValueId" type="xs:string" />
  <xs:attribute name="Description" type="xs:string" />
  <xs:attribute name="Id" type="IdType" use="required" />
  <xs:attribute name="LabelCss" type="LabelCssType" />
  <xs:attribute name="LabelText" type="LabelTextType" />
  <xs:attribute name="Image32by32" type="ImageUrlType" />
  <xs:attribute name="Image32by32Class" type="ImageClassType" />
  <xs:attribute name="Image32by32Left" type="ImagePositionType" />
  <xs:attribute name="Image32by32Top" type="ImagePositionType" />
  <xs:attribute name="Image16by16" type="ImageUrlType" />
  <xs:attribute name="Image16by16Class" type="ImageClassType" />
</xs:complexType>

```

```

<xs:attribute name="Image16by16Left" type="ImagePositionType" />
<xs:attribute name="Image16by16Top" type="ImagePositionType" />
<xs:attribute name="MenuItemId" type="MenuItemIdType" />
<xs:attribute name="QueryCommand" type="CommandType" />
<xs:attribute name="Sequence" type="SequenceType" />
<xs:attribute name="TemplateAlias" type="TemplateAliasType" />
<xs:attribute name="ToolTipImage32by32" type="ImageUrlType" />
<xs:attribute name="ToolTipImage32by32Class" type="ImageClassType" />
<xs:attribute name="ToolTipImage32by32Left" type="ImagePositionType" />
<xs:attribute name="ToolTipImage32by32Top" type="ImagePositionType" />
<xs:attribute name="ToolTipTitle" type="xs:string" />
<xs:attribute name="ToolTipDescription" type="xs:string" />
<xs:attribute name="ToolTipHelpKeyWord" type="xs:string" />
<xs:attribute name="ToolTipShortcutKey" type="xs:string" />
<xs:attribute name="ModernImage" type="xs:string" />
</xs:complexType>

<xs:complexType name="UnitType">
  <xs:sequence>
    <xs:element name="UnitAbbreviation" type="UnitAbbreviationType"
minOccurs="1" maxOccurs="unbounded" />
  </xs:sequence>
  <xs:attribute name="Name" type="UnitNameType" />
  <xs:attribute name="MinimumValue" type="ValueType" />
  <xs:attribute name="MaximumValue" type="ValueType" />
  <xs:attribute name="DecimalDigits" type="xs:integer" />
  <xs:attribute name="Interval" type="xs:double" />
</xs:complexType>

<xs:complexType name="UnitAbbreviationType">
  <xs:attribute name="Sequence" type="SequenceType" />
  <xs:attribute name="Value" type="UnitNameType" />
</xs:complexType>
</xs:schema>

```

See also

[Customize commands and the ribbon](#)

[Ribbon Types Schema](#)

[Customization XML Reference](#)

Form XML schema

Article • 11/30/2022

The following is the schema definition for form customizations for model-driven apps. More information: [Customize forms. Download the schemas](#).

Schema

XML

```
<?xml version="1.0"?>
<xs:schema xmlns:xs="https://www.w3.org/2001/XMLSchema"
    elementFormDefault="qualified"
    attributeFormDefault="unqualified">
    <xs:include schemaLocation="RibbonCore.xsd" />
    <xs:element name="form"
        type="FormType" />
    <xs:complexType name="ClientFileIncludeAttributeType">
        <xs:attribute name="src"
            use="required">
            <xs:simpleType>
                <xs:restriction base ="xs:string">
                    <xs:pattern value ="\$/webresource:|/)(.)+" />
                </xs:restriction>
            </xs:simpleType>
        </xs:attribute>
    </xs:complexType>
    <xs:complexType name="ClientResourcesType">
        <xs:all>
            <xs:element name="internalresources"
                minOccurs ="0"
                maxOccurs ="1">
                <xs:complexType>
                    <xs:all>
                        <xs:element name ="clientincludes"
                            minOccurs ="0"
                            maxOccurs ="1">
                            <xs:complexType >
                                <xs:choice minOccurs="0"
                                    maxOccurs = "100000" >
                                    <xs:element name ="internaljavascriptfile"
                                        type="ClientFileIncludeAttributeType"
                                        minOccurs ="0"
                                        maxOccurs ="1" />
                                    <xs:element name ="internalcssfile"
                                        type="ClientFileIncludeAttributeType"
                                        minOccurs ="0"
                                        maxOccurs ="1" />
                                </xs:choice>
                            </xs:complexType>
                        </xs:element>
                    </xs:all>
                </xs:element>
            </xs:all>
        </xs:element>
    </xs:all>
</xs:complexType>
```

```
</xs:complexType>
</xs:element>
<xs:element name = "clientvariables"
            minOccurs="0"
            maxOccurs = "1">
<xs:complexType>
    <xs:sequence>
        <xs:element name = "internaljavascriptvariable"
                    minOccurs = "0"
                    maxOccurs = "100000">
<xs:complexType>
    <xs:attribute name="name"
                  use = "required">
        <xs:simpleType>
            <xs:restriction base ="xs:string">
                <xs:pattern value="LOCID_([A-Za-z0-9_])+" />
                <xs:maxLength value = "32"/>
            </xs:restriction>
        </xs:simpleType>
    </xs:attribute>
    <xs:attribute name = "resourceid"
                  use="required" >
        <xs:simpleType>
            <xs:restriction base ="xs:string">
                <xs:pattern value="([A-Za-z0-9_.])+" />
            </xs:restriction>
        </xs:simpleType>
    </xs:attribute>

    </xs:complexType>
</xs:element>
</xs:sequence>

</xs:complexType>
</xs:element>
</xs:all>

</xs:complexType>
</xs:element>
<xs:element name="isvresources"
            minOccurs = "0"
            maxOccurs = "1">
<xs:complexType>
    <xs:sequence>
        <xs:element name = "clientincludes"
                    minOccurs = "0"
                    maxOccurs = "1">
<xs:complexType >
    <xs:sequence>
        <xs:element name = "webresource"
                    minOccurs = "0"
                    maxOccurs = "100000">
<xs:complexType>
    <xs:attribute name="path"
```

```
                type="xs:string"
                use ="required"/>
            <xs:attribute name="type"
                use ="required">
                <xs:simpleType>
                    <xs:restriction base="xs:string">
                        <xs:enumeration value="jscript"/>
                        <xs:enumeration value ="css" />
                    </xs:restriction>
                </xs:simpleType>
            </xs:attribute>

        </xs:complexType>
    </xs:element>
</xs:sequence>

    </xs:complexType>
</xs:element>
</xs:sequence>

</xs:complexType>
</xs:element>
</xs:all>

</xs:complexType>
<xs:complexType name="FormDisplayConditionsType">
    <xs:choice minOccurs="0"
        maxOccurs="1">
        <xs:element name="Everyone"
            minOccurs="1"
            maxOccurs="1">
            <xs:complexType>
                </xs:complexType>
            </xs:element>
            <xs:element name="Role"
                minOccurs="1"
                maxOccurs="unbounded">
                <xs:complexType>
                    <xs:attribute name="Id"
                        type="FormGuidType"
                        use="required" />

                </xs:complexType>
            </xs:element>
        </xs:choice>
        <xs:attribute name="FallbackForm"
            type="xs:boolean"
            use="optional" />
        <xs:attribute name="Order"
            type="xs:integer"
            use="optional" />

    </xs:complexType>
<xs:complexType name="FormLocalizedLabel">
```

```

<xs:attribute name="LCID"
              type="xs:integer" />
<xs:attribute name="Text"
              type="xs:string" />

</xs:complexType>
<xs:complexType name="FormLocalizedTitles">
  <xs:sequence minOccurs="1"
               maxOccurs="unbounded">
    <xs:element name="Title"
                type="FormLocalizedLabel" />
  </xs:sequence>

</xs:complexType>
<xs:complexType name="FormNavBarAreasType">
  <xs:sequence>
    <xs:element name="NavBarArea"
                minOccurs="0"
                maxOccurs="unbounded">
      <xs:complexType>
        <xs:sequence>
          <xs:element name="Titles"
                      minOccurs="1"
                      maxOccurs="1"
                      type="FormLocalizedTitles" />
        </xs:sequence>
        <xs:attribute name="Id"
                      type="xs:string"
                      use="required" />
      </xs:complexType>
    </xs:element>
  </xs:sequence>

</xs:complexType>
<xs:complexType name="FormNavBarType">
  <xs:choice minOccurs="0"
             maxOccurs="100000">
    <xs:element minOccurs="0"
                maxOccurs="1"
                name="NavBarItem">
      <xs:complexType>
        <xs:sequence>
          <xs:element name="Titles"
                      minOccurs="1"
                      maxOccurs="1"
                      type="FormLocalizedTitles" />
        </xs:sequence>
        <xs:attribute name="Icon"
                      type="xs:string"
                      use="required" />
        <xs:attribute name="Url"
                      type="xs:string"
                      use="required" />
        <xs:attribute name="Id"
                      type="xs:string"
                      use="required" />
      </xs:complexType>
    </xs:element>
  </xs:choice>
</xs:complexType>

```

```
        type="xs:string"
        use="required" />
<xs:attribute name="PassParams"
        type="FormCRM_Boolean"
        use="optional" />
<xs:attribute name="Sequence"
        type="xs:nonNegativeInteger"
        use="optional" />
<xs:attribute name="Area"
        type="xs:string"
        use="optional" />
<xs:attribute name="Client"
        type="xs:string"
        use="optional" />
<xs:attribute name="AvailableOffline"
        type="xs:boolean"
        use="optional" />

</xs:complexType>
</xs:element>
<xs:element name="NavBarByRelationshipItem"
            minOccurs="0"
            maxOccurs="1">
<xs:complexType>
    <xs:all>
        <xs:element name="Titles"
                    minOccurs="0"
                    maxOccurs="1"
                    type="FormLocalizedTitles" />
        <xs:element name="ToolTip"
                    minOccurs="0"
                    maxOccurs="1">
            <xs:complexType>
                <xs:sequence>
                    <xs:element name="Titles"
                                minOccurs="1"
                                maxOccurs="1"
                                type="FormLocalizedTitles" />
                </xs:sequence>
            </xs:complexType>
        </xs:element>
    </xs:all>
</xs:complexType>
</xs:element>
<xs:element name="Privileges"
            minOccurs="0"
            maxOccurs = "1">
<xs:complexType>
    <xs:sequence >
        <xs:element name = "Privilege"
                    minOccurs = "1"
                    maxOccurs = "100000">
            <xs:complexType>
                <xs:attribute name = "Entity"
                            type = "xs:string"
                            use = "required"/>
                <xs:attribute name="Privilege" type="xs:string" use="optional" />
            </xs:complexType>
        </xs:element>
    </xs:sequence>
</xs:complexType>
</xs:element>
```

```

                type ="xs:string"
                use ="required"/>

            </xs:complexType>
        </xs:element>
    </xs:sequence>

    </xs:complexType>
</xs:element>
</xs:all>
<xs:attribute name="RelationshipName"
              type="xs:string"
              use="required" />
<xs:attribute name="Id"
              type="xs:string"
              use="required" />
<xs:attribute name="Area"
              type="xs:string"
              use="optional" />
<xs:attribute name="TitleResourceId"
              type="xs:string"
              use="optional" />
<xs:attribute name="Client"
              type="xs:string"
              use="optional" />
<xs:attribute name="AvailableOffline"
              type="xs:boolean"
              use="optional" />
<xs:attribute name="Icon"
              type="xs:string"
              use="optional" />
<xs:attribute name="Sequence"
              type="xs:nonNegativeInteger"
              use="optional" />
<xs:attribute name="Show"
              type="xs:boolean"
              use="optional" />
<xs:attribute name="ViewId"
              type="FormISVGuid"
              use="optional" />

        </xs:complexType>
    </xs:element>
</xs:choice>
<xs:attribute name="ValidForCreate"
              type="FormCRM_Boolean"
              use="optional" />
<xs:attribute name="ValidForUpdate"
              type="FormCRM_Boolean"
              use="optional" />

</xs:complexType>
<xs:complexType name="FormNavigationType">
    <xs:all>
        <xs:element name="NavBarAreas"

```



```

<xs:attribute name="relationship"
    type="xs:string" />

    </xs:complexType>
</xs:element>
</xs:sequence>

</xs:complexType>
</xs:element>
<xs:element name="controlDescriptions"
    minOccurs="0"
    maxOccurs="1">
<xs:complexType>
<xs:sequence>
<xs:element name="controlDescription"
    minOccurs="0"
    maxOccurs="unbounded">
<xs:complexType>
<xs:sequence>
<xs:element name="customControl"
    minOccurs="0"
    maxOccurs="unbounded">
<xs:complexType>
<xs:sequence>
<xs:element name="parameters"
    minOccurs="0"
    maxOccurs="1">
<xs:complexType>
<xs:sequence>
<xs:any minOccurs="0"
    maxOccurs="unbounded"
    processContents="skip"></xs:any>
</xs:sequence>
</xs:complexType>
</xs:element>
</xs:sequence>
<xs:attribute name="id"
    type="FormGuidType"
    use="required" />
<xs:attribute name="formFactor"
    type="xs:integer"
    use="optional" />
<xs:attribute name="name"
    type="xs:string"
    use="optional" />
<xs:attribute name="version"
    type="xs:string"
    use="optional" />

    </xs:complexType>
</xs:element>
</xs:sequence>
<xs:attribute name="forControl"
    type="xs:string"
    use="required" />
```

```
</xs:complexType>
</xs:element>
</xs:sequence>

</xs:complexType>
</xs:element>
<xs:element name="tabs"
            minOccurs="1"
            maxOccurs="1">
<xs:complexType>
<xs:sequence>
<xs:element name="tab"
            minOccurs="1"
            maxOccurs="100">
<xs:complexType>
<xs:all>
<xs:element name="labels"
            type="FormXmlLabelsType"
            minOccurs="0"
            maxOccurs="1" />
<xs:element name="tabheader"
            type="FormXmlHeaderFooterType"
            minOccurs="0"
            maxOccurs="1"/>
<xs:element name="tabfooter"
            type="FormXmlHeaderFooterType"
            minOccurs="0"
            maxOccurs="1"/>
<xs:element name="columns"
            minOccurs="1"
            maxOccurs="1">
<xs:complexType>
<xs:sequence>
<xs:element name="column"
            minOccurs="1"
            maxOccurs="3">
<xs:complexType>
<xs:sequence>
<xs:element name="sections"
            minOccurs="0"
            maxOccurs="1">
<xs:complexType>
<xs:sequence>
<xs:element name="section"
            minOccurs="0"
            maxOccurs="unbounded">
<xs:complexType>
<xs:all>
<xs:element name="labels"
            type="FormXmlLabelsType"
            minOccurs="0"
            maxOccurs="1" />
<xs:element name="rows"
            type="FormXmlRowsType"
            minOccurs="0"
            maxOccurs="1" />
```

```
        minOccurs="0"
        maxOccurs="1">
    <xss:complexType>
        <xss:sequence>
            <xss:element name="row"
                minOccurs="0"

maxOccurs="unbounded">
    <xss:complexType>
        <xss:sequence>
            <xss:element
name="cell"

minOccurs="0"

maxOccurs="unbounded">
    <xss:complexType>
        <xss:all>
            <xss:element
name="labels"

type="FormXmlLabelsType"

minOccurs="0"

maxOccurs="1" />
            <xss:element
name="control"

type="FormXmlControlType"

minOccurs="0"

maxOccurs="1" />
            <xss:element
name="events"

type="FormXmlEventsType"

minOccurs="0"

maxOccurs="1" />
            </xss:all>
            <xss:attribute
name="auto"

type="xs:boolean" />
            <xss:attribute
name="addedby"

type="xs:string" />
            <xss:attributeGroup
ref="FormXmlCellCommon"/>
        </xss:complexType>
    </xss:element>
```

```

                </xs:sequence>
                <xs:attribute
name="addedby"
type="xs:string" />
            <xs:attributeGroup
ref="FormXmlRowCommon"/>
                </xs:complexType>
                </xs:element>
            </xs:sequence>
            <xs:attribute name="addedby"
type="xs:string" />
        />

                </xs:complexType>
                </xs:element>
            </xs:all>
            <xs:attribute name="group"
type="xs:string" />
            <xs:attribute name="name"
type="xs:string" />
            <xs:attribute name="showlabel"
type="xs:boolean" />
            <xs:attribute name="labelid"
type="FormGuidType"
use="optional" />
            <xs:attribute name="showbar"
type="xs:boolean" />
            <xs:attribute name="id"
type="FormGuidType" />
            <xs:attribute name="IsUserDefined"
type="xs:string" />
            <xs:attribute name="height"
type="xs:string" />
            <xs:attribute name="locklevel"
type="xs:nonNegativeInteger" />
            <xs:attribute name="layout"
type="xs:string" />
            <xs:attribute name="addedby"
type="xs:string" />
            <xs:attribute name="visible"
type="xs:boolean" />
            <xs:attribute
name="availableforphone"
type="xs:boolean" />
            <xs:attribute name="rowheight"
type="xs:nonNegativeInteger"
use="optional" />
            <xs:attribute name="autoexpand"
type="xs:boolean"
use="optional" />
        <xs:attributeGroup
ref="FormXmlSectionCommon"/>

```

```
                </xs:complexType>
            </xs:element>
        </xs:sequence>
        <xs:attribute name="addedby"
                      type="xs:string" />

                </xs:complexType>
            </xs:element>
        </xs:sequence>
        <xs:attribute name="width"
                      type="FormPercentageType"
                      use="required" />

                </xs:complexType>
            </xs:element>
        </xs:sequence>

                </xs:complexType>
            </xs:element>
            <xs:element name="events"
                        type="FormXmlEventsType"
                        minOccurs="0"
                        maxOccurs="1" />
        </xs:all>
        <xs:attribute name="group"
                      type="xs:string" />
        <xs:attribute name="name"
                      type="xs:string" />
        <xs:attribute name="verticallayout"
                      type="xs:boolean" />
        <xs:attribute name="showlabel"
                      type="xs:boolean" />
        <xs:attribute name="labelid"
                      type="FormGuidType"
                      use="optional" />
        <xs:attribute name="id"
                      type="FormGuidType" />
        <xs:attribute name="IsUserDefined"
                      type="xs:string" />
        <xs:attribute name="locklevel"
                      type="xs:nonNegativeInteger" />
        <xs:attribute name="addedby"
                      type="xs:string" />
        <xs:attribute name="expanded"
                      type="xs:boolean" />
        <xs:attribute name="visible"
                      type="xs:boolean" />
        <xs:attribute name="availableforphone"
                      type="xs:boolean" />
        <xs:attribute name="collapsible"
                      type="xs:boolean" />

                </xs:complexType>
            </xs:element>
```

```

</xs:sequence>
<xs:attribute name="showlabels"
              type="xs:boolean" />
<xs:attribute name="addedby"
              type="xs:string" />
<xs:attribute name="filterby"
              type="xs:string" />
<xs:attribute name="dashboardCategory"
              type="xs:string" />
<xs:attribute name="timeframe"
              type="xs:string" />
<xs:attribute name="primaryentitylogicalname"
              type="xs:string" />
<xs:attribute name="entityview"
              type="xs:string" />
<xs:attribute name="tilespresent"
              type="xs:boolean" />

</xs:complexType>
</xs:element>
<xs:element name="header"
            type="FormXmlHeaderFooterType"
            minOccurs="0"
            maxOccurs="1"/>
<xs:element name="footer"
            type="FormXmlHeaderFooterType"
            minOccurs="0"
            maxOccurs="1"/>
<xs:element name="events"
            type="FormXmlEventsType"
            minOccurs="0"
            maxOccurs="1" />
<xs:element name="formLibraries"
            type="FormXmlLibraryType"
            minOccurs="0"
            maxOccurs="1" />
<xs:element name="externaldependencies"
            type="FormXmlExternalDependenciesType"
            minOccurs="0"
            maxOccurs="1" />
<xs:element name="formparameters"
            type="FormParametersType"
            minOccurs="0"
            maxOccurs="1">
<xs:unique name="UniqueName">
    <xs:selector xpath = "./querystringparameter" />
    <xs:field xpath = "@name" />
</xs:unique>
</xs:element>
<xs:element name ="clientresources"
            type ="ClientResourcesType"
            minOccurs ="0"
            maxOccurs ="1"></xs:element>
<xs:element name="Navigation"
            type="FormNavigationType"

```

```

        minOccurs="0"
        maxOccurs="1"/>
<xss:element name="DisplayConditions"
        type="FormDisplayConditionsType"
        minOccurs="0"
        maxOccurs="1" />
<xss:element name="RibbonDiffXml"
        type="RibbonEntityDiffXmlType"
        minOccurs="0"
        maxOccurs="1" />
</xss:all>
<xss:attribute name="enablerelatedinformation"
        type="xs:boolean" />
<xss:attribute name="relatedInformationCollapsed"
        type="xs:boolean" />
<xss:attribute name="hasmargin"
        type="xs:boolean" />
<xss:attribute name="addedby"
        type="xs:string" />
<xss:attribute name="shownavigationbar"
        type="xs:boolean" />
<xss:attribute name="showImage"
        type="xs:boolean" />
<xss:attribute name="maxWidth"
        use="optional">
    <xss:simpleType>
        <xss:restriction base="xs:positiveInteger">
            <xss:minInclusive value="400" />
        </xss:restriction>
    </xss:simpleType>
</xss:attribute>

</xss:complexType>
<xss:complexType name="FormXmlControlType">
    <xss:sequence>
        <xss:element name="labels"
            type="FormXmlLabelsType"
            minOccurs="0"
            maxOccurs="1" />
        <xss:element name="parameters"
            minOccurs="0"
            maxOccurs="1">
            <xss:complexType>
                <xss:choice minOccurs="1"
                    maxOccurs="1">
                    <!-- LATER: (TobinZ, 2008-07-24) - Divide this list up into sets
that are valid together. -->
                    <xss:choice minOccurs="1"
                        maxOccurs="unbounded">
                        <xss:element name="Url"
                            type="xs:string"
                            minOccurs="0"
                            maxOccurs="1" />
                        <xss:element name="PassParameters"
                            type="xs:boolean"

```

```

        minOccurs="0"
        maxOccurs="1" />
<xss:element name="Security"
        type="xs:boolean"
        minOccurs="0"
        maxOccurs="1" />
<xss:element name="Scrolling"
        type="xs:string"
        minOccurs="0"
        maxOccurs="1" />
<xss:element name="Border"
        type="xs:string"
        minOccurs="0"
        maxOccurs="1" />
<xss:element name="Preload"
        type="xs:string"
        minOccurs="0"
        maxOccurs="1" />
<xss:element name="IsPassword"
        type="xs:boolean"
        minOccurs="0"
        maxOccurs="1" />
<xss:element name="IsColorValue"
        type="xs:boolean"
        minOccurs="0"
        maxOccurs="1" />
<!--Web Resource related parameters. Included in this section
since they
can include Url, PassParameters etc-->
<xss:element name="Height"
        type="xs:unsignedInt"
        minOccurs="0"
        maxOccurs="1"/>
<xss:element name="Width"
        type="xs:unsignedInt"
        minOccurs="0"
        maxOccurs="1"/>
<xss:element name="AltText"
        type="xs:string"
        minOccurs="0"
        maxOccurs="1"/>
<xss:element name="SizeType"
        type="WebResourceSizeType"
        minOccurs="0"
        maxOccurs="1"/>
<xss:element name="ShowInROF"
        type="xs:boolean"
        minOccurs="0"
        maxOccurs="1" />
<xss:element name="ShowOnMobileClient"
        type="xs:boolean"
        minOccurs="0"
        maxOccurs="1" />
<xss:element name="HorizontalAlignment"
        type="ImageHorizontalAlignmentType"

```

```
        minOccurs="0"
        maxOccurs="1"/>
<xss:element name="VerticalAlignment"
              type="ImageVerticalAlignmentType"
              minOccurs="0"
              maxOccurs="1"/>
<xss:element name="Data"
              type="xs:string"
              minOccurs="0"
              maxOccurs="1" />
<xss:element name="WebResourceId"
              type="xs:string"
              minOccurs="0"
              maxOccurs="1" />
<!--Parameters for Rich Editor Control-->
<xss:element name="ReadOnly"
              type="xs:boolean"
              minOccurs="0"
              maxOccurs="1" />
<xss:element name="ShowDialogs"
              type="xs:boolean"
              minOccurs="0"
              maxOccurs="1" />
<xss:element name="IsViewExpandable"
              type="xs:boolean"
              minOccurs="0"
              maxOccurs="1" />
<xss:element name="HideToolbar"
              type="xs:boolean"
              minOccurs="0"
              maxOccurs="1" />
<xss:element name="ToolbarJSON"
              type="xs:string"
              minOccurs="0"
              maxOccurs="1" />
<xss:element name="ExpandedToolbarJSON"
              type="xs:string"
              minOccurs="0"
              maxOccurs="1" />
<xss:element name="HiddenToolbarJSON"
              type="xs:string"
              minOccurs="0"
              maxOccurs="1" />
<xss:element name="ClassName"
              type="xs:string"
              minOccurs="0"
              maxOccurs="1" />
</xss:choice>
<!--Parameters for unbound lookup control-->
<xss:choice minOccurs="1"
            maxOccurs="unbounded">
<xss:element name="TargetEntities"
              minOccurs="0"
              maxOccurs="1">
<xss:complexType>
```

```

<xs:sequence>
    <xs:element name="TargetEntity"
        minOccurs="1"
        maxOccurs="unbounded">
        <xs:complexType>
            <xs:all>
                <xs:element name="EntityLogicalName"
                    type="xs:string"
                    minOccurs="1"
                    maxOccurs="1" />
                <xs:element name="DefaultViewId"
                    type="FormGuidType"
                    minOccurs="0"
                    maxOccurs="1" />
                <xs:element name="IsDeDupLookup"
                    type="xs:boolean"
                    minOccurs="0"
                    maxOccurs="1" />
                <xs:element name="UnboundLookupStyle"
                    type="xs:string"
                    minOccurs="0"
                    maxOccurs="1" />
            </xs:all>
        </xs:complexType>
    </xs:element>
</xs:sequence>
</xs:complexType>
</xs:element>
</xs:choice>
<!-- Parameters for the subgrid control and reference panel
subgrid control --&gt;
&lt;xs:choice minOccurs="1"
        maxOccurs="unbounded"&gt;
    &lt;xs:element name="ViewId"
        type="FormGuidType"
        minOccurs="0"
        maxOccurs="1" /&gt;
    &lt;xs:element name="IsUserView"
        type="xs:boolean"
        minOccurs="0"
        maxOccurs="1" /&gt;
    &lt;xs:element name="IsUserChart"
        type="xs:boolean"
        minOccurs="0"
        maxOccurs="1" /&gt;
    &lt;xs:element name="RelationshipName"
        type="xs:string"
        minOccurs="0"
        maxOccurs="1" /&gt;
    &lt;xs:element name="RelationshipRoleOrdinal"
        type="RelationshipRoleOrdinalType"
        minOccurs="0"
        maxOccurs="1" /&gt;
    &lt;xs:element name="TargetEntityType"
        type="xs:string"
</pre>

```

```
        minOccurs="1"
        maxOccurs="1" />
<xss:element name="AutoExpand"
        type="GridResizeType"
        minOccurs="0"
        maxOccurs="1" />
<xss:element name="RecordsPerPage"
        type="xs:unsignedShort"
        minOccurs="0"
        maxOccurs="1" />
<xss:element name="MaxRowsBeforeScroll"
        type="xs:integer"
        minOccurs="0"
        maxOccurs="1" />
<xss:element name="EnableQuickFind"
        type="xs:boolean"
        minOccurs="0"
        maxOccurs="1" />
<xss:element name="EnableJumpBar"
        type="xs:boolean"
        minOccurs="0"
        maxOccurs="1" />
<xss:element name="EnableViewPicker"
        type="xs:boolean"
        minOccurs="0"
        maxOccurs="1" />
<xss:element name="ViewIds"
        type="xs:string"
        minOccurs="0"
        maxOccurs="1" />
<xss:element name="ChartGridMode"
        type="xs:string"
        minOccurs="0"
        maxOccurs="1" />
<xss:element name="VisualizationId"
        type="xs:string"
        minOccurs="0"
        maxOccurs="1" />
<xss:element name="EnableChartPicker"
        type="xs:boolean"
        minOccurs="0"
        maxOccurs="1" />
<xss:element name="EnableContextualActions"
        type="xs:boolean"
        minOccurs="0"
        maxOccurs="1" />
<xss:element name="TeamTemplateId"
        type="FormGuidType"
        minOccurs="0"
        maxOccurs="1" />
<xss:element name="GridUIMode"
        type="xs:string"
        minOccurs="0"
        maxOccurs="1" />
<xss:element name="ReferencePanelSubgridIconUrl"
```

```
        type="xs:string"
        minOccurs="0"
        maxOccurs="1" />
    </xs:choice>
    <!-- Parameters for Power BI Tile control -->
    <xs:choice minOccurs="1"
        maxOccurs="unbounded">
        <xs:element name="PowerBIDashboardId"
            type="xs:string"
            minOccurs="0"
            maxOccurs="1" />
        <xs:element name="TileId"
            type="xs:string"
            minOccurs="0"
            maxOccurs="1" />
        <xs:element name="TileUrl"
            type="xs:string"
            minOccurs="0"
            maxOccurs="1" />
    </xs:choice>
    <!-- Parameters for the lookup control -->
    <xs:choice minOccurs="1"
        maxOccurs="unbounded">
        <xs:element name="DefaultViewId"
            type="FormGuidType"
            minOccurs="0"
            maxOccurs="1" />
        <xs:element name="FilterRelationshipName"
            type="xs:string"
            minOccurs="0"
            maxOccurs="1" />
        <xs:element name="DependentAttributeName"
            type="xs:string"
            minOccurs="0"
            maxOccurs="1" />
        <xs:element name="DependentAttributeType"
            type="xs:string"
            minOccurs="0"
            maxOccurs="1" />
        <xs:element name="AutoResolve"
            type="xs:boolean"
            minOccurs="0"
            maxOccurs="1" />
        <xs:element name="ResolveEmailAddress"
            type="xs:boolean"
            minOccurs="0"
            maxOccurs="1" />
        <xs:element name="DefaultViewReadOnly"
            type="xs:boolean"
            minOccurs="0"
            maxOccurs="1" />
        <xs:element name="ViewPickerReadOnly"
            type="xs:boolean"
            minOccurs="0"
            maxOccurs="1" />
```

```

<xs:element name="AllowFilterOff"
            type="xs:boolean"
            minOccurs="0"
            maxOccurs="1" />
<xs:element name="DisableMru"
            type="xs:boolean"
            minOccurs="0"
            maxOccurs="1" />
<xs:element name="DisableQuickFind"
            type="xs:boolean"
            minOccurs="0"
            maxOccurs="1" />
<xs:element name="DisableViewPicker"
            type="xs:boolean"
            minOccurs="0"
            maxOccurs="1" />
<xs:element name="AvailableViewIds"
            type="xs:string"
            minOccurs="0"
            maxOccurs="1" />
<xs:element name="EntityLogicalName"
            type="xs:string"
            minOccurs="0"
            maxOccurs="1" />
<xs:element name="IsInlineNewEnabled"
            type="xs:boolean"
            minOccurs="0"
            maxOccurs="1" />
<xs:element name="InlineViewIds"
            type="xs:string"
            minOccurs="0"
            maxOccurs="1" />
<xs:element name="UnboundLookupTypes"
            type="xs:string"
            minOccurs="0"
            maxOccurs="1" />
<xs:element name="UnboundLookupBrowse"
            type="xs:boolean"
            minOccurs="0"
            maxOccurs="1" />
<xs:element name="UnboundLookupControlType"
            type="xs:string"
            minOccurs="0"
            maxOccurs="1" />
<xs:element name="ShowAsBreadcrumbControl"
            type="xs:boolean"
            minOccurs="0"
            maxOccurs="1" />
</xs:choice>
<!-- Parameters for the TextBox -->
<xs:choice minOccurs="1"
            maxOccurs="unbounded">
<xs:element name="MaxLength"
            type="xs:integer"
            minOccurs="0"

```

```

                maxOccurs="1" />
            <xss:element name="Format"
                         type="FormatType"
                         minOccurs="0"
                         maxOccurs="1" />
        </xs:choice>
        <!-- Parameters for the Label -->
        <xs:choice minOccurs="1"
                   maxOccurs="unbounded">
            <xss:element name="IsTitle"
                         type="xs:boolean"
                         minOccurs="0"
                         maxOccurs="1" />
        </xs:choice>
        <!-- Parameters for the Numbers (i.e Whole, Decimal, Currency)-->
    >
    <xs:choice minOccurs="1"
               maxOccurs="unbounded">
        <xss:element name="MinValue"
                         type="xs:double"
                         minOccurs="0"
                         maxOccurs="1" />
        <xss:element name="MaxValue"
                         type="xs:double"
                         minOccurs="0"
                         maxOccurs="1" />
        <xss:element name="Precision"
                         type="xs:integer"
                         minOccurs="0"
                         maxOccurs="1" />
    </xs:choice>
    <!-- Parameters for the PickList Control and Two Value
Option(Radio) Control -->
    <xs:choice minOccurs="1"
               maxOccurs="unbounded">
        <xss:element name="DefaultValue"
                         type="xs:string"
                         minOccurs="0"
                         maxOccurs="1" />
        <xss:element name="OptionSetId"
                         type="FormGuidType"
                         minOccurs="0"
                         maxOccurs="1" />
    </xs:choice>
    <!-- Parameters for the quickformcollection control and
reference panel quick form collection control -->
    <xs:choice minOccurs="1"
               maxOccurs="unbounded">
        <xss:element name="QuickForms"
                         type="xs:string"
                         minOccurs="1"
                         maxOccurs="1" />
        <xss:element name="ControlMode"
                         type="xs:string"
                         minOccurs="0"

```

```

        maxOccurs="1" />
<xss:element name="ReferencePanelQuickFormCollectionIconUrl"
    type="xs:string"
    minOccurs="0"
    maxOccurs="1" />
<xss:element name="DisplayAsCustomer360Tile"
    type="xs:boolean"
    minOccurs="0"
    maxOccurs="1" />
</xss:choice>
<!-- Parameters for the tabs control -->
<xss:choice minOccurs="1"
    maxOccurs="unbounded">
<xss:element name="DefaultTabId"
    type="xs:string"
    minOccurs="0"
    maxOccurs="1" />
<xss:element name="ShowArticleTab"
    type="xs:boolean"
    minOccurs="0"
    maxOccurs="1" />
<xss:group ref="SearchWidgetControlParameters"
    minOccurs="0"
    maxOccurs="unbounded" />
</xss:choice>
<!-- Link Control parameters -->
<xss:choice minOccurs="1"
    maxOccurs="unbounded">
<xss:element name="LinkControlDefinitionId"
    type="FormGuidType"
    minOccurs="0"
    maxOccurs="1" />
<xss:element name="ShowLinkControlLabel"
    type="xs:boolean"
    minOccurs="0"
    maxOccurs="1" />
</xss:choice>
<!-- Bing Maps Control parameters -->
<xss:choice minOccurs="1"
    maxOccurs="unbounded">
<xss:element name="AddressField"
    type="xs:string"
    minOccurs="1"
    maxOccurs="1" />
</xss:choice>
<!-- Timer Control parameters -->
<xss:choice minOccurs="1"
    maxOccurs="unbounded">
<xss:element name="FailureTimeField"
    type="xs:string"
    minOccurs="1"
    maxOccurs="1" />
<xss:element name="SuccessConditionName"
    type="xs:string"
    minOccurs="1"
    maxOccurs="1" />
</xss:choice>

```

```

                maxOccurs="1" />
<xss:element name="SuccessConditionValue"
              type="xs:string"
              minOccurs="1"
              maxOccurs="1" />
<xss:element name="FailureConditionName"
              type="xs:string"
              minOccurs="0"
              maxOccurs="1" />
<xss:element name="FailureConditionValue"
              type="xs:string"
              minOccurs="0"
              maxOccurs="1" />
<xss:element name="WarningConditionName"
              type="xs:string"
              minOccurs="0"
              maxOccurs="1" />
<xss:element name="WarningConditionValue"
              type="xs:string"
              minOccurs="0"
              maxOccurs="1" />
<xss:element name="CancelConditionName"
              type="xs:string"
              minOccurs="0"
              maxOccurs="1" />
<xss:element name="CancelConditionValue"
              type="xs:string"
              minOccurs="0"
              maxOccurs="1" />
<xss:element name="PauseConditionName"
              type="xs:string"
              minOccurs="0"
              maxOccurs="1" />
<xss:element name="PauseConditionValue"
              type="xs:string"
              minOccurs="0"
              maxOccurs="1" />
</xss:choice>
<!-- Search Widget parameters -->
<xss:choice minOccurs="1"
           maxOccurs="unbounded">
<xss:group ref="SearchWidgetControlParameters"
           minOccurs="0"
           maxOccurs="unbounded" />
</xss:choice>
<!-- Queue Control parameters -->
<xss:choice minOccurs="1"
           maxOccurs="unbounded">
<xss:element name="StreamObjects"
              minOccurs="1"
              maxOccurs="1" >
<xss:complexType>
<xss:sequence>
<xss:element name="ShowAsTiles"
              type="xs:boolean"

```

```

        minOccurs="1"
        maxOccurs="1" />
<xs:element name="StreamObject"
            minOccurs="1"
            maxOccurs="unbounded" >
<xs:complexType>
    <xs:sequence>
        <xs:element name="LogicalEntityName"
                    type="xs:string"
                    minOccurs="1"
                    maxOccurs="1" />
        <xs:choice minOccurs="1"
                    maxOccurs="1">
            <!-- Parameters for stream objects of type queue
-->
            <xs:choice minOccurs="1"
                        maxOccurs="unbounded">
                <xs:element name="QueueId"
                            type="FormGuidType"
                            minOccurs="1"
                            maxOccurs="1" />
                <xs:element name="QueueViewId"
                            type="FormGuidType"
                            minOccurs="1"
                            maxOccurs="1" />
            </xs:choice>
            <!-- Parameters for stream objects of type
entity view -->
            <xs:choice minOccurs="1"
                        maxOccurs="unbounded">
                <xs:element name="EntityViewId"
                            type="FormGuidType"
                            minOccurs="1"
                            maxOccurs="1" />
            </xs:choice>
            <!-- Parameters for stream objects of type saved
query on queue -->
            <xs:choice minOccurs="1"
                        maxOccurs="unbounded">
                <xs:element name="SavedQueryID"
                            type="FormGuidType"
                            minOccurs="1"
                            maxOccurs="1" />
                <xs:element name="QueueViewIdForSavedQuery"
                            type="FormGuidType"
                            minOccurs="1"
                            maxOccurs="1" />
            </xs:choice>
            </xs:choice>
        </xs:sequence>
        <xs:attribute name="type"
                      type="xs:nonNegativeInteger"
                      use="required" />
        <xs:attribute name="id"
                      type="FormGuidType"

```

```
        use="required" />

            </xs:complexType>
            </xs:element>
        </xs:sequence>

            </xs:complexType>
            </xs:element>
        </xs:choice>
        <!-- Date Range Control parameters -->
        <xs:choice minOccurs="1"
                    maxOccurs="unbounded">
            <xs:element name="AttributeLogicalName"
                        type="xs:string"
                        minOccurs="0"
                        maxOccurs="1" />
            <xs:element name="TimeFrame"
                        type="xs:string"
                        minOccurs="0"
                        maxOccurs="1" />
        </xs:choice>
    </xs:choice>

        </xs:complexType>
    </xs:element>
</xs:sequence>
<xs:attribute name="id"
                type="xs:string" />
<xs:attribute name="uniqueid"
                type="FormGuidType" />
<xs:attribute name="classid"
                type="FormGuidType" />
<xs:attribute name="labelid"
                type="FormGuidType" />
<xs:attribute name="datafieldname"
                type="xs:string" />
<xs:attribute name="disabled"
                type="xs:boolean" />
<xs:attribute name="addedby"
                type="xs:string" />
<xs:attribute name="isunbound"
                type="xs:boolean" />
<xs:attribute name="isrequired"
                type="xs:boolean" />
<xs:attribute name="relationship"
                type="xs:string" />
<xs:attribute name="indicationOfSubgrid"
                type="xs:boolean" />

</xs:complexType>
<xs:complexType name="FormXmlLibraryType">
    <xs:sequence>
        <xs:element name="Library"
                    minOccurs="1"
                    maxOccurs="50">
```

```
<xs:complexType>
  <xs:attribute name="name"
                type="xs:string"
                use="required" />
  <xs:attribute name="libraryUniqueId"
                type="xs:string"
                use="required" />

</xs:complexType>
</xs:element>
</xs:sequence>

</xs:complexType>
<xs:simpleType name="CrmEventType">
  <xs:restriction base="xs:string">
    <xs:enumeration value="DataEvent" />
    <xs:enumeration value="ControlEvent" />
  </xs:restriction>
</xs:simpleType>
<xs:complexType name="FormXmlHandlerType">
  <xs:sequence>
    <xs:element name="dependencies"
               minOccurs="0"
               maxOccurs="1">
      <xs:complexType>
        <xs:sequence>
          <xs:element name="dependency"
                     minOccurs="0"
                     maxOccurs="unbounded">
            <xs:complexType>
              <xs:attribute name="id"
                            type="xs:string" />

            </xs:complexType>
          </xs:element>
        </xs:sequence>
      </xs:complexType>
    </xs:element>
  </xs:sequence>

  </xs:complexType>
</xs:element>
</xs:sequence>

</xs:complexType>
</xs:element>
</xs:sequence>
<xs:attribute name="functionName"
                type="xs:string"
                use="required" />
<xs:attribute name="libraryName"
                type="xs:string"
                use="required" />
<xs:attribute name="handlerUniqueId"
                type="xs:string"
                use="required" />
<xs:attribute name="enabled"
                type="xs:boolean" />
<xs:attribute name="passExecutionContext"
                type="xs:boolean" />
<xs:attribute name="parameters"
                type="xs:string" />
```

```
</xs:complexType>
<xs:complexType name="FormXmlEventsType">
  <xs:sequence>
    <xs:element name="event"
      minOccurs="1"
      maxOccurs="unbounded">
      <xs:complexType>
        <xs:all>
          <xs:element name="Handlers"
            minOccurs="0"
            maxOccurs="1">
            <xs:complexType>
              <xs:sequence>
                <xs:element name="Handler"
                  type="FormXmlHandlerType"
                  minOccurs="0"
                  maxOccurs="50" />
              </xs:sequence>

            </xs:complexType>
          </xs:element>
          <xs:element name="InternalHandlers"
            minOccurs="0"
            maxOccurs="1">
            <xs:complexType>
              <xs:sequence>
                <xs:element name="Handler"
                  type="FormXmlHandlerType"
                  minOccurs="0"
                  maxOccurs="50" />
              </xs:sequence>

            </xs:complexType>
          </xs:element>
          <xs:element name="dependencies"
            minOccurs="0"
            maxOccurs="1">
            <xs:complexType>
              <xs:sequence>
                <xs:element name="dependency"
                  minOccurs="0"
                  maxOccurs="unbounded">
                  <xs:complexType>
                    <xs:attribute name="id"
                      type="xs:string" />

                  </xs:complexType>
                </xs:element>
              </xs:sequence>

            </xs:complexType>
          </xs:element>
        </xs:all>
        <xs:attribute name="name"
```

```

                type="xs:string" />
        <xs:attribute name="BehaviorInBulkEditForm"
                      type="BehaviorInBulkEditForm" />
        <xs:attribute name="application"
                      type="xs:boolean" />
        <xs:attribute name="active"
                      type="xs:boolean" />
        <xs:attribute name="eventType"
                      type="CrmEventType" />
        <xs:attribute name="attribute"
                      type="xs:string" />

    </xs:complexType>
</xs:element>
</xs:sequence>

</xs:complexType>
<xs:complexType name="FormXmlExternalDependenciesType">
    <xs:sequence>
        <xs:element name="dependency"
                    minOccurs="1"
                    maxOccurs="unbounded">
            <xs:complexType>
                <xs:attribute name="id"
                              type="xs:string" />

            </xs:complexType>
        </xs:element>
    </xs:sequence>

</xs:complexType>
<xs:complexType name="FormXmlLabelsType">
    <xs:sequence>
        <xs:element name="label"
                    minOccurs="0"
                    maxOccurs="unbounded">
            <xs:complexType>
                <xs:attribute name="description"
                              use="required"
                              type="xs:string" />
                <xs:attribute name="languagecode"
                              use="required"
                              type="xs:positiveInteger" />
                <xs:attribute name="addedby"
                              type="xs:string" />

            </xs:complexType>
        </xs:element>
    </xs:sequence>

</xs:complexType>
<xs:complexType name="FormXmlHeaderFooterType">
    <xs:sequence>
        <xs:element name="rows"
                    minOccurs="1"

```

```

        maxOccurs="1">
    <xss:complexType>
        <xss:sequence>
            <xss:element name="row"
                minOccurs="0"
                maxOccurs="unbounded">
                <xss:complexType>
                    <xss:sequence>
                        <xss:element name="cell"
                            minOccurs="0"
                            maxOccurs="unbounded">
                            <xss:complexType>
                                <xss:all>
                                    <xss:element name="labels"
                                        type="FormXmlLabelsType"
                                        minOccurs="0"
                                        maxOccurs="1" />
                                    <xss:element name="control"
                                        type="FormXmlControlType"
                                        minOccurs="0"
                                        maxOccurs="1" />
                                </xss:all>
                                <xss:attributeGroup ref="FormXmlCellCommon" />
                            </xss:complexType>
                        </xss:element>
                    </xss:sequence>
                    <xss:attributeGroup ref="FormXmlRowCommon" />
                </xss:complexType>
            </xss:element>
        </xss:sequence>
    </xss:complexType>
</xss:element>
</xss:sequence>
<xss:attribute name="id"
    type="FormGuidType"
    use="required" />
<xss:attributeGroup ref="FormXmlSectionCommon" />
</xss:complexType>
<xss:attributeGroup name="FormXmlSectionCommon">
    <xss:attribute name="columns"
        type="xs:nonNegativeInteger" />
    <xss:attribute name="labelwidth"
        type="xs:nonNegativeInteger" />
    <xss:attribute name="celllabelalignment"
        use="optional">
        <xss:simpleType>
            <xss:restriction base="xs:string">
                <xss:enumeration value="Center"/>
                <xss:enumeration value="Left"/>
                <xss:enumeration value="Right"/>
            </xss:restriction>
        </xss:simpleType>
    </xss:attribute>
    <xss:attribute name="celllabelposition"

```

```
        use="optional">
    <xs:simpleType>
        <xs:restriction base="xs:string">
            <xs:enumeration value="Top"/>
            <xs:enumeration value="Left"/>
        </xs:restriction>
    </xs:simpleType>
</xs:attribute>

</xs:attributeGroup>
<xs:attributeGroup name="FormXmlRowCommon">
    <xs:attribute name="height"
        type="xs:string" />

</xs:attributeGroup>
<xs:attributeGroup name="FormXmlCellCommon">
    <xs:attribute name="id"
        type="FormGuidType" />
    <xs:attribute name="showlabel"
        type="xs:boolean" />
    <xs:attribute name="labelid"
        type="FormGuidType"
        use="optional" />
    <xs:attribute name="locklevel"
        type="xs:nonNegativeInteger" />
    <xs:attribute name="rowspan"
        type="xs:nonNegativeInteger" />
    <xs:attribute name="colspan"
        type="xs:nonNegativeInteger" />
    <xs:attribute name="userspacer"
        type="xs:boolean" />
    <xs:attribute name="ispreviewcell"
        type="xs:boolean"/>
    <xs:attribute name="visible"
        type="xs:boolean" />
    <xs:attribute name="availableforphone"
        type="xs:boolean" />
    <xs:attribute name="isstreamcell"
        type="xs:boolean" />
    <xs:attribute name="ischartcell"
        type="xs:boolean" />
    <xs:attribute name="istilecell"
        type="xs:boolean" />

</xs:attributeGroup>
<xs:simpleType name="BehaviorInBulkEditForm">
    <xs:restriction base="xs:string">
        <xs:enumeration value="Disabled" />
        <xs:enumeration value="EnabledButNoRender" />
        <xs:enumeration value="Enabled" />
    </xs:restriction>
</xs:simpleType>
<xs:simpleType name="FormCRM_Boolean">
    <xs:restriction base="xs:unsignedByte">
        <xs:minInclusive value="0" />
```

```

        <xs:maxInclusive value="1" />
    </xs:restriction>
</xs:simpleType>
<xs:simpleType name="FormGuidType">
    <xs:annotation>
        <xs:documentation xml:lang="en">
            The representation of a GUID, generally the id of an element.
        </xs:documentation>
    </xs:annotation>
    <xs:restriction base="xs:string">
        <xs:pattern value="\{[a-fA-F0-9]{8}-[a-fA-F0-9]{4}-[a-fA-F0-9]{4}-[a-fA-F0-9]{4}\}" />
    </xs:restriction>
</xs:simpleType>
<xs:simpleType name="FormISVGuid">
    <xs:restriction base="xs:string">
        <xs:pattern value="\{?[0-9a-fA-F]{8}-[0-9a-fA-F]{4}-[0-9a-fA-F]{4}-[0-9a-fA-F]{4}\}?" />
    </xs:restriction>
</xs:simpleType>
<xs:simpleType name="FormParameterAttributeType">
    <xs:restriction base="xs:string">
        <xs:enumeration value="Boolean" />
        <xs:enumeration value="DateTime" />
        <xs:enumeration value="Double" />
        <xs:enumeration value="EntityType" />
        <xs:enumeration value="Integer" />
        <xs:enumeration value="Long" />
        <xs:enumeration value="PositiveInteger" />
        <xs:enumeration value="SafeString" />
        <xs:enumeration value="UniqueId" />
        <xs:enumeration value="UnsignedInt" />
    </xs:restriction>
</xs:simpleType>
<xs:simpleType name="FormParameterPassAsAttributeType">
    <xs:restriction base="xs:string">
        <xs:enumeration value="QueryString" />
    </xs:restriction>
</xs:simpleType>
<xs:simpleType name="FormPercentageType">
    <xs:restriction base="xs:string">
        <xs:pattern value="^(100|[0-9]{1,2})%$" />
    </xs:restriction>
</xs:simpleType>
<xs:simpleType name="FormQueryStringParameterNameAttributeType">
    <xs:restriction base="xs:string">
        <xs:pattern value="(?![cC][rR][mM]_)([A-Za-z0-9])+([_])+([A-Za-z0-9_])*/>
    </xs:restriction>
</xs:simpleType>
<xs:simpleType name="GridResizeType">
    <xs:restriction base="xs:string">
        <xs:enumeration value="Auto"/>
        <xs:enumeration value="Fixed"/>
        <xs:enumeration value="AutoWithFixedMax"/>
    </xs:restriction>
</xs:simpleType>

```

```
</xs:restriction>
</xs:simpleType>
<xs:simpleType name="ImageHorizontalAlignmentType">
  <xs:restriction base ="xs:string">
    <xs:enumeration value ="Left" />
    <xs:enumeration value ="Right" />
    <xs:enumeration value ="Center" />
    <xs:enumeration value ="NotSet" />
  </xs:restriction>
</xs:simpleType>
<xs:simpleType name="ImageVerticalAlignmentType">
  <xs:restriction base ="xs:string">
    <xs:enumeration value ="Top" />
    <xs:enumeration value ="Middle" />
    <xs:enumeration value ="Bottom" />
    <xs:enumeration value ="NotSet" />
  </xs:restriction>
</xs:simpleType>
<xs:simpleType name="RelationshipRoleOrdinalType">
  <xs:restriction base="xs:unsignedByte">
    <xs:enumeration value="1" />
    <xs:enumeration value="2" />
  </xs:restriction>
</xs:simpleType>
<xs:simpleType name = "WebResourceSizeType">
  <xs:restriction base ="xs:string">
    <xs:enumeration value = "StretchToFit" />
    <xs:enumeration value = "StretchMaintainAspectRatio" />
    <xs:enumeration value = "Original" />
    <xs:enumeration value = "Specific" />
  </xs:restriction>
</xs:simpleType>
<xs:simpleType name="FormatType">
  <xs:restriction base ="xs:string">
    <xs:enumeration value = "SingleLineOfText" />
    <xs:enumeration value = "WholeNumber" />
    <xs:enumeration value = "DecimalNumber" />
    <xs:enumeration value = "Currency" />
    <xs:enumeration value="Date" />
    <xs:enumeration value="DateTime" />
    <xs:enumeration value="DateAndTime" />
    <xs:enumeration value="Url" />
    <xs:enumeration value="Ticker" />
    <xs:enumeration value="Email" />
    <xs:enumeration value="TextArea" />
  </xs:restriction>
</xs:simpleType>
<xs:simpleType name="solutionactionType">
  <xs:restriction base="xs:string">
    <xs:enumeration value="Added" />
    <xs:enumeration value="Removed" />
    <xs:enumeration value="Modified" />
  </xs:restriction>
</xs:simpleType >
<xs:attributeGroup name="FormXmlBaseElementCommon">
```

```
<xs:attribute name="solutionaction"
    type="solutionactionType" />
</xs:attributeGroup>
<xs:complexType name ="solutionStringType">
    <xs:simpleContent>
        <xs:extension base="xs:string">
            <xs:attributeGroup ref="FormXmlBaseElementCommon"/>
        </xs:extension>
    </xs:simpleContent>
</xs:complexType>
<xs:complexType name ="solutionBooleanType">
    <xs:simpleContent>
        <xs:extension base="xs:boolean">
            <xs:attributeGroup ref="FormXmlBaseElementCommon"/>
        </xs:extension>
    </xs:simpleContent>
</xs:complexType>
<xs:complexType name ="solutionFormGuidType">
    <xs:simpleContent>
        <xs:extension base="FormGuidType">
            <xs:attributeGroup ref="FormXmlBaseElementCommon"/>
        </xs:extension>
    </xs:simpleContent>
</xs:complexType>
<xs:complexType name ="solutionUnsignedIntType">
    <xs:simpleContent>
        <xs:extension base="xs:unsignedInt">
            <xs:attributeGroup ref="FormXmlBaseElementCommon"/>
        </xs:extension>
    </xs:simpleContent>
</xs:complexType>
<xs:complexType name ="solutionWebResourceSizeType">
    <xs:simpleContent>
        <xs:extension base="WebResourceSizeType">
            <xs:attributeGroup ref="FormXmlBaseElementCommon"/>
        </xs:extension>
    </xs:simpleContent>
</xs:complexType>
<xs:complexType name ="solutionImageHorizontalAlignmentType">
    <xs:simpleContent>
        <xs:extension base="ImageHorizontalAlignmentType">
            <xs:attributeGroup ref="FormXmlBaseElementCommon"/>
        </xs:extension>
    </xs:simpleContent>
</xs:complexType>
<xs:complexType name ="solutionFormatType">
    <xs:simpleContent>
        <xs:extension base="FormatType">
            <xs:attributeGroup ref="FormXmlBaseElementCommon"/>
        </xs:extension>
    </xs:simpleContent>
</xs:complexType>
<xs:complexType name ="solutionImageVerticalAlignmentType">
    <xs:simpleContent>
        <xs:extension base="ImageVerticalAlignmentType">
```

```
    <xs:attributeGroup ref="FormXmlBaseElementCommon"/>
  </xs:extension>
</xs:simpleContent>
</xs:complexType>
<xs:complexType name ="solutionRelationshipRoleOrdinalType">
  <xs:simpleContent>
    <xs:extension base="RelationshipRoleOrdinalType">
      <xs:attributeGroup ref="FormXmlBaseElementCommon"/>
    </xs:extension>
  </xs:simpleContent>
</xs:complexType>
<xs:complexType name ="solutionGridResizeType">
  <xs:simpleContent>
    <xs:extension base="GridResizeType">
      <xs:attributeGroup ref="FormXmlBaseElementCommon"/>
    </xs:extension>
  </xs:simpleContent>
</xs:complexType>
<xs:complexType name ="solutionUnsignedShortType">
  <xs:simpleContent>
    <xs:extension base="xs:unsignedShort">
      <xs:attributeGroup ref="FormXmlBaseElementCommon"/>
    </xs:extension>
  </xs:simpleContent>
</xs:complexType>
<xs:group name="SearchWidgetControlParameters">
  <xs:choice>
    <xs:element name="FilterResults"
      type="xs:string"
      minOccurs="0"
      maxOccurs="1" />
    <xs:element name="AllowChangingFiltersOnUI"
      type="xs:boolean"
      minOccurs="0"
      maxOccurs="1" />
    <xs:element name="ShowLanguageFilter"
      type="xs:boolean"
      minOccurs="0"
      maxOccurs="1" />
    <xs:element name="ShowDepartmentFilter"
      type="xs:boolean"
      minOccurs="0"
      maxOccurs="1" />
    <xs:element name="EnableAutoSuggestions"
      type="xs:boolean"
      minOccurs="0"
      maxOccurs="1" />
    <xs:element name="SearchForAutoSuggestionsUsing"
      type="xs:string"
      minOccurs="0"
      maxOccurs="1" />
    <xs:element name="EnableRating"
      type="xs:boolean"
      minOccurs="0"
      maxOccurs="1" />
```

```
<xs:element name="ShowRatingUsing"
            type="xs:string"
            minOccurs="0"
            maxOccurs="1" />
<xs:element name="AutoSuggestionSource"
            type="xs:string"
            minOccurs="0"
            maxOccurs="1" />
<xs:element name="SelectPrimaryCustomer"
            type="xs:string"
            minOccurs="0"
            maxOccurs="1" />
<xs:element name="NumberOfResults"
            type="xs:unsignedInt"
            minOccurs="0"
            maxOccurs="1" />
<xs:element name="ShowContextualActions"
            type="xs:string"
            minOccurs="0"
            maxOccurs="1" />
<xs:element name="ActionList"
            type="xs:string"
            minOccurs="0"
            maxOccurs="1" />
<xs:element name="ReferencePanelSearchWidgetIconUrl"
            type="xs:string"
            minOccurs="0"
            maxOccurs="1" />
<xs:element name="SelectDefaultLanguage"
            type="xs:string"
            minOccurs="0"
            maxOccurs="1" />
        </xs:choice>
    </xs:group>
</xs:schema>
```

See also

[Customize forms](#)

[Create, install, and update a managed solution](#)

[Create, export, or import an Unmanaged solution](#)

[Form XML schema](#)

Publish request schema

Article • 11/30/2022

The following is the schema definition for the [PublishXmlRequest](#) message. For more information, see [Publish customizations](#). [Download the schemas](#).

Schema

```
XML

<?xml version="1.0" encoding="utf-8"?>
<xs:schema attributeFormDefault="unqualified"
            elementFormDefault="qualified"
            xmlns:xs="https://www.w3.org/2001/XMLSchema">
<xs:element name="importexportxml">
    <xs:complexType>
        <xs:sequence>
            <xs:element name="entities"
                        minOccurs="0">
                <xs:complexType>
                    <xs:sequence minOccurs="0"
                                maxOccurs="unbounded">
                        <!-- Name of the entity to publish-->
                        <xs:element maxOccurs="unbounded"
                                    name="entity"
                                    type="xs:string" />
                    </xs:sequence>
                </xs:complexType>
            </xs:element>
            <xs:element name="ribbons"
                        minOccurs="0">
                <xs:complexType>
                    <xs:sequence minOccurs="0"
                                maxOccurs="unbounded">
                        <!-- Application ribbon. : This value is not used. We publish the
                            application ribbon if there is at least one <ribbon> node present -->
                        <xs:element maxOccurs="unbounded"
                                    name="ribbon"
                                    type="xs:string" />
                    </xs:sequence>
                </xs:complexType>
            </xs:element>
            <xs:element name="dashboards"
                        minOccurs="0">
                <xs:complexType>
                    <xs:sequence minOccurs="0"
                                maxOccurs="unbounded">
                        <!-- Guid of the systemform to publish-->
                        <xs:element maxOccurs="unbounded"
                                    name="dashboard" />
                    </xs:sequence>
                </xs:complexType>
            </xs:element>
        </xs:sequence>
    </xs:complexType>
</xs:element>
```

```

        type="xs:string" />
    </xs:sequence>
</xs:complexType>
</xs:element>
<xs:element name="optionsets"
            minOccurs="0">
    <xs:complexType>
        <xs:sequence minOccurs="0"
                     maxOccurs="unbounded">
            <!-- Name of the optionset to publish-->
            <xs:element maxOccurs="unbounded"
                        name="optionset"
                        type="xs:string" />
        </xs:sequence>
    </xs:complexType>
</xs:element>
<xs:element name="sitemaps"
            minOccurs="0">
    <xs:complexType>
        <xs:sequence minOccurs="0"
                     maxOccurs="unbounded">
            <!-- Guid of the sitemap to publish : This value is not used. We
publish the sitemap if there is at least one <sitemap> node present-->
            <xs:element maxOccurs="unbounded"
                        name="sitemap"
                        type="xs:string" />
        </xs:sequence>
    </xs:complexType>
</xs:element>
<xs:element name="webresources"
            minOccurs="0">
    <xs:complexType>
        <xs:sequence minOccurs="0"
                     maxOccurs="unbounded">
            <!-- Guid of the web resource to publish -->
            <xs:element maxOccurs="unbounded"
                        name="webresource"
                        type="xs:string" />
        </xs:sequence>
    </xs:complexType>
</xs:element>
</xs:sequence>
</xs:complexType>
</xs:element>
</xs:schema>

```

See also

[PublishXmlRequest](#)

[Publish customizations](#)

Visualization data description schema

Article • 12/16/2022

The following is the schema for the data description XML string for charts in visualization. This can be used to validate the contents of the data description XML string while creating a chart. For more information, see [Understanding Charts: Underlying Data and Chart Representation](#). Download the schemas ↗ and see the file `VisualizationDataDescription.xsd` in the folder.

Schema

```
XML

<?xml version='1.0' encoding='utf-8'?>
<xsschema attributeFormDefault='unqualified'
           elementFormDefault='qualified'
           xmlns:xss='https://www.w3.org/2001/XMLSchema'>
<xselement name='datadefinition'>
  <xsccomplexType>
    <xsssequence>
      <xselement name='fetchcollection'>
        <xsccomplexType>
          <xsssequence>
            <xselement maxOccurs='unbounded'
                       name='fetch'>
              <xssannotation>
                <!--FetchXML goes here-->
              </xssannotation>
            </xselement>
          </xsssequence>
        </xsccomplexType>
      </xselement>
      <xselement name='categorycollection'>
        <xsccomplexType>
          <xsssequence>
            <xselement name='category'
                       type='CategoryType'
                       minOccurs='1'
                       maxOccurs='unbounded' />
          </xsssequence>
        </xsccomplexType>
      </xselement>
      <xsssequence>
    </xsccomplexType>
  </xselement>
<xsccomplexType name = 'CategoryType'>
  <xsssequence>
    <xselement minOccurs='1'>
```

```
        maxOccurs='unbounded'
        name='measurecollection'>
<xs:complexType>
<xs:sequence>
<xs:element minOccurs='1'
            maxOccurs='unbounded'
            name='measure'>
<xs:complexType>
<xs:attribute name='alias'
              type='xs:string'
              use='required' />
</xs:complexType>
</xs:element>
</xs:sequence>
</xs:complexType>
</xs:element>
</xs:sequence>
<xs:attribute name='alias'
              type='xs:string'
              use='optional' />
</xs:complexType>
</xs:schema>
```

See also

[Customize Visualizations and Dashboards](#)

[Understanding Charts: Underlying Data and Chart Representation](#)

[Sample Charts](#)

[Use FetchXML to construct a query](#)