User interface development home page

Article • 12/31/2024

This article contains links to topics about developing user interface elements.

The user interface for Finance and Operation applications differs significantly from the interface for Microsoft Dynamics AX 2012. The client in Dynamics AX 2012 is a Microsoft Win32 application that has extensions that use ActiveX, WinForm, or WPF controls. The X++ application logic runs on the client for the form and table methods, and some logic occurs on the server. For controls, both the X++ logic application programming interface (API) and the physical Win32 control are tightly connected on the client. The client is an HTML web client that runs in all major browsers. These browsers include Microsoft Edge, Internet Explorer 11, Chrome, and Safari (see System requirements). The move to a web client has produced the following changes to client forms and controls:

- The physical presentation of forms and controls is now HTML, JavaScript, and CSS within the browser.
- Form controls are split into logical and physical parts. The X++ logical API and related state run on the server.
- The logical and physical parts are kept in sync through service calls that communicate changes from each side. For example, a user action on the client creates a service call to the server that is either sent immediately or queued so that it can be sent later.
- The server tier keeps the form state in memory while the form is open.

The form metamodel continues to be used to define controls and application logic. This approach supports almost all the existing Form, Form DataSource, and Form Control metamodel and X++ override methods. However, some control types, properties, and override methods have been removed, either because of incompatibility with the new platform or for performance reasons. For example, ActiveX and ManagedHost controls can no longer be used to add custom controls, because they are incompatible with the HTML platform. Instead, a new extensible control framework has been added that lets you add additional controls.

Tutorials

- Build the Rental Charge Type form
- Build the customer form

Forms

Build the Rental Charge Type form

Article • 01/23/2025

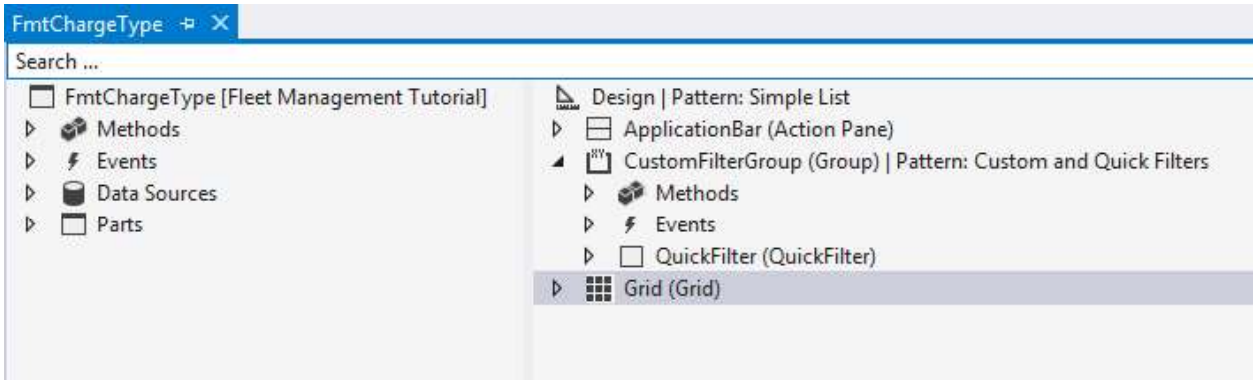
In this lab you'll create a Simple List form. A Simple List form can show reference or secondary data that has six or fewer fields. For example, the form that you create will list and describe the types of rental charges.

Prerequisites

For this tutorial, you'll need to access the environment using Remote Desktop, and be provisioned as an administrator on the instance. For more information, see Access Instances.

Overview

To create the form, you'll start from the existing form, **FmtChargeType**. This form uses the Simple List pattern. The following illustration shows the **FmtChargeType** form with the required controls from the Simple List pattern.



Adhering to the form pattern ensures that this Simple List form has the same structure and layout as other Simple List forms.

Key concepts

- Create a Simple List form using a pattern.
- Bind a table to the form.
- Add controls to the form.
- View the form using Visual Studio and a browser.

Setup

Build the Customer form

Article • 01/23/2025

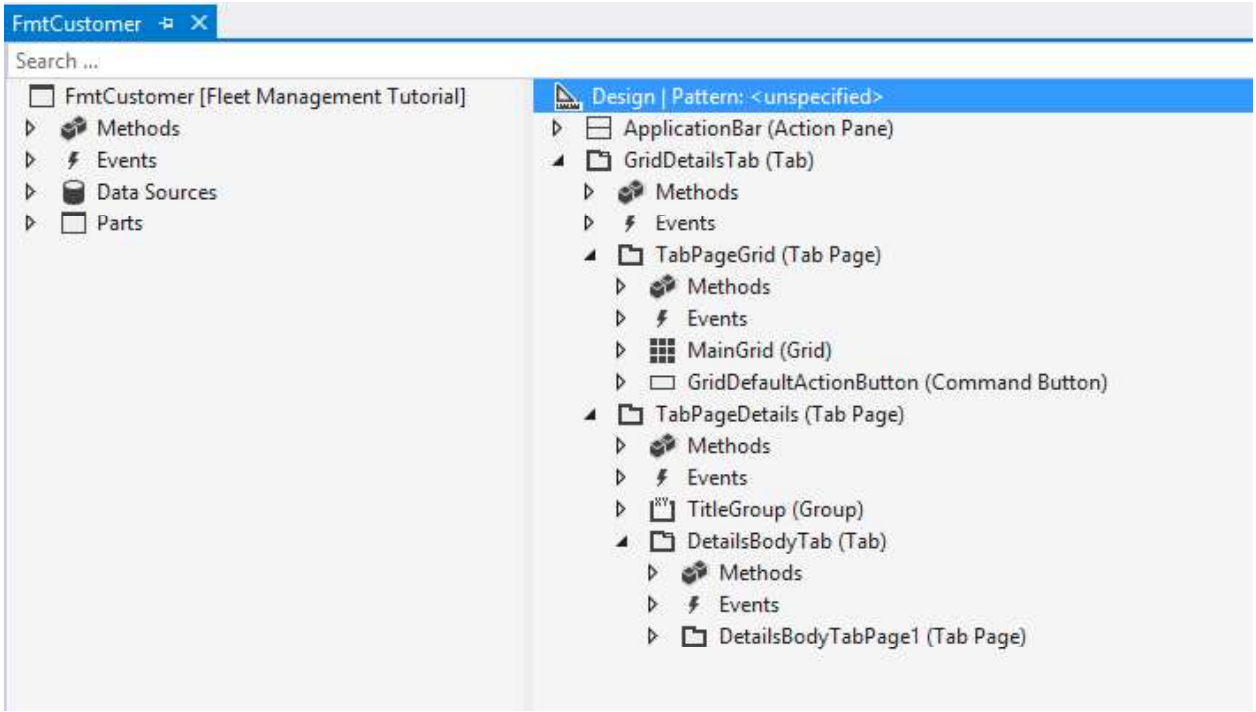
In this lab you'll create a Master Details form and apply the appropriate form pattern and subpatterns. A Master Details form shows primary data that has many fields. For example, the form that you create will show customer information.

Prerequisites

For this tutorial, you will need to access the environment using Remote Desktop, and be provisioned as an administrator on the instance. For more information, see Access Instances.

Overview

To create the form, you'll start from the existing form, **FmtCustomer**. The form represents the old Master Details template. As a part of the tutorial, you'll apply the Master Details pattern, which will enforce a consistent structure for this form type. The following illustration shows the **FmtCustomer** starting artifact.



Key concepts

- Create a Master Details form.
- Apply a form pattern to a form.

Build navigation

Article • 01/23/2025

In this tutorial, you will add navigational elements to a workspace and the navigation pane.

Prerequisites

For this tutorial, you need to access the environment using Remote Desktop, and be provisioned as an administrator on the instance. For more information, see Access Instances.

Key concepts

- A *workspace* is an overview page that is specific to a particular subject area. Workspaces are common to all users. In this tutorial, you will add content into an existing workspace.
- The *dashboard* is the default home page for each user.
- *Tiles* are securable objects that can be shown on a workspace or the dashboard. They can be secured by using menu items.

Setup

If this is the first tutorial that you are working on, review Access Instances and make sure that you provision your administrator user if you are working on a local VM.

Import the tutorial project

If you have already imported the Fleet management tutorial project, skip to the next section.

1. Download the Fleet Management sample from https://github.com/Microsoft/FMLab, save it to **C:**, and unzip it.
2. In Visual Studio, on the **Finance and operations** menu, click **Import Project**.
3. In the **Import Project** window, next to the **Filename** text box, click the ellipsis button.
4. In the **Select the file to import** window, browse to **C:\FMLab**, click **FMTutorialDataModel.axpp,** and then click **Open**.
5. In the Project file location text box, enter **C:\FMLab.**

Modify a workspace with a tile, list, and data cache

Article • 01/03/2025

In this tutorial, you will create a new tile and include it in the summary section of a workspace, build a new list for a workspace, and create a data cache for the list in the workspace.

Prerequisites

For this tutorial, you must access the environment by using Remote Desktop, and you must be provisioned as an administrator on the instance. For more information, see Deploy and access development environments.

Key concepts

- Learn about and use form patterns that are related to workspaces.
- Create a new tile, and include it in the **Summary** section of a workspace.
- Build a new list for a workspace.
- Create a data cache for the list in the workspace.

Setup

Import the tutorial project and transactional data

Use Microsoft Visual Studio to import the tutorial project. The tutorial project includes the artifacts that you will use to complete this tutorial. Use Visual Studio to open the FMTutorial project and load the data for the tutorial. You will use the **FMTDataHelper** class to load data for the Fleet Management tutorial. If this is the first tutorial that you're working on, review Deploy and access development environments, and make sure that you provision your administrator user if you're working on a local virtual machine (VM).

1. Download the **FMTutorialDataModel.axpp** file from the Microsoft Dynamics Lifecycle Services (LCS) methodology, and copy it to the **Downloads** folder of the VM.
2. On the desktop, double-click the Visual Studio shortcut to open the development environment.
3. On the **Dynamics 365** menu, click **Import Project**.

Navigation concepts

Article • 01/03/2025

This articles describes the primary navigation concepts including the dashboard, the new navigation search feature, the navigation pane, workspaces, and tiles.

Navigation concepts

The primary navigation concepts are:

- Dashboard
- Navigation pane
- Workspaces
- Tiles
- Navigation search

The dashboard is a new concept, whereas the navigation pane and workspaces are updates to existing concepts. To implement the navigation concepts, the user interface model uses several standard page types. When you create an application, you should follow the conventions for these pages to present a consistent experience for the user. The following sections provide more detail about the pages that underlie these concepts, and include information about the modeling of these and other types of pages.

Dashboard

The dashboard is the first page that users see when they access the client. The dashboard contains tiles that show important details from the system. Content that was previously displayed in Cues on Role Center pages in Microsoft Dynamics AX 2012 is now available on the dashboard. You can return to the dashboard at any time by clicking **Dynamics 365** on the navigation bar at the top of the application frame.

The dashboard primarily consists of a large section of workspace tiles. There might also be a Getting Started tool, which isn't shown in the preceding screenshot. The dashboard's workspace tile section is built from a menu structure that has its root in the **NavPaneMenu** menu. The menu is modified by a set of menu extensions, and those extensions contain one or more tile references that correspond to the tiles that users see in that section.

Page layout in the web client

Article • 12/31/2024

This article discusses layout in the web client. Layout is a design process that specifies how controls appear on a page.

Introduction

Layout is a design process that specifies how the controls on a page appear in the web client. Layout occurs within container controls. The following table lists the container controls.

Expand table

Container	Description
Form.Design	The root of the page. It functions as a special kind of container.
Group	The general-purpose container control. Group controls can be nested as required.
Tab	A control that contains TabPage controls and has many possible **Tab.Style** values, such as **Tab**, **FastTab**, **Vertical Tab**, and **Panorama**.
TabPage	The appearance of each TabPage control depends on its **Tab.Style** value.
ButtonGroup	A special type of Group control that contains buttons.

A grid is a special type of control that has some container behaviors, such as flexible sizing (**SizeToAvailable**). However, a grid has special visualizations and isn't a general-purpose container control.

Layout: Dynamics AX 2012 vs. Finance and operations apps

Layout in Dynamics AX 2012

In Microsoft Dynamics AX 2012, the arrangement of controls in containers is almost always vertical, and columns are manually set to provide some horizontal spread.

Examples

Dynamics Symbol font

Article • 03/19/2024

The Dynamics Symbol font defines the set of out-of-box symbols that are available in the product. These symbols are primarily used for buttons, tiles, and image controls.

To access the list of available symbols (the name and an image), visit the Dynamics Symbol Font page. A description of the various locations where symbols are used in the product and usage guidelines for each location is also included.

Feedback

Saved views

Article • 12/28/2024

Personalization plays an important role in allowing users and organizations to optimize the user experience to meet their needs. For more information on personalization, see Personalize the user experience.

Traditional personalization allows users to only have one set of personalizations per page. *Saved view* features expand on personalization in several important ways:

- Views permit users to have multiple named sets of personalizations per form that they can quickly switch between as needed. This feature allows a user to create multiple optimized views of a page, where each view is tailored to fit the needs of performing a particular business task.
- Views created for particular page types can also include user-added filters or sorts, which allows users to quickly return to commonly filtered datasets. For more information, see What pages support views.
- Views can be published to users in specific security roles and specific legal entities. Therefore, any user who has a specified role and access to a specified legal entity can access and use that view, even if that user doesn't have permission to personalize. This publish capability lets organizations define corporate, standard views that are optimized for their business. For more information, see the Managing personalizations at an organizational level with views section.
- Unlike traditional personalization, views aren't automatically saved when a user performs personalizations or filters a list. Explicit saves are required to give users the flexibility to create a view before or after the changes that are associated with that view are made. This requirement also ensures that view definitions aren't unintentionally changed by filters or personalizations that aren't intended for long-term use. Items that the system automatically stores as part of typical page usage (for example, column widths, or the expanded or collapsed state of sections) are saved per view.
- Views can be added to workspaces as tiles, lists, or links. Therefore, a filtered data set can be surfaced in a workspace, and users can associate a set of personalizations that is relevant to that data set with a tile or link.

Switching between views

After views are made available for an environment, the top of any page that supports views includes a collapsed view selector control that shows the name of the current view.

Build forms that fully utilize saved views

Article • 12/27/2024

Saved views are an important expansion of personalization capabilities finance and operations applications. While the Saved views article provides general details about this feature, this article focuses on the more technical elements of saved views as well as aspects of form development that may be impacted by views.

"User-perceived" pages

Traditionally, a set of personalizations has a 1:1 link to a modeled form. For many pages, this link makes sense to the user, as the user's perception of the page matches the way in which the form is modeled. However, in some cases the 1:1 link of a modeled form to a set of personalizations isn't intuitive or obvious because users don't see or care about the boundaries between modeled forms.

Saved views try to eliminate this confusion by letting users create views on "user-perceived" pages. Therefore, users don't have to understand how forms are modeled to understand how and when personalizations are applied. Consider the following two scenarios:

- **More than one "user-perceived" page in a single modeled form:** The standard modeling of Master Details and Transaction Details forms (for example, the **CustTable** and **PurchTable** forms, respectively) consists of more than one "user-perceived" page: a grid page and a details page.

  Because users aren't aware when this transition from list to details crosses a form boundary, view support in Details forms is handled differently to allow views to be defined separately for the grid and details portions. This means that the view selector for the "grid" and "details" can show different sets of available views. The special casing of view support on these forms also allows the "grid" views to allow filters in their view definitions, whereas the "details" view only needs personalizations.

- **More than one modeled form in a single "user-perceived" page:** The ability to embed subforms into modeled forms (via FactBoxes or form parts) leads to situations where more than one modeled form corresponds to a single "user-perceived" page. For example, consider the details portion of the **All customers** page, which has a number of FactBoxes and two FastTabs whose contents come from form parts. With traditional personalization, contrary to a user's expectations, exporting the personalizations for the **CustTable** form wouldn't include

Test forms that use custom patterns

Article • 12/31/2024

This article how to test forms using custom patterns.

Introduction

By adhering to form patterns, you gain various benefits. For example, form patterns correctly set layout properties so that forms are laid out responsively. However, when form pattern coverage is lacking (for example, there currently isn't support for many extensible controls), or when a form or container has unique requirements/uses that don't fit any pattern, developers can set the pattern to Custom. The developer then becomes responsible for ensuring a correct and responsive form layout.

Forms that use custom patterns

You can find the forms that use custom patterns by using the **Form Patterns** report. For information on running the report, see Form pattern add-ins. After running the report, filter the **Percent covered controls** column to show forms that have less than 100-percent coverage. For forms that have a top-level Custom pattern, **Custom** will appear in the **Patterns** column.

Testing configurations

Key resolution

- 1366 × 768 is a typical resolution on screen sizes that are between 12 and 23 inches. Therefore, this resolution provides a good baseline for testing.
- It's also a good idea to test on a higher resolution, such as 1920 × 1080.

Parameters

- Browser
  - Internet Explorer 11
  - Google Chrome
  - Microsoft Edge
  - Apple Safari (on iPad) – You'll have to point Safari to a cloud URL.
    - Landscape and portrait modes

Create shareable, secured URLs (deep links)

Article • 10/03/2024

Learn how to create shareable, secured URLs to forms and records.

Overview

Developers can create shareable and secured URLs (also known as deep links) to specific forms that are root navigable. An optional data context can be passed to the form to display filtered or specific data when the form is opened. The URL Generator enables scenarios such as embedding links in reports, email, and external applications. The URL Generator enables users to quickly and easily locate the specified forms or data by navigating using the generated link.

Purpose

- Empower developers to generate URLs that can be used to navigate to root navigable forms in a specified instance.
- Empower developers to optionally specify a data context that should be displayed when navigating to the specified form.
- Empower users to share, save, and access the generated URLs from any browser with Internet access.
- Secure the URLs to prevent unauthorized access to the system, forms, or data.
- Secure the URLs to prevent exposure of sensitive data or tampering.

Security

Site access

Access to the domain/client is controlled through the existing login and SSL mechanism.

Form access

Access to forms is controlled through Menu items, as Menu items are the entry points where security is enforced. If a user navigates using a URL that contains a Menu item that the user doesn't have access to, the Menu item security prevents the form from

Accessibility in forms, products, and controls

Article • 12/31/2024

This article describes best practices for enabling accessibility in your form, product, or control. An accessibility checklist is also included.

Accessibility is about inclusion, that is, a person with a disability can perform the same task as a person without that disability. Making an accessible control or form should be as fundamental as making it secure, high-performing, or easy-to-understand.

Keyboard

The bedrock of accessibility is keyboard-only access. When you can use the keyboard to perform all the actions of a form, then it can be used by a nonsighted person or a person with restricted or limited use of their hands. Using a keyboard means that all controls can be reached via the tab sequence, direct action (such as **Ctrl+S** for Save), or through some other shortcut key that enables the user to move to a control, such as Navigation Pane, App Bar, Message Center, or Message Bar. A simple test is to disconnect your mouse and complete all core and secondary scenarios using only the keyboard.

Color

The use of color is encouraged and is a common way to express state or status of a record or other piece of information. However, color can't be the only way that state or status is communicated. An accompanying symbol, help text, or additional column should include a textual description of the state or status. A simple test is to identify all use of color in your system and ensure that color isn't being used to express a state or status. A common example is to use the color red to indicate "needs attention," or the color green to mean an "OK" status.

Images

When an image is displayed, there should be a label that describes the image. If the image expresses state or status of a record, then accompanying help text or an additional column should include a textual description of the state or status. If the image is symbolic, like a logo, then it doesn’t require a textual description. If you have

Customize field descriptions

Article • 08/12/2022

This article describes how you can customize existing field descriptions and add your own descriptions.

There are descriptions for some of the more complex fields. These descriptions appear when you hover over a field. You can customize these descriptions if, for example, you want to add company-specific information. You can also add descriptions for additional fields. You create field descriptions by using the **HelpText** property for field controls. The **HelpText** property is no longer specified for table fields and data types, as it was in previous versions. Additionally, the inheritance of the **HelpText** property from data types and table fields to form controls is obsolete. Field descriptions are intended to be specific to an individual field, in the context of the other controls and information that are available on the page. To add and customize field descriptions, you must have access to the development environment. Like other metadata changes, new descriptions should be added in a new model to prevent them from being overwritten when a new version of Operations is released. For more information, see Customize through extension and overlayering.

Customize a field description or add a new description

The same procedure is used to customize existing field descriptions and to add new field descriptions. However, when you customize an existing description, you replace the existing label reference.

1. In Application Explorer, find the relevant page (form), and add it to your project.
2. In the node for the page, find the relevant field control. Make a note of the name, so that you can use it as part of the label ID.
3. Add a new label for your description. You can follow the conventions that Microsoft uses to name the label files for field descriptions and to create the label file IDs. For more information, see the next section.
4. In the **HelpText** property of the field control, add a reference to the label.

Label file names and label IDs

The field descriptions that are provided by Microsoft are stored in separate label files. There is one label file per module per model. The pattern for the label file names is

Action controls

Article • 01/23/2025

Actions are an essential component of any enterprise resource planning (ERP) system, and are triggered by mouse click, keyboard, or touch.

Introduction

Actions are an essential component of any enterprise resource planning (ERP) system. Actions can be accessed through various mechanisms:

- Buttons on standard Action Panes
- Buttons on Toolbars
- Buttons directly on the form canvas
- Right-click context menus
- Keyboard shortcuts
- The new action search feature

Note that, in general, actions that are triggered by right-click context menus or keyboard shortcuts are meant to have a corresponding button available elsewhere in the user interface. Action controls can be triggered by using touch or a mouse click. Many system-provided actions can also be triggered by using the keyboard.

Buttons

Buttons are the foundation of action controls. They can be modeled inside of standard Action Panes or in Toolbars, which are discussed later in the article. They can also be added as stand-alone buttons on the page (for example, the **OK** and **Cancel** buttons at the bottom of a dialog box, or buttons for actions that are specific to an individual field). The following button types continue to be available:

- A **button** is a basic button, for which the entire functionality must be implemented in code.
- A **command button** specifies a command or task to run.
- A **menu item button** specifies a menu item to navigate to or run.
- A **drop dialog button** opens a flyout dialog box, the contents of which are retrieved via a menu item.
- A **menu button** is a button container that opens a flyout that contains a list of other buttons.

Input controls and grid column sizes

Article • 12/31/2024

This article describes how to create a consistent look and feel for forms by controlling the size of controls and grids.

Overview

Many frameworks offer complete freedom over the width of input controls. However, that level of freedom can lead to inconsistent presentation of data and a non-uniform layout for similar forms. For example, the customer name in one form might show 10 characters of information, whereas the customer name in another form might show 20 characters of information. To provide streamlined, easy-to-read interfaces, discrete sizing helps guarantee consistent presentation of data. The introduction of discrete sizing is a significant change to the basic input controls. The control framework attempts to provide a fresh, clean user experience that provides simplicity and consistency. As part of an attempt to provide consistent and uniform layout of forms, each input control is sized to one of four sizes: extra-small (XS), small (S), medium (M), or large (L). These sizes are determined by inspecting the explicitly specified width in the **DisplayLength** property of the control or the corresponding extended data type (EDT).

Sizing grid columns

Column sizing in a grid control differs slightly from the legacy algorithm, which tried to provide an initial appealing presentation for each grid, based partly on the contents of the first page of data. The new approach disregards the contents of the first page of data. Instead, the end user decides which columns are most important to view and the ideal viewing width for each of those columns. By using personalization, each user can change the width of grid columns, and the client keeps track of that user's preference. In the new, simplified approach, the default column sizing is based on 75 percent of the base input control sizing. In most cases, we recommended that developers not override the default sizing. However, if you must resize the column width programmatically or in the model, see the next two sections of this article for guidance.

Discrete sizing

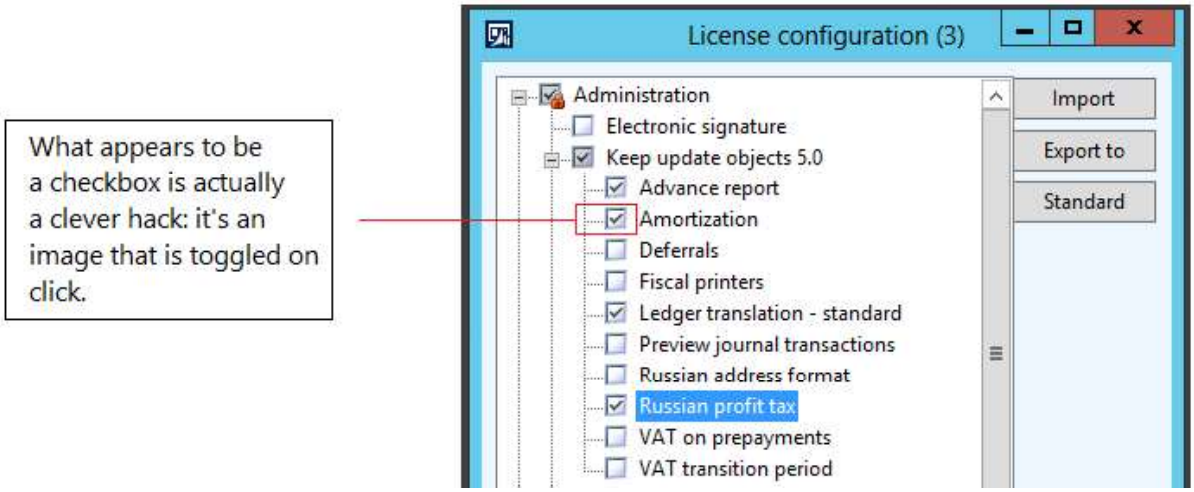
Expand table

Check box support in tree controls

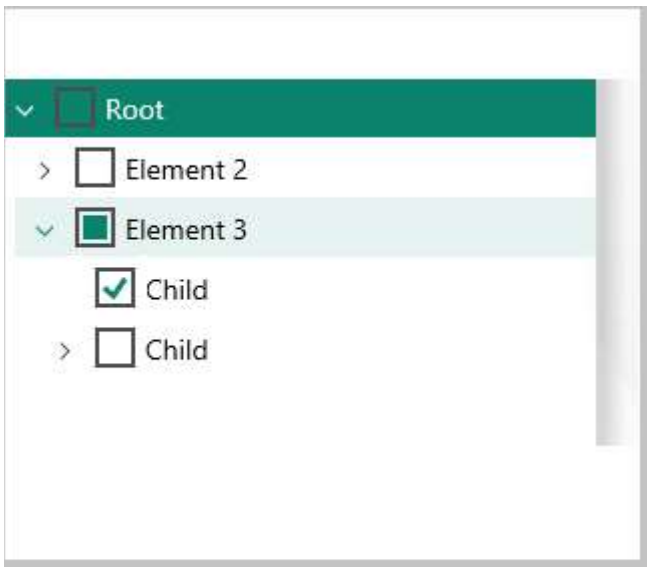
Article • 12/31/2024

This article is intended as a primer for using check box controls in the tree control. It's not a general “how to” for using tree controls.

Microsoft Dynamics AX 2012 includes several examples of tree controls that were enhanced so that they both show data in a tree hierarchy and let the user select one or more nodes by using check boxes. In Dynamics AX 2012, the tree control had no built-in support for check box controls. Instead, an image of a check box was added for each node in the tree control. The image state for each node was then toggled as the user clicked the check box.



The current version has greatly simplified the experience for the developer. Check box support is now built into the tree control.



You no longer have to use images to include a check box, and you also don't have to explicitly set the state of the check box state when it's selected. The control doesn’t use

Filtering options

Article • 06/19/2024

This article explains the filtering options that are available.

Introduction

Microsoft Dynamics 365 finance and operations offer the following filtering options.

Expand table

Filter option	Description
Filter by grid	The user defines filter conditions in input fields below the grid column headers.
Filter by selection (filter by field)	The user selects a field value and uses that value as a filter condition.
Advanced filter	The user opens a dialog box that contains advanced filtering options (filter on columns, not on the form; join additional data sources; sort by multiple columns; and so on).

Finance and operations offers the following filtering options.

Expand table

Filter option	Description
Filter pane	An inline pane that slides in from the left, and that contains multiple filter criteria that can be applied to the targeted content.
QuickFilter	A framework-provided filtering mechanism that can appear above any list or grid, and that provides fast single-column filtering.
Grid column filtering	The user can define filter conditions and perform single-column sorting by using a drop dialog that is opened from the grid column header.
Advanced filter or sort	For most advanced filtering scenarios, the migrated **Advanced filter** page is available.

Filter expressions

Power Apps Host control

Article • 12/31/2024

In Microsoft Power Apps, you can manage organizational data through apps that you created, or apps that someone else created and shared with you. Apps run on mobile devices such as phones or in a browser. Apps can also be embedded in finance and operations apps by developers using the Microsoft Visual Studio developer experience. To learn more about Power Apps, see https://powerapps.microsoft.com.

Host an app from Power Apps on a page

1. In Power Apps, find the web-based app that you want to host, and record or copy the **App ID** value.
2. In Visual Studio, open your project, and then, in the form designer, add an instance of a Power Apps Host control to your page.
3. In the **Properties** pane, enter the **App ID** value.
4. If your app shares or is linked to the current data source on your page, you can pass the ID of the primary or linked key field for the data that you want your app to show. In this case, provide the ID as the value of the **Entity ID**, **Entity ID Data Source/Field**, or **DataMethod** property. This value will then be passed to your app as a parmeter value, and your app must use that value to obtain the linked data.
5. In some cases, your app might be hosted in a development or sandbox Power Apps environment that is provided by Microsoft. In this case, you must supply that override URL as the value of the **Power Apps Environment Override** property.

Sizing is determined by the container that you put your control in. If you put your control in a form pattern that has limited available space, and your app has been designed to be larger than the available space, your embedded app will have scroll bars.

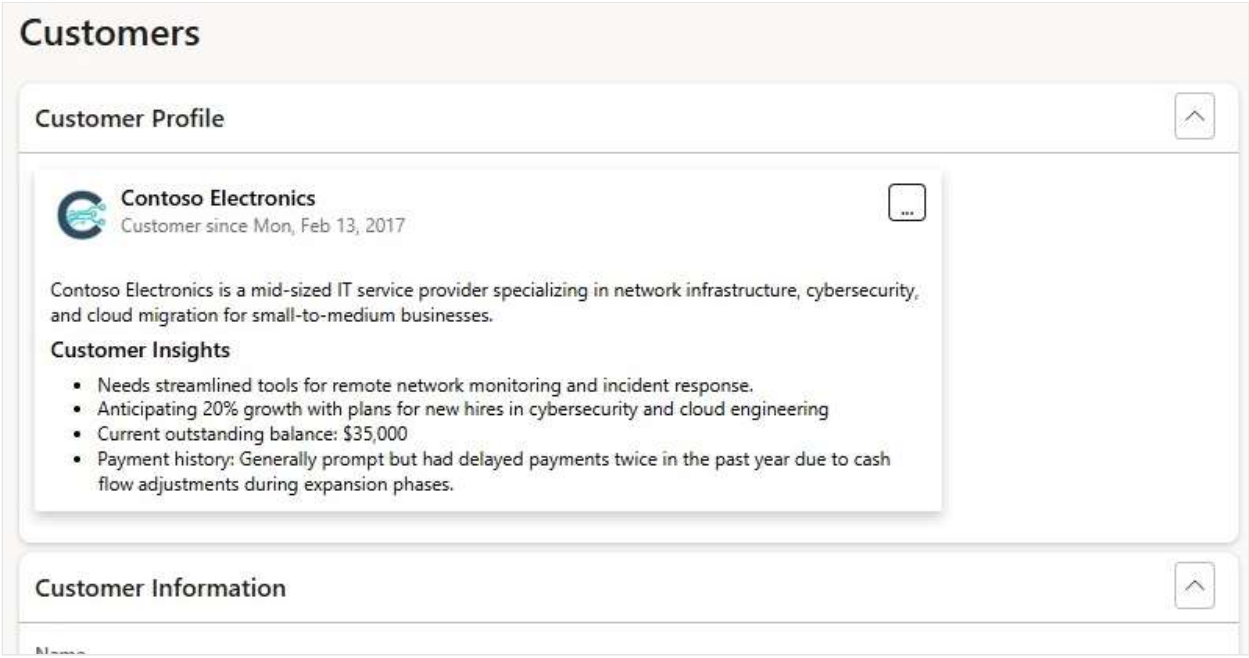
Feedback

Adaptive Card controls

Article • 11/05/2024

Adaptive Cards are a platform-agnostic way for developers to create rich, interactive, card-like interfaces that can be embedded in various applications, such as Microsoft Teams, Outlook, Copilot Studio chatbots, and other chat or web applications. The key benefit of Adaptive Cards is that developers can design a card once and then use it across multiple platforms without having to adjust the layout for each one. The open card exchange format lets developers describe content as a simple JavaScript Object Notation(JSON) object. Therefore, it's easy to structure and manipulate cards dynamically.

The Adaptive Cards control for finance and operations apps lets developers render Adaptive Cards natively on application pages. Learn more on the Adaptive Cards site, which includes documentation, samples, and a designer that you can use to start to build your own Adaptive Cards.



Using Adaptive Cards

To use an Adaptive Card on a page in finance and operations apps, follow these steps.

1. Define the data contract for the data properties that populate the Adaptive Card.
2. Configure the Adaptive Card template to define the card's layout and formatting.
3. Add the Adaptive Card control to the page.
4. Define the data that populates the card.

Code migration - Context menu code

Article • 12/31/2024

A programming model is required for context menus (shortcut menus). This article outlines the process for migrating context menu code from Microsoft Dynamics AX 2012 to finance and operations. It also includes user experience (UX) guidelines for context menus.

In Dynamics AX 2012 and earlier versions, developers modified right-click context menus (shortcut menus) by using the **PopupMenu** class. This class relied on Microsoft Windows application programming interfaces (APIs) that aren't available on the web. In finance and operations, the **ContextMenu** APIs have been created as replacements to provide similar functionality. Previously, the **context()** and **showContextMenu()** method overrides were the entry points for modifying context menus for specific controls. These overrides typically contained code to add options to the context menu, and also to process the user's selection. The code for processing the user's selection used a wait model. Because these overrides are being removed and the wait model is being eliminated, developers must now create two overrides: **getContextMenuOptions()** to add options to the context menu and **selectedMenuOption()** to process the user's selection.

Migrate context menu code

Migration from the **PopupMenu** APIs to the **ContextMenu** APIs can be broken down into three main steps.

Step 1. Add a constant for each menu option that must be added

The old **insertItem()** method in the **PopupMenu** class returned an identifier for the menu option that was being added. This identifier was saved into a variable for future reference. Because developers will define the menu identifier, it's a good idea to define constants for each option to help with code readability.

- At the form level, add a constant for each menu option that is being added to the context menu. The value must be unique within each context menu. Note that you must modify the old variable name if it conflicts with another variable on the form or control.

Code migration - Mouse double-click logic

Article • 12/31/2024

In finance and operations, the **mouseDblClick()** override has been deprecated, and you will need to move this logic to new controls.

In Microsoft Dynamics AX 2012, the mouse double-click event was used for various reasons. For example, it helped provide a better user experience and provided an alternative way to run certain scenarios. Here are some examples of common usage patterns:

- Moving elements between two lists or tree controls
- Opening a new form to get more details about the selected field
- Running complex business logic
- Selecting a field in a lookup

Strategy overview

Before you begin to use the form, it's a good idea to fix all best practice warning messages that state, "The mouseDblClick control method has been deprecated and should not be used." Otherwise, the form might be useless, or it might work only in limited ways.

Migrate code from mouseDblClick() methods

As we mentioned earlier, there were various reasons for using the **mouseDblClick()** method in Dynamics AX 2012. This section explains how to migrate some of the most common scenarios.

Moving items between two lists controls

In Dynamics AX 2012, a mouse double-click was often used in List Panel scenarios, where two list controls appeared side by side. Often, when a user double-clicked an item in one list control, that item was moved to the second list control. Migration of this **mouseDblClick()** scenario involves alignment to the List Panel pattern. You have two options for migrating this usage pattern:

Contextual data entry for lookups

Article • 01/23/2025

In data entry scenarios, it is common for a user to attempt to identify an entity in terms of some more descriptive or natural language attribute if that entity is formally identified by a synthetic key, such as a number sequence. The contextual data entry feature allows users to type in either the synthetic key or a more descriptive attribute directly into a lookup field. This page explains how contextual data entry works and also provides implementation details and tips for developers who want their lookups to have this behavior.

Introduction

In data entry scenarios, it is common for a user to attempt to identify an entity in terms of some more descriptive or natural language attribute if that entity is formally identified by a synthetic key, such as a number sequence. A user will typically attempt to enter an **Account Name** instead of an **Account ID** for the **Customer Account** when creating a Sales Order. This is because most interaction with a customer is done using their actual name instead of some synthetic identifier. Unfortunately, any user's attempt to enter an **Account Name** will fail because the **Customer account** control's underlying foreign key relates to a field that is a synthetic key—a number sequence—and Dynamics AX 2012 (and older) will always attempt to validate the entered value directly. Therefore, if the **Account ID** was unknown to the user, the user would be forced to perform some type of searching step, such as opening the **Customer account** control's lookup and filtering on the **Account Name** column to identify the correct **Account ID** (see the image below).

HierarchyViewer control

Article • 01/03/2025

This article provides information about the HierarchyViewer control, which lets you represent hierarchical relationships for people, products, or organizations.

The HierarchyViewer control lets you represent hierarchical relationships for people, products, or organizations. It's used primarily as a graphical means to help you understand hierarchical relationships in a traditional top-down manner, and as way to navigate to the entity that is represented by the focused node. The HierarchyViewer control lets you walk through deeply nested, multilevel content in a compact space. The control expands and collapses nodes to control the parts of the tree structure that are shown. Because it's an unbound control, the HierarchyViewer data is managed by an abstraction class and is used primarily as a way to visualize data in a simple tree relationship. For hierarchy data in a traditional tree, there's a standard tree control.

The HierarchyViewer control shows four levels of information at any given time. The current node is the current focus of the tree, which isn't necessarily the root node. The current node is represented by the largest physical node in the current view and it has a colored bar on the left. Above the current node is a trail of smaller parent nodes from the root node down to the current node. Below the current node is a level of children nodes, and there can be an indefinite number of nodes at this level. By default, three children nodes are shown at a time on each page, but you can change this by adjusting the **Number of children** property. The **Next** and **Previous** link buttons allow the user to page to other nodes at the child level. Finally, there's a level of grandchildren nodes that are shown for each child node. Each child can have an indefinite number of grandchild nodes, and the number of grandchildren shown at one time for each child node is controlled by the **Number of Grandchildren** property. Users can use the **Next** and **Previous** arrow buttons to page up and down through members at the grandchild level. The interactive display of nodes requires no business logic.

Business logic interaction

The read-only HierarchyViewer control offers data visualization and navigation. It can be used to select an entity (employee, product, or organization), and corresponding data can then be managed though other display and input fields on the form outside the HierarchyViewer control.

```
X++
```

Lookup controls

Article • 12/31/2024

This article discusses how to enable lookup behavior on controls. It also discusses how to create multi-select lookups and outlines lookup scenarios that are no longer supported.

Enabling lookup behavior in controls

Controls bound to an Extended Data Type

Controls with their Extended Data Type property set (no FormDataSource in play) have a lookup under the following conditions:

1. If the EDT has its Table Relations or Table References nodes populated.
2. If the FormHelp property is set (custom lookup); doesn't require rule #1 to be true.
3. If the control has lookup or lookupReference methods overridden. This rule also applies to fully unbound controls (no EDT, field, or data method). This condition includes overrides via registerOverrideMethod and others.

Controls bound to a form data source

Controls that are bound to a data source can have a lookup under the following conditions: **Field bound**

1. The "lookup" or "lookupReference" (Reference Controls) methods are overridden.
   a. If the FormDataSource field has lookup or lookupReference methods overridden.
   b. If the control has lookup or lookupReference methods overridden.

      - This condition includes overrides via registerOverrideMethod and others.

2. If the field has an EDT, then rule #2 from the "Controls bound to an Extended Data Type" section applies.

3. If the bound field maps to a relation per DBFGetRef rules.
   a. High level rules:

      i. If there's an EDT relation backing the field, with the Table Relations node populated and Ignore EDT Relations is false on the field, the relation is used (has a lookup).

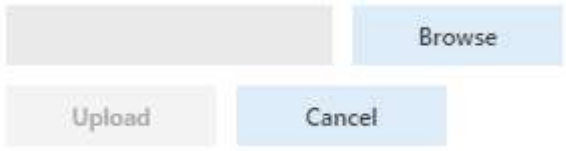
File upload control

Article • 12/31/2024

This article provides information about the file upload control. This control lets users upload files.

Overview

The file upload control lets users upload a file. It also lets developers control the upload process and manage the file that is uploaded, based on their requirements.



The file upload control can have three styles. You control the style by using the **Style** property.

- The **Standard** style shows the file name field together with **Browse**, **Upload**, and **Cancel** buttons.
- The **Minimal** style shows only the **Browse** button.
- The **MinimalWithFileName** style shows the file name field and the **Browse** button.

The **FileTypesAccepted** property of the file upload control lets you limit the types of files that users can upload. The file types that users can upload are primarily controlled by the associated upload strategy. The **FileTypesAccepted** property on the file upload control should be used only if further restrictions are required. If the upload control tries to specify file types that are restricted by the upload strategy, the **Browse** button becomes unavailable.

Expand table

Allowed file types	Allowed file types from the upload strategy	Final result
".jpg,.png"	".jpg,.png,.gif,.txt"	".jpg,.png"
"image/png"	"image/*"	"image/png"
"image/*"	"image/png"	The **Browse** button is unavailable.

System-defined buttons

Article • 12/31/2024

This article describes the system-defined buttons.

Overview

Several system-defined buttons are automatically present on the Action Pane. In general, these system-defined buttons should be applicable and should kept available to the end user. However, in rare cases (for example, if a more specialized control is required, or if a system-defined button isn't useful or applicable for a particular form), developers might have to explicitly suppress or override a system-defined button. For example, in some situations, a MenuButton that lets the user select from multiple "New" options might be preferable to the system-defined **New** button.

List of system-defined buttons

The following tables give the full list of system-defined buttons. The tables also provide information that will be useful if these buttons must be conditionally or completely suppressed or overridden.

Common buttons

Expand table

Button	Button name macro*	Comments
Export		Don't suppress this button.
Attach	#SystemDefinedAttachButton	Don't suppress this button, because we will suppress it on forms that aren't set up for attachments.
Show filters	#SystemDefinedShowFiltersButton	By default, **Visible**=**No** on TOC forms.

* System-defined button name macros are found in the SysSystemDefinedButtons macro file.

Buttons that are specific to Details forms

Images on a page or in a grid

Article • 12/31/2024

This article describes the steps for displaying images on a page or in a grid. The article also provides background about some of the ways that images can be used, and the APIs that are used.

> **Note**
>
> For accessibility, when you use an image to indicate status or show data, the image must be accompanied by a tooltip, enhanced preview, label, or other textual representation that describes the value or status that the image represents.

Finance and Operation apps do not use embedded resources for images. Instead, it uses lightweight symbols. The coding pattern has changed slightly to support the new image control.

For ImageList uses, the runtime accepts the old **ImageID** value and maps it to a symbol, so that existing code continues to work.

> **Note**
>
> In some cases, there is no image even after runtime mapping, and this behavior is intentional.

AX 2012 displays images in a grid column to indicate status. These images were sometimes retrieved from embedded resources that are no longer available.

AX 2012 offers the following storage options for images:

- An embedded resource where images are offered as part of the kernel itself
- An Application Object Server (AOS) resource where developers or independent software vendors (ISVs) can add their own image resources
- A file location where developers or ISVs can load images at run time
- A database field that is stored as a bitmap

The following storage options are available for images:

- An AOS resource where developers or ISVs can add their own image resources
- A URL location where developers or ISVs can load images at run time
- A database field that is stored as a container.

Font and background colors for input, table, and grid controls

Article • 12/31/2024

This article provides information about the new color picker control that lets users select a color.

Traditionally, color has been considered an ideal way to communicate with a user. For example, the color red is often used to draw the user's attention to information that is important. However, some users can't distinguish certain colors or shades, and some users are blind. Therefore, we don't recommend that you use color alone to communicate information to the user. Instead, you should use color together with a symbol or additional text to convey information to all users.

Color selection in Dynamics AX 2012

In Microsoft Dynamics AX 2012, color selection had these characteristics:

- It used the Win32 color picker.
- It required Win32 application programming interfaces (APIs) for RGB/decimal conversion. (The input control accepted a decimal value for RGB.)

X++

```
Public void lookup()
{
    #DEFINE.COLORVALUE(64)
    Int r,g,b
    container choosencolor;
    Binary customcolors = new Binary(#COLORVALUE);
    CCColor colorvalue;

    Super();

    [r,g,b] = WinAPI::RGBint2Con(this.backgroundColor());

    chosenColor = WinAPI::chooseColor(element.hWnd(),r,g,b, customColors,
true);

    If(chosencolor)
    {
        [r, g, b] = chosencolor;
        Colorvalue = WinAPI::RGB2int(r,g,b);
        This.backgroundColor(colorValue);
        employeeWorkPlannerForm.parmAbsensceColor(colorvalue);
```

Right-to-left language support and bidirectional text

Article • 12/31/2024

In the area of right-to-left (RTL) language support, one consideration is the combination of RTL text and left-to-right (LTR) text in the same string. This article discusses the issue of bidirectional text and how it's handled.

A great example of right-to-left language support: Microsoft Word

In the area of right-to-left (RTL) language support, one consideration is the combination of RTL text and left-to-right (LTR) text in the same string. One example of a program that implements this functionality correctly is Microsoft Word. If you're trying to understand the correct behavior of mixed language presentation, you can use Word for validation. The problem is that most software just implements the Unicode standard to display bidirectional data, without evaluating how that data is actually used. Additionally, there's no attempt to provide the interactive experience that the user actually requires.

To understand how Word “gets it right” and provides a great experience, you can inspect the XML of a Word document. There, you will see that Word tracks (and stores together with the run of characters) the keyboard that is used to enter each character, and that it treats each character as a member of the language that is associated with the keyboard. Therefore, the character is given the behavioral aspects of that language.

Keeping track of character orientation in a financial program that might record billions of transactions and multi-billions of characters would produce significant transnational and spatial overhead if we stored contextual information for each character. Therefore, this behavior would be considered only for special conditions.

Bidirectional text

To support Arabic and Hebrew, both of which are RTL languages, there is an RTL orientation for the controls in each form, so that an RTL reader can interact with the form in a natural reading manner. For the most part, RTL orientation of the controls works as expected and provides RTL users with the experience that they expect. Finance and operations apps and modern browsers support RTL orientation, and the apps

Create icons for workspace tiles

Article • 01/23/2025

This article provides guidelines and recommendations for creating and assigning icons to custom workspace tiles.

The dashboard contains a set of workspace tiles to which the user has access. Each of these tiles contains an icon specific to that workspace. For out-of-the-box workspaces provided by Microsoft, the icons used on the workspace tiles generally correspond to a symbol from the Dynamics Symbol font. This article discusses the guidelines and recommendations for creating and assigning icons to tiles for workspaces created by Microsoft Certified Partners or individual customers.

Implementation details

For workspace icons, we recommend using an AOT resource for the icon. While the out-of-the-box symbols will work, we recommend creating your own so that multiple workspaces don't use the same icons. For each workspace that needs an icon, create a new image file that adheres to the guidelines below. Note that the recommended guidance for newer versions of the product has changed.

Modeling details

When you create a workspace tile, you need to follow these guidelines:

- Add an AOTResource for each new icon.
- On the tile corresponding to the workspace, set the following properties:
  - ImageLocation=AOTResource
  - NormalImage=<name of AOTResource>

Icon creation

Guidelines for creating images for custom workspace tiles are below. The recommended dimensions for the image and icon are based on the out-of-the-box workspace icons. While images of other sizes are allowed, the size and positioning of the icon relative to the full image should be maintained regardless of the image size.

Following these recommendations ensures that your workspace icon matches the styling and size of other workspace icons and that the content of your workspace icon does not get cropped by the CSS applied to the image.

Public JavaScript APIs for extensible controls

Article • 03/11/2025

This article shows where to find documentation about the public JavaScript APIs that can be used by extensible controls.

To minimize future breaks in extensible controls, an effort was made to differentiate between the public and nonpublic JavaScript application programming interfaces (APIs) available to extensible controls. Control authors should ensure they only use public APIs, as **any nonpublic API may be removed or modified in a future release**. One of the planned modifications is to prefix the names of the nonpublic APIs with underscores to clearly denote access level. Documentation for the full set of public APIs can be found in Extensible Controls - Public JavaScript APIs.

Feedback

Control checklist

Article • 08/12/2022

This article categorizes and describes all the release criteria for controls.

Introduction

Typically, when you author a new control, the primary focus is on scenario functionality and technical implementation. However, before a control can be considered ready for shipment, it should conform to a set of best practice, quality, and development release criteria, as outlined in this article.

Control criteria checklist

This checklist assumes that you're familiar with the basics of control development. The following items highlight important implementation requirements that should be met by all controls.

Basic usage ready

A control must meet these requirements to be considered a functionally compatible and complete control.

Classes

These naming conventions are best practices but aren't functional requirements of a control:

- The Runtime class is named **[Name of control]Control**.
- The Design-time class is named **Build[Name of control]Control**.
- Any control-specific Component classes are named **[Name of control][Name of component]Component**.
- Any generic Component classes are named **Build[Name of component]Component**.

Resources

These naming conventions are best practices but aren't functional requirements of a control:

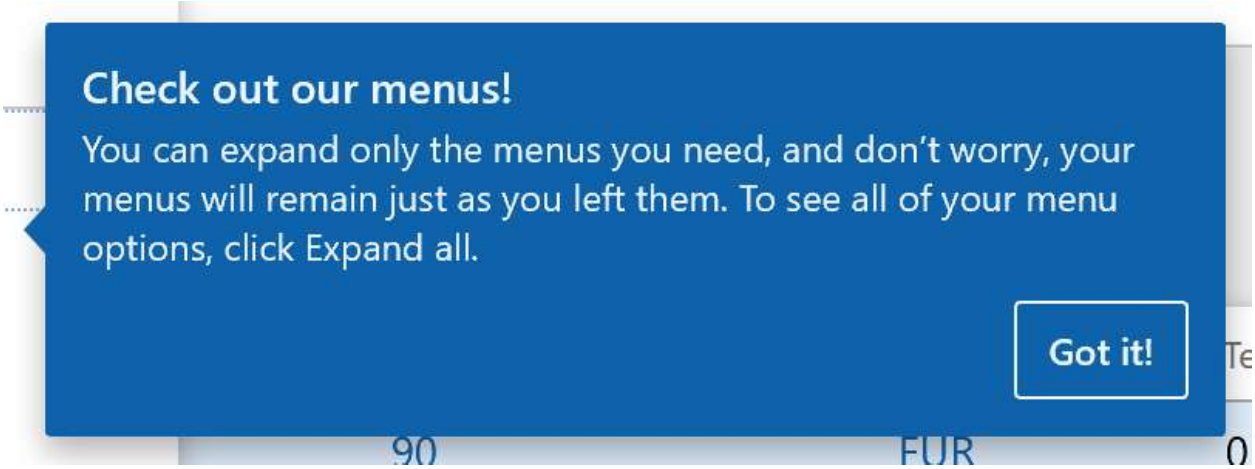
Feature callouts

Article • 12/31/2024

Introduction

While documentation is helpful for explaining new features, it's also important to raise awareness of these new capabilities as users encounter the feature while using the product. As a result, feature callouts are available in Platform update 26. You can use feature callouts to point out a new capability to a user and optionally provide a hyperlink for the user to learn more about the feature.

In this article, the APIs that are used to construct feature callouts are discussed in detail.



The "Got it" button

When a feature callout is triggered, the user can simply click the **Got it** button to dismiss the popup. This saves the state of this feature callout in the personalization subsystem, which prevents that specific feature callout from being triggered again.

Resetting feature callouts

Even though the feature callout state is stored in personalization, clearing personalizations will not delete the state of all previously dismissed feature callouts. Instead, separate actions have been added to reset all feature callouts so that they fire again. These actions are located on the **Personalization** tab on the **Usage data** page as well as on the **Manage per user** tab on the **Personalization** page.

Disabling feature callouts

Slider and MessageBox dialogs

Article • 01/03/2025

Dialogs

There are two dialogs that replace the existing dialog box, the Slider and the MessageBox from Dynamics AX 2012:

- Slider
- MessageBox

The following sections discuss the specific goals for each concept.

Slider

The slider, or slider dialog, is a dialog box that "slides" in on top of the active page's content from the right edge of the screen.

After a slider opens, the user can dismiss it in two ways:

- Perform an action within the slider that causes the underlying form to dismiss itself. For example, click **Cancel**, or enter required information and then click **OK**.
- Click outside the slider in the shaded area to the left. This cancels the slider, and no further actions are performed.

A slider contains a modeled form and is used to gather information from the user. Therefore, a slider should be used in most situations where a dialog box has been used in the past. For example, a slider is typically used when the user creates a new record, as in the preceding screen shot. However, a slider should not be used for simple notifications or messages to the user. For these situations, a MessageBox should be used, as described in the next section. To model a slider, you create a form, and then set the **Style** property to **Dialog** on the **Form.Design** node. You then model the form elements that you require (for example, fields and buttons). The caption is defined by **Form.Design.Caption**. To simplify the process for creating sliders, we have provided the **SysBPStyle_Dialog** form as a template for modeling slider dialogs. To use this template, copy it into a new form, and then extend it as you require.

MessageBox

A MessageBox is a type of dialog that is rendered as a "lightbox" on top of an existing page. A MessageBox appears as a full-width modal pop-up.

Messaging APIs - Action center, message bar, and message details

Article • 01/21/2025

This article describes the messaging system in finance and operations apps, specifically in terms of the application programming interfaces (APIs) that are used to create and route messages to end users.

Introduction

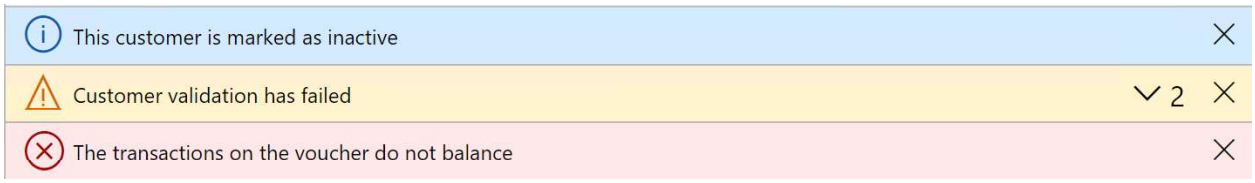
A new messaging system was created for finance and operations apps to improve this experience. Compared to earlier versions, the messaging system for finance and operations apps includes the following features:

- Improved association of a message with its context (form versus global).
- Improved level of interruption (none, subtle, and interrupting).
- Improved clarity between types of messages and their use.
- The control that is used to display messages is deterministic and based on form context.

Backwards compatibility of info(), warning()/checkfailed(), and error()

The **info()**, **warning()**, and **error()** application programming interfaces (APIs) from earlier versions of finance and operations apps are still supported; however, these APIs now sit upon the framework's new messaging system. Messages are routed deterministically to the message or Action center (in a non-interrupting manner) by using the context of the API call to determine the best way to present the message to the user. In general, if the use of the API originated from a form, the message appears in a message bar on that same form. (Drop dialogs and slider dialogs are both considered forms.)

The following illustration shows **info**, **warning/checkfailed**, and **error** message bars that correspond to page actions, or synchronous-authored messages that come from **info()**, **warning()**, and **error()**.



Messaging system for finance and operations apps

Article • 01/19/2024

This article describes the rich, powerful messaging system in finance and operations apps.

A new messaging system was created for finance and operations apps to improve this experience. Compared to earlier versions, the messaging system for finance and operations apps includes the following features:

- Improved association of a message with its context (form versus global).
- Improved level of interruption (none, subtle, and interrupting).
- Improved clarity between types of messages and their use.
- The control that is used to display messages is deterministic and based on form context.

Where can messages be surfaced to users?

Messages in finance and operations apps are generally shown in one of these places: message bars, the Action center, or message boxes.

Message bars – Messages for synchronous tasks on the current page

Message bars are available on primary pages, and in drop dialogs and slider dialogs. Message bars are used primarily for data validation. They can also be used to communicate messages about the state of a page or data, such as messages that are used for date effectivity. Message bars can express **info**, **warning**, and **error** statuses. Message bars should not be used for messages that require the user's immediate attention. A message bar appears when a message is first received and must be used to communicate messages only about the current page. Messages that are sent to message bars are associated with the current page. Therefore, when the user navigates away from a page that includes message bars, those messages won't appear on the new page. However, if the user navigates back to the original page, the page's messages will once again appear. Include the following information in messages:

- The condition that generated the message.

Form patterns for migrated forms

Article • 01/23/2025

This article provides information that will help you select the best form pattern for the forms that you migrate.

Introduction

The selection of a form pattern is an important step in the process of migrating a form. A pattern that is a good fit for the target form reduces the amount of migration work that is required. By contrast, a pattern that isn't a good fit can cause wasted time and effort. Therefore, it's important that you do some investigation, so that you can select the best form pattern for the form that you're migrating. Here is some guidance and tips for determining the appropriate pattern for a form:

- Investigate the form's metadata in the form designer. Pay close attention to the following details:
  - Form name
  - Form.Design.Style
  - Control names
  - The way that the controls are organized
  - The number and names of the data sources
- Investigate the form's visuals by running the form and looking at the way information is displayed.

Selecting a form pattern via metadata

Use Form.Design.Style for guidance

The **Form.Design.Style** property often contains the name of the pattern that was previously targeted for the form. If the **Style** property correctly matches the metadata, you can use the following table to find a pattern that is likely to be a good fit for the form.

Expand table

Form.Design.Style value	Corresponding pattern
DetailsFormMaster	Details Master

Form styles and patterns

Article • 12/31/2024

This article describes the concept of form patterns and discusses the process for applying and removing patterns. A list of frequent questions are also answered in this article.

Dynamics AX 2012: Form styles and templates

In Microsoft Dynamics AX 2012, several form styles were introduced and formalized. Primary data types are represented by the List Page and Details Form styles. Secondary data types are represented by the Simple List and Details Form and Simple List Form styles. In addition to these core form types, other form styles exist for supporting forms, such as Table of Contents for settings and Drop Dialog for dialog forms, and Lookup for lookup forms. Other less formal form patterns, such as Wizard, also exist. Developers who wanted to build a new form of a specific style in Dynamics AX 2012 often used the corresponding template form as a starting point. After they included form content and made any modifications that were required, developers could then run the Form Style Checker add-in to validate their form in terms of structure and property values against that form style's template form.

Finance and operations: Form patterns

Form patterns (a new concept that is the evolution of the Dynamics AX 2012 form templates, style, and Form Style Checker) are now an integrated part of the form development experience. These patterns provide form structure, based on a particular style (including required and optional controls), and also provide many default control properties. In addition to top-level form patterns, subpatterns can be applied to container controls, and that provide guidance and consistency for subcontent on a form (for example, on a FastTab). Patterns have made form development easier by providing a guided experience for applying patterns to forms to guarantee that they're correct and consistent. Patterns help validate form and control structures, and also the use of controls in some places. Patterns also help guarantee that each new form that a user encounters is immediately recognizable in appearance and function. Form patterns can provide many default control properties, and these also contribute to a more guided development experience. Because patterns provide many default layout properties, they help guarantee that forms have a responsive layout. Finally, patterns also help guarantee better compatibility with upgrades. Many of the existing form styles and templates from Dynamics AX 2012 continue to be supported. However, legacy form styles and

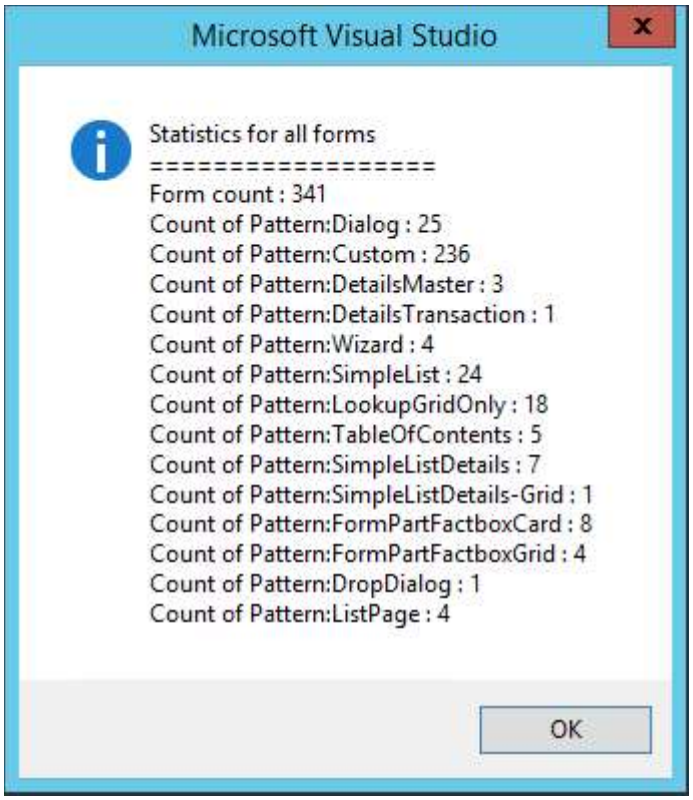
Visual Studio add-ins that support form patterns

Article • 12/31/2024

The tools for Visual Studio include a number of add-ins that support pattern usage.

Form statistics add-in

The **Form statistics** add-in provides a summary of the pattern usage for forms. When you access the **Form statistics** add-in from the **Dynamics 365** menu, it displays statistics for all forms. When you access the add-in from the shortcut menu for a form that is open in the form designer, it displays statistics for that form only.



Forms Pattern report

The **Form Patterns** report provides pattern information about every form, including whether the form uses a top-level form pattern, is a custom form, or is not specifying a form pattern. To generate the **Form Patterns** report, start Microsoft Visual Studio, click the **DYNAMICS 365** menu, expand **Add-ins**, and then click **Run the form patterns report**. The process will take several seconds. After the report has been generated, a dialog box will provide the location of the report. Browse to the specified location, and

General form guidelines

Article • 01/03/2025

This article contains the guidelines that apply to all forms, regardless of form pattern. This checklist must be used in addition to any pattern-specific guidelines.

Verification checklist

The verification checklist shows the steps for manually verifying that the form complies with the UX guidelines. This checklist doesn't include any guidelines that will be enforced automatically through the development environment. Open the form in the browser, and walk through these steps.

Standard form guidelines

Specific form patterns and subpatterns might have exceptions to these guidelines.

- The form layout is responsive when the browser is resized or the app is run on different device sizes. In other words, all the fields should be accessible to the user either by reflowing the layout or by scrolling to the fields.

- Make sure that the form's default View/Edit state is correct. By default, forms are in View mode. If a form should always be in Edit mode, you must explicitly set **Form.Design.ViewEditMode=Edit**.
  - The View/Edit state should be appropriate to the state of the entity. For example, if the state of the entity is **Posted**, and the form can't be edited, the default state should be View mode (and Edit mode should be disabled).

- **Form captions**:
  - Avoid setting the form caption programmatically. Instead, consider setting the **TitleDataSource** property on the form design node to enable the framework to provide the caption dynamically.
  - If you can't avoid setting the form caption programmatically, make sure that it's short (no more than 30 characters). This guideline exists because a large font size is used for the form in some form types.
    - **Exceptions:** Custom Lookups, FactBoxes
  - Form captions should provide the user with the context of the "type" of entity. The font size and the position of the form caption will vary, depending on the type of form.

Details Master form pattern

Article • 01/03/2025

This article provides information about the Details Master form pattern. A details form is the primary method for entering data.

Usage

A details form is the primary method for entering data. These forms let the user view, edit, and act upon data. All content on these form types is structured into FastTabs that can be expanded and collapsed, so that multiple FastTabs can be open at the same time. The FastTabs can contain fields or a grid, and each FastTab can have a local toolbar. Two patterns are described in this document:

- **Detail Master –** This is the basic Detail Master pattern. This is the pattern that you should use by default.
- **Detail Master w/ Tabs –** You should use this pattern when an entity requires many FastTabs (more than 15) that can be grouped into categories.

In both cases, the grid view is structured the same.

Pattern changes

Here are the main changes to this pattern since Microsoft Dynamics AX 2012:

- Added a List style grid to the left of the Details view content.
- Merged List Page and Details Master into a single form.
  - Improves performance when moving between a list and details.
  - Enables bulk editing in the initial list.
  - Allows for elimination of the list page preview pane.
- View/Edit, New, Delete, Save, Refresh, Attachments, and Export to Excel actions are all provided by the foundation and should not have explicit app buttons unless the foundation-provided button is removed.
- Master Details forms that previously used the TOC extension should now use the Master Details w/Standard Tabs pattern.

Model

Details Master (basic) – High-level structure

Details Transaction form pattern

Article • 01/03/2025

This article provides information about the Details Transaction form pattern. Forms that use this pattern can have two details views that the user can switch between - a Header view and a Line view.

Usage

A details form with lines (Details Transaction form) consists of one form that can have two details views that the user can switch between. The Header view contains all fields that are related to or part of the header. The Line view contains the lines grid, line details, and a section that contains a collection of the most important header fields.

Pattern changes

Here are the main changes to this pattern since Microsoft Dynamics AX 2012:

- A List style grid has been added to the left of the Details view content, which is shown in either Header view or Line view.
- List Page and Details Master have been merged into a single form. This change has the following benefits:
  - It improves performance when users move between the list and details.
  - It enables bulk editing in the initial list.
  - It allows for elimination of the list page preview pane.
- View/Edit, New, Delete, Save, Refresh, Attachments, and Export to Excel actions are all provided by the foundation and should not have explicit app buttons unless the foundation-provided button is removed.

Model

High-level structure

- Design
  - ActionPane (ActionPane)
  - SidePanel (Group)
    - QuickFilter

Form Part Section List form patterns

Article • 01/03/2025

This article provides information about the Form Part Section List form patterns. These workspace-specific patterns have been developed to show filtered lists inside workspaces.

Usage

The Form Part Section List form patterns are workspace-specific patterns that are used to show filtered lists. The tabbed section of the workspace contains a set of vertical tabs. Each tab contains a Form Part Control that points to a form that contains one of the Form Part Section List patterns. Two patterns are described in this article:

- **Form Part Section List –** This is the default Section pattern. It allows for a single list of data, together with an optional header group that contains filters and/or actions. Most content areas in the tabbed section of a workspace will use this pattern.
- **Form Part Section List - Double –** This variant enables a second list of data to appear to the right of the primary list. By default, the secondary list is hidden. To show it, the user clicks a button on the Toolbar above the primary list.

Pattern changes for finance and operations apps

These patterns did not exist for Microsoft Dynamics AX 2012.

Model

Form Part Section List: High-level structure

- Design | Container
  - *Header (Group) [Optional]* – This must use one of the Filters and Toolbar subpatterns.
  - Grid
  - *GridDefaultAction (Button) [Optional]*
  - *SeeMoreButton (Button) [Optional]*

List Page form pattern

Article • 01/03/2025

This article provides information about the List Page form pattern. A list page presents a set of data on a UI that is optimized for browsing records, so that you can find and work with a specific record.

Usage

A list page presents a set of data on a user interface that is optimized so that you can browse records, find the right record, and then take an action upon that record. The list page lets the user search, filter, and sort the data. FactBoxes on the right side of the grid show related data for the active record. Actions that are relevant to the record are located on the ActionPane at the top of the page. The use of this pattern is now discouraged when there is a 1:1 correspondence between the List Page and Details page. Current guidance is to use this pattern only in other situations, such as when list pages have no backing details pages or have multiple backing details page (for example, when project quotations and sales quotations are shown together in the same List Page).

Pattern changes

Here are the main changes to this pattern since Microsoft Dynamics AX 2012:

- FormTemplate/InteractionClass is now optional when you build new pages.
- List Page and Details Master/Details Transaction are merged into a single form when there is a 1:1 correspondence between the List Page and Details Page.
  - Improves performance when the user moves between the list and details.
  - Allows for bulk editing in the initial list.
- The **Preview** pane has been eliminated.

Model

High-level structure

- Design
  - ActionPane (ActionPane)

Simple Details form pattern

Article • 01/03/2025

This article describes the Simple Details form pattern. This pattern is used when only a simple set of fields must be presented to the user.

Usage

The Simple Details pattern is used when only a simple set of fields must be presented to the user. Examples include the display of totals and customer balances. Typically, view mode is used for the Simple Details pattern. However, in cases where the form provides editable information, the edit mode should be synced to the parent form. Four patterns are described in this document:

- **Simple Details w/Toolbar and Fields –** This is the basic Simple Details pattern, in which several fields are displayed in the form. The fields can optionally appear inside Groups.
- **Simple Details w/Fast Tabs –** This is the Simple Details pattern that should be used when fields are organized into FastTabs.
- **Simple Details w/Standard Tabs –** This is the Simple Details pattern that should be used when fields are organized into traditional tabs.
- **Simple Details w/Panorama –** This is the Simple Details pattern that should be used when information is intended to be displayed in a panorama format.

Pattern changes

There are no planned changes for the use of this pattern in the current version of Microsoft Dynamics AX.

Model

Simple Details w/Toolbar and Fields – High-level structure

- Design
  - ActionPane (ActionPane)
  - Body (Group) – **Note:** A field subpattern is used.

Simple Details w/FastTabs – High-level structure

Simple List and Details form pattern

Article • 01/03/2025

This article provides information about the Simple List and Details form pattern. This pattern is used to maintain data for entities of medium complexity.

Usage

The Simple List and Details (SL+D) pattern is used to maintain data for entities of medium complexity. Entities of medium complexity are those entities that have six or more fields. The Simple List pattern should be used for simple entities that have fewer than six fields. There are some exceptions where entities that have up to 15 fields are still considered simple entities. The Simple List and Details pattern is prescribed when these conditions are met:

- The underlying data has more than six fields.
- There are between zero and five child data collections.

Three patterns are described in this document:

- **Simple List and Details – List Grid –** This is the basic SL+D pattern. This is the pattern that should be used by default.
- **Simple List and Details – Tabular Grid –** This is the SL+D pattern that should be used if the number of fields in the “simple list” part of the form is larger than expected (see the "Pattern changes" section later in this article).
- **Simple List and Details – Tree –** This is the SL+D pattern that should be used if the “simple list” part of the form is actually a tree.

Pattern changes

Here are the main changes to this pattern since Microsoft Dynamics AX 2012:

- The top ActionPane strip control has been converted to a standard ActionPane.
- **New**, **Delete**, and **Edit** buttons are provided by the framework.
- View mode is used by default.
- A Quick Filter control has been added above the “list” part of the form.
- Whenever possible, use the list-style grid for the “list” part of the form. A tabular grid is an acceptable alternative in some situations, such as when these conditions are met:

Simple List form pattern

Article • 01/03/2025

This article provides information about the Simple List form pattern. This pattern is used to maintain data for simple entities.

Usage

The Simple List pattern is used to maintain data for simple entities. Simple entities are entities that have six or fewer fields and no parent/child relationships. There are some exceptions where entities that have up to 15 fields are still considered simple entities.

Pattern changes

Here are the main changes to this pattern since Microsoft Dynamics AX 2012:

- The top ActionPane strip control has been converted to a standard ActionPane.
- **New**, **Delete**, and **Edit** buttons are provided by the framework.
- View mode is used by default.
- A Quick Filter has been added above the grid.
- When the form is used as a dependent form, the parent form record context is automatically shown above the form caption.
  - The page title group for dependent form usage was removed, because it will be provided by the framework.
- The pattern allows for multiple selections in the grid.

Model

High-level structure

- Design
  - ActionPane (ActionPane)
  - Custom Filter (Group)
    - Quick Filter (Quick Filter)
    - *OtherFilters ($Field) [0..N]*
  - TabularGrid (Grid)

Table of Contents form pattern

Article • 01/03/2025

This article provides information about the Table of Contents form pattern. This pattern should be used when two or more logically related forms are required for setup configuration.

Usage

The Table of Contents pattern should be used when two or more logically related forms are required for setup configuration. The vertical arrangement of tabs implies the order of completion. This form pattern is also used for collections of unrelated items, such as tab pages that have a different root entity per tab. This form pattern contains a collection of smaller content regions, each of which follows a container subpattern such as Toolbar and List, Nested Simple List and Details, or Fields and Field Groups.

Pattern changes

Here are the main changes to this pattern since Microsoft Dynamics AX 2012:

- The Content Body child container uses dynamic columns for a responsive layout.
- An optional secondary instruction has been added under the Title Group.

Model

High-level structure

- Design
  - Tab (Style=VerticalTabs)
    - TabPage *[repeats 1..N times]*
      - Title (Group)
        - MainInstruction (StaticText)
        - *SecondaryInstruction (StaticText) [Optional]*
      - Body (Group) | FastTabContent (Tab)

Core components

Task Double form pattern

Article • 01/03/2025

This article provides information about the Task Double form pattern. This pattern was previously used to present a parent and child entity in the same form.

Usage

This type of form has previously been used when you wanted to present parent/child entities in the same form. This isn't a recommended pattern for new forms. No new forms should be created that use this pattern. This pattern will provide structure and stability for legacy forms, and will also provide a migration path to more modern form patterns.

Pattern changes

Here are the main changes to this pattern since Microsoft Dynamics AX 2012:

- The form opens in view mode.
- The top ActionPane strip control has been converted to a standard ActionPane.
- The **Overview** label on the parent tab has been changed to **List**.
- The contents of the tab container use dynamic columns for a responsive layout.
- The label for the child tab's list should be **<x> list**, where **<x>** is replaced by an appropriate string, based on the entity. For example, if the child entity is usually called Charges, the label for the tab should be **Charges list**.
  - Exception: If the child entity is "lines" of some sort, the word "list" should not be added to the end.

Model

High-level structure

- Design
  - ActionPane (Action Pane)
  - *CustomFilter (Group) [Optional]*
  - ParentTab (Tab)
    - ParentList (TabPage) **– Note:** The Toolbar and List subpattern is used.

Task Single form pattern

Article • 01/03/2025

This article provides information about the Task Single form pattern. This pattern was previously used to present data that users would perceive as originating from a single data source that had multiple records.

Usage

This type of form was used when you wanted to present data that users will perceive as originating from a single data source with multiple records. This isn't a recommended pattern for new forms. No new forms should be created that use this pattern. This pattern will provide structure and stability for legacy forms, and will also provide a migration path to more modern form patterns.

Pattern changes

Here are the main changes to this pattern since Microsoft Dynamics AX 2012:

- The form opens in view mode.
- Commands have been moved to the standard ActionPane from a Toolbar (ActionPane strips).
- The **Overview** label on the first tab has been changed to **List**.
- The content of the tab container uses dynamic columns for a responsive layout.

Model

High-level structure

- Design
  - ActionPane (Action Pane)
  - *CustomFilter (Group) [Optional]*
  - Tab (Tab)
    - Overview (TabPage)
      - Grid (Grid)
      - *RowExtension (Group) [Optional]*

Wizard form pattern

Article • 01/03/2025

This article provides information about the Wizard form pattern. A wizard is a special form of user assistance that takes the user through a task by using an ordered series of tab pages.

Usage

A wizard is a special form of user assistance that takes the user through a task by using an ordered series of tab pages. Wizards are especially useful for complex or infrequent tasks that the user might have difficulty learning or doing, or for tedious, frequently performed tasks.

Pattern changes

Here are the main changes to this pattern since Microsoft Dynamics AX 2012:

- The secondary instruction for a wizard step was previously defined in the Help Text property of that step's Tab Page. This instruction will now be modeled on the Tab Page as a Static Text control.

Model

High-level structure

- Design (Style=Wizard; Caption=<wizard title>)
  - WizardContent (Tab)
    - WizardContentPage (TabPage) *[repeats 1..N times, can be named anything; Caption set to page title]*
      - MainInstruction (StaticText)
      - Body (Group)

Core components

1. Apply the Wizard pattern on **Form.Design**.
2. Address BP Warnings:

Workspace form pattern

Article • 01/03/2025

This article discusses workspace form patterns. Workspaces are the primary way that users navigate to tasks and specific pages. A workspace should be created for every significant business activity that is supported.

Usage

Workspaces are intended to be the primary way that users navigate to tasks and specific pages. A workspace should be created for every significant business "activity" that you want to support. An "activity" is less granular than a task but more granular than a legacy "area page." A workspace is intended to provide a one-page overview of the activity and to help users understand the current status, upcoming workload, and performance of the process or user. Users should be able to start the most typical tasks for the activity directly from the workspace. If possible, users should also be able to complete tasks directly in the workspace, based on the overview that is surfaced on the page. Currently, there are three workspace patterns:

- **Operational workspace –** This pattern is the standard pattern that is currently used for workspace development. Because of the set of components that it permits, this pattern has superior performance over the deprecated "workspace" pattern. For this reason, and to ensure visual and behavioral consistency with the other workspaces in the system, we recommend that you use this pattern. As of version 10.0.25, this pattern has been updated so that it no longer uses panorama controls and no longer scrolls horizontally. Forms that use this pattern now scroll vertically and use restyled FastTabs for the content sections.
    - **Operational workspace w/Tabs –** This variant of the **Operational workspace** pattern is available as of version 10.0.25. It uses standard tabs at the highest level to organize the workspace into different sections. Each standard tab can include a standard Operational workspace layout, a links section, or more custom content, such as embedded Power BI reports.

- (Deprecated) **Tabbed workspace –** This pattern was initially created to facilitate embedded Power BI reports and a more vertical orientation of workspaces. It's now deprecated and should be replaced by the **Operational workspace w/Tabs** pattern wherever possible.

- (Obsolete) **Workspace –** This pattern is mentioned only for the sake of completeness and can't be used after version 10.0.25. We recommend that any

Advanced selection form pattern

Article • 01/03/2025

This article provides information about the Advanced Selection form pattern. This Dialog form pattern lets users filter and select items from a large, wide list. Like the List Panel pattern, this pattern should be used when the primary user task is to select a set of items.

Usage

The Advanced Selection form pattern should be used when the primary user task is to select a set of items. This task is usually accomplished through a multiselect list. However, in many scenarios, users must select items that aren't contiguous and, at the same time, must see the set of items that they're selecting. This pattern resembles the List panel pattern, in that the user selects items in one list and adds them to another. However, this pattern allows for custom filters and a “wide” list on top, and uses most of the screen "real estate" of the page (typically, it's a Large dialog). Use this pattern when a user must be able to filter and select in a large, wide list.

Related patterns

- List Panel subpattern
- Dialog form pattern

UX guidelines

The verification checklist shows the steps for manually verifying that the form complies with UX guidelines. This checklist doesn’t include any guidelines that are enforced automatically through the development environment. Open the form in the browser, and walk through these steps.

- **Standard form guidelines:**
  - Standard form guidelines are consolidated into the General form guidelines document.
- **Advanced selection guidelines:**
  - By default, the Quick filter should use the name or description column.
  - The list can display up to 15 columns.
  - The main instruction should instruct users what they need to do.
  - When there's no data, the grid shouldn't automatically add a new record.

Dialog form pattern

Article • 01/03/2025

This article provides information about the Dialog form pattern. A dialog box represents an action or activity that users can explicitly commit or cancel. It's used when a user initiates a specific task or process, and the system requires user input about how or whether to proceed.

Usage

A dialog box represents an action or activity that users can explicitly commit or cancel. It's used when a user initiates a specific task or process, and the system requires user input about how or whether to proceed. Dialogs are modal and require that users interact with the controls in the dialog before they can return to the parent page. Dialogs also can have multiple sizes. Selection of a dialog size is subjective, and will vary, depending on the form elements that you've modeled on the dialog. The sizes are as follows:

- **Small** – This size is a one-column-wide dialog. If your dialog contains a relatively small amount of content (all simple fields, and no wide tables or other wide elements), you can probably use this size.
- **Medium –** This size is a two-column-wide dialog. If your dialog contains more content than can comfortably fit within a small dialog, but a full-width dialog isn't required, you should use this size.
- **Large –** This size is a three-column-wide dialog. If your dialog contains more content than can comfortably fit within a medium dialog, but a full-width dialog isn't required, you should use this size.
- **Full** – A large dialog is nearly the full width of the browser viewport. Its size varies, depending on the viewport width, and it will always be the largest dialog size option. Use this size if your dialog has a lot of wide elements, or if it requires an unusually large amount of horizontal space.

For more detail about the various dialog sizes, see the table in the appendix of this article, under "Selecting the correct dialog size." We strongly recommend that you review that table. Five patterns are described in this document:

- **Dialog –** This is the basic dialog pattern. Use this dialog if you don't have a reason to use one of the other Dialog patterns.
- **Dialog w/tabs –** This is a more specific version of the Dialog pattern. It incorporates a Tab control in the dialog. You can also optionally provide a header

Drop Dialog form pattern

Article • 01/03/2025

This article provides information about the Drop Dialog form pattern. This pattern is used to initiate actions when the number of fields is seven or fewer.

Usage

The Drop Dialog pattern is used to initiate actions when the number of fields is seven or fewer. Drop dialogs are quick and easy for users to use, and are more lightweight than a full dialog that is presented as a slider. Drop dialogs should feel as lightweight to use as a menu. Two patterns are described in this document:

- **Drop dialog –** This is the basic Drop dialog pattern. If your Drop dialog is editable, this is the correct pattern to use.
- **Drop dialog (read only) –** This Drop dialog pattern is for informational forms that aren't editable. This variation doesn't have an **OK** button.

Pattern changes

Here are the main changes to this pattern since Microsoft Dynamics AX 2012:

- Manual handling of error messages is no longer required.

Model

Drop dialog (basic) – High-level structure

- Design
  - *SecondaryInstruction (StaticText) [optional]*
  - DialogContent (Group)
  - DialogCommitContainer (ButtonGroup)
    - OKButton ($Button)

Drop dialog (read only) – High-level structure

- Design

Lookup form pattern

Article • 01/23/2025

This article provides information about the Lookup form pattern. Custom lookup forms should be used when a standard framework-provided lookup would not provide the correct data, or when advanced visualization of the data is required.

Usage

Custom lookup forms should be used when a standard framework-provided lookup (which is typically generated by using the AutoLookup field group that is defined on the table definition), would not provide the correct data, or when advanced visualization of the data is required. Three patterns are described in this document:

- **Lookup basic –** This is the basic Lookup pattern that has just one list or tree, and also optional custom filters and actions.
- **Lookup w/tabs –** This Lookup pattern is used when more than one view of the lookup can be made available to the user. Tab captions aren't shown. Instead, the tab is selected through a combo box.
- **Lookup w/preview –** This more advanced Lookup pattern enables a preview of the current record in the lookup grid.

Wireframe

Lookup basic

FactBox form patterns

Article • 01/03/2025

This article provides information about the FactBox form patterns. FactBoxes are used to provide related information for a record.

Usage

In general, FactBoxes are used to provide related information for a record. They help guarantee that the user doesn't have to open additional forms to get important information, such as totals, balances, overdue orders, and email addresses. The Factbox Grid pattern should be used when there's a child collection (potential for multiple rows) of related information. Two patterns are described in this document:

- **Form Part FactBox Grid –** This FactBox pattern is used when there's a child collection (potential for multiple rows) of related information.
- **Form Part FactBox Card –** This FactBox pattern is used when there's just a set of related fields that must be shown.

Pattern changes

Here are the main changes to this pattern since Microsoft Dynamics AX 2012:

- A group was added around the optional button to make it easier to position the button.

Model

Form Part FactBox Grid – High-level structure

- Design
  - Grid
  - *GridDefaultAction (Button) [Optional]*
  - *ButtonGroup (ButtonGroup) [Optional]*
    - Button

Form Part FactBox Card – High-level structure

Custom Filter Group subpattern

Article • 01/03/2025

Usage

This subpattern is used to show a small collection of input controls (no more than five) that apply a custom filter to a grid or form section. Fields in the Custom Filter Group should be limited to the following field types, which have constrained inputs and can be applied to the query:

- StringEdits with Lookups
- Date fields
- ReferenceGroup
- Comboboxes
- Checkboxes
- Quick Filter

Two patterns are described in this document. The only difference between these patterns is whether the Quick Filter control is mandatory or optional:

- **Custom Filters** – In this subpattern, the QuickFilter control is optional.
- **Custom and Quick Filters** – In this subpattern, the QuickFilter control is mandatory.

Model

Custom Filters – High-level structure

- CustomFilter (Group)
  - *QuickFilter (QuickFilter) [Optional]*
  - *FieldGroups (Group) [0..N]*
    - Fields ($Field) [1..N]
  - *Fields ($Fields) [0..N]*

Custom and Quick Filters – High-level structure

- CustomFilter (Group)

Dimension Entry Control subpattern

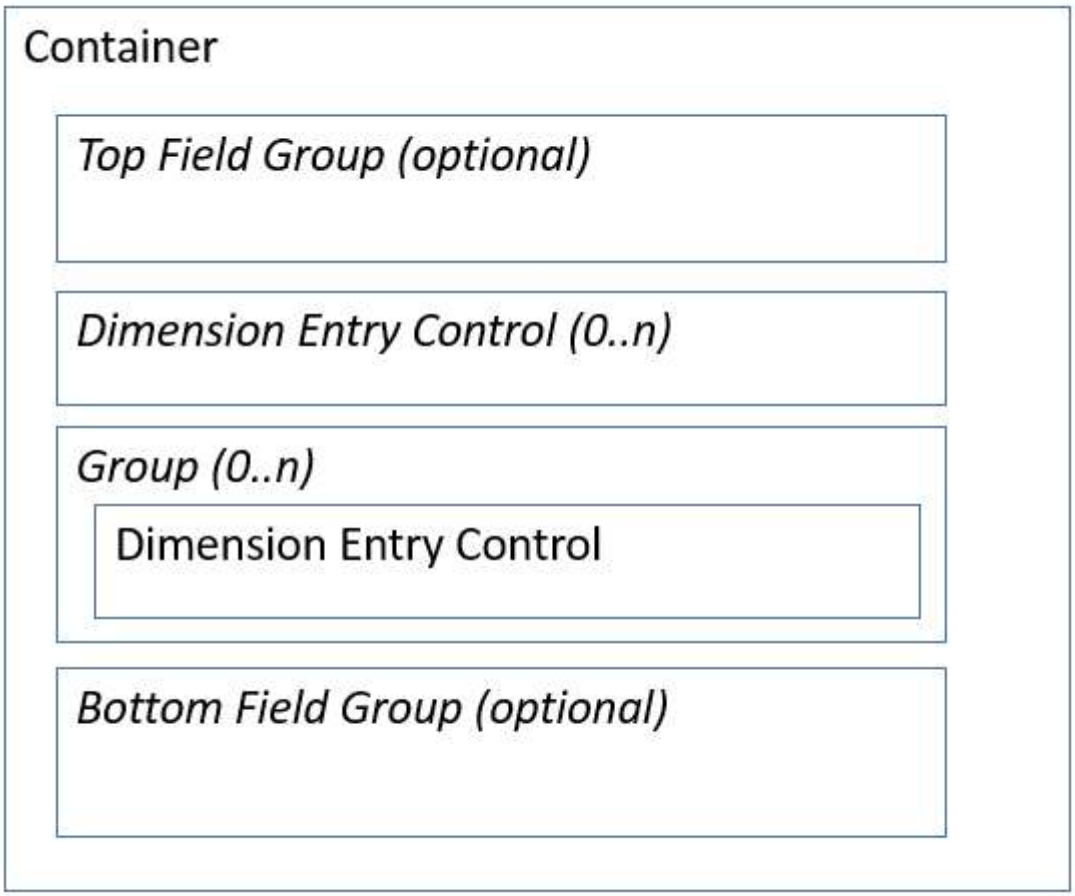
Article • 06/19/2024

This article provides information about the Dimension Entry Control subpattern. This subpattern is used when you have a group or tab page that uses the Dimension Entry control (DEC).

Usage

The Dimension Entry Control pattern is used when you have a group or tab page that uses the Dimension Entry control (DEC).

Wireframe



Model

High-level structure

TabPage | Group *TopFieldGroup (Group) [Optional]* – **Note:** A field subpattern is used.

Dimension Expression Builder subpattern

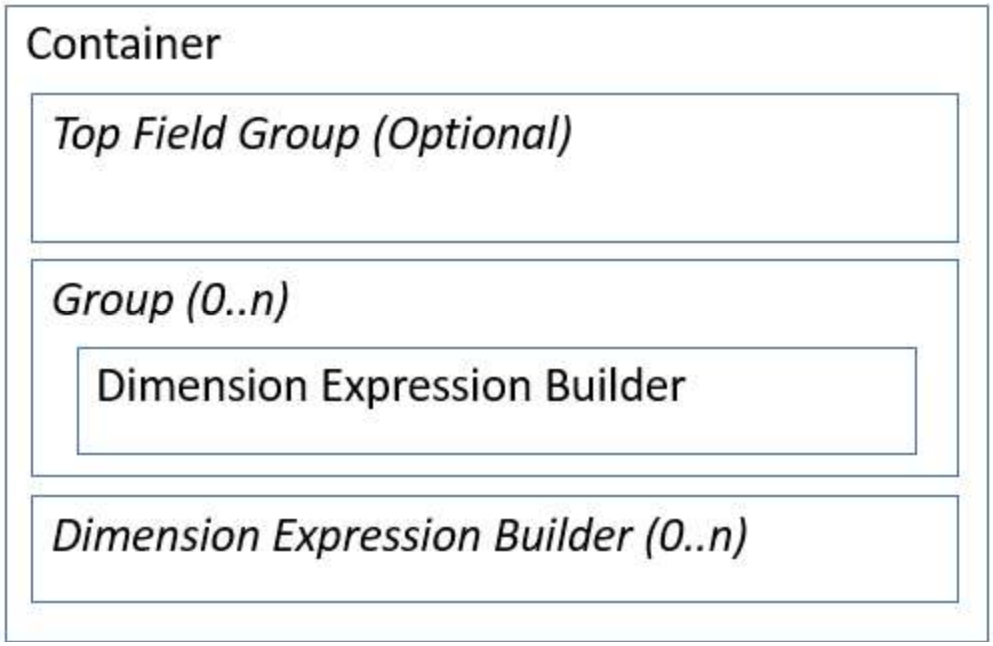
Article • 06/19/2024

This article describes the Dimension Expression Builder subpattern, which is applied to container controls that use the Dimension Expression Builder control.

Usage

The Dimension Expression Builder pattern is used when you have a group or tab page that uses the Dimension Expression Builder control.

Wireframe



Model

High-level structure

TabPage | Group

TopFieldGroup (Group) [Optional] – **Note:** A field subpattern is used.

DEBGroup (Group) [0..N]

Fields and Field Groups subpattern

Article • 01/03/2025

This article provides information about the Field and Field Groups form subpattern. This is the most common data entry subpattern. It uses a dynamic number of columns to present multiple fields or groups of fields.

Usage

Field and Field Groups is the most common data entry subpattern and uses a dynamic number of columns to present multiple fields or groups of fields. This subpattern is not used with controls that have dynamic height or width (for example Grid, Tree, RadioButton, ListBox, or ListView), or controls that have larger height or width (for example, Chart). The group controls within this pattern can be used either to group fields under a label or to bind to a table field group.

Typical contents

- Groups or Fields as immediate children of the FastTab
- Groups containing Fields
- Can contain other subpatterns:
  - Horizontal fields and button group

Pattern changes

Here are the main changes to this pattern since Microsoft Dynamics AX 2012:

- Removed explicit columns and the use of groups to force fields into two or three (or more) columns.
- Changed from fixed columns to dynamic columns.

Model

High-level structure

- [Container] (Columns=Fill)
  - *FieldGroups (Group) [0..N]*
    - Fields ($Field) [1..N]

Filters and Toolbar subpatterns

Article • 01/03/2025

This article provides information about the Filters and Toolbar subpatterns. These workspace-specific subpatterns have been developed to show filters and/or actions inside panorama sections that host lists and charts.

Usage

The Filters and Toolbar subpatterns are workspace-specific subpatterns that have been developed to show filters and/or actions inside panorama sections that host lists and charts. Fields in the filtering parts of these subpatterns should be limited to the following field types. All these field types have constrained inputs and can be applied to the query.

- StringEdits with Lookups
- Date fields
- ReferenceGroup
- Comboboxes
- Checkboxes
- Quick Filter

Two subpatterns are described in this article:

- **Filters and Toolbar - Inline** – In this subpattern, any defined actions appear on the same line as the filter fields.
- **Filters and Toolbar - Stacked** – In this subpattern, any defined actions appear on a separate line below the filter fields.

Model

Filters and Toolbar - Inline: High-level structure

- Group (ArrangeMethod=HorizontalLeft)
    - *FilterGroup (Group) [Optional]*
        - *QuickFilter (QuickFilter) [Optional]*
        - *FilterFIelds ($Field) [0..N]*
    - *Toolbar (ActionPane) [Optional]*

Fill Text subpattern

Article • 01/03/2025

This article provides information about the Fill Text subpattern. This subpattern is used when a single String or StaticText control must stretch to the full width of the container, so that users have more space to enter information.

Usage

Fill Text is used when you need a single String or StaticText control to stretch to the full width of the container. This subpattern is typically used for multi-line string controls that require more space for users to enter information.

Model

High-level structure

[Container]

String | StaticText

Core components

- Apply the Fill Text subpattern to the container control.

Related container patterns

- Fields and Field Groups

UX guidelines

None

Resources

Typically used by patterns

Horizontal Fields and Buttons Group subpattern

Article • 01/03/2025

This article provides information about the Horizontal Fields and Buttons Group form subpattern. This subpattern is used when actions must be defined for an individual field on a form.

Usage

This subpattern is used when actions must be defined for an individual field on a form. The buttons are laid out just to the right of the field to visually associate the actions with the field. The buttons should display only an icon (no text). Actions that are associated with a section or an entire form should be placed in a Toolbar or ActionPane above that section or form.

Typical contents

- 1–2 fields
- 1–3 buttons

Pattern changes

Here are the main changes to this pattern since Microsoft Dynamics AX 2012:

- The layout of fields and buttons will use a single column, where **ArrangeMethod**=**HorizontalLeft**.

Model

High-level structure

- Group (ArrangeMethod=HorizontalLeft)
  - Field
  - *Field (optional)*
  - Buttons (1–3 buttons)

Image Preview subpattern

Article • 01/03/2025

This article provides information about the Image Preview form subpattern. This subpattern can be used for most images that appear within a form container, especially within a FastTab or Group.

Usage

Image Preview can be used for most images that appear within a form container, especially within a FastTab or Group. This subpattern can be used in conjunction with the FieldsAndFieldGroup and FillText subpatterns to combine images and any associated fields. This subpattern isn't used for tiles or buttons, or for field status images.

Typical contents

- Toolbar (ActionPane where **Style**=**Strip**)
- Image
- Can contain subpatterns:
  - Fields and Field Groups
  - Fill text

Pattern changes

Here are the main changes to this pattern since Microsoft Dynamics AX 2012:

- Fields are to the right of the image, if there are any fields.
- An ActionPane above the image can be used for associated actions (for example, Upload and Select).

Model

Image only – High-level structure

- [Container] (Columns = Fixed – 1)
  - *Toolbar (ActionPane) [Optional]*
  - Image

List Panel subpattern

Article • 01/03/2025

This article provides information about the List Panel form subpattern. Application teams use this subpattern to manage two lists that move data between each other.

Usage

List Panel is the subpattern that application teams use to manage two lists that move data between each other. This pattern is meant to represent a modeled version of the **SysListPanel** class (programmatic) approach of managing two lists that move data between each other. The List Panel subpattern can be applied on the following controls:

- TabPage control
- Group control

Pattern changes

Here are the main changes to this pattern since Microsoft Dynamics AX 2012:

- List (Grid/ListView) and Tree controls are supported.
- The right panel is the selected section.
- The left panel is the available section.
- Six buttons are available as actions:
  - Add
  - Remove
  - Add All (Optional)
  - Remove All (Optional)
  - Move Up (Optional)
  - Move Down (Optional)

Model

High-level structure

- [Container]
  - *CustomFilterGroup (Group) [Optional]*
  - ListPanelGroup (Group)

Nested Simple List and Details subpattern

Article • 01/03/2025

This article provides information about the Nested Simple List and Details (NSL+D) subpattern. This subpattern is used to display information about a secondary or child entity when that child entity is presented within another form type.

Usage

This article describes a variant of the Simple List and Details (SL+D) pattern that is named the Nested Simple List and Details (NSL+D) subpattern. Whereas the SL&D form pattern is used to display information about the primary entity on the form, the NSL+D subpattern is used to display information about a secondary or child entity when that child entity is presented within another form type. The amount of information that is related to the child entity should be too much for a grid (10 or more fields) but not enough for the child entity to deserve its own form. The NSL+D subpattern has a few differences from the SL+D form pattern:

- You may not nest an NSL+D subpattern within another NSL+D subpattern.
- The NSL+D subpattern uses a Toolbar for contextual actions.
- The details portion of the NSL+D subpattern is simpler than the SL+D pattern. The NSL+D subpattern uses only groups, whereas the SL+D pattern organizes content into FastTabs.

Pattern changes

Here are the main changes to this pattern since Microsoft Dynamics AX 2012:

- This pattern is new. Any pattern changes to the SL+D pattern can be found in the Simple List and Details pattern document.

Model

High-level structure

- <Container>

Section Chart form pattern

Article • 01/03/2025

This article provides information about the Section Chart form pattern. This pattern is primarily used in conjunction with the Operational Workspace pattern, and specifically on forms that contain a chart control.

Usage

The Section Chart form pattern is intended to be used primarily in conjunction with the Operational Workspace pattern. Specifically, the chart section or summary section contains Form Part Controls that point to forms that contain charts. These referenced forms are intended to use the Section Chart pattern.

Pattern changes for finance and operations apps

This pattern didn't exist for Microsoft Dynamics AX 2012.

Model

High-level structure

- Form Design
  - *HeaderGroup (Group) [Optional]* – This uses one of the Filters and Toolbar subpatterns.
  - Chart

Core components

Apply the Section Chart pattern to the appropriate form/container.

Related container patterns

- Workspace
- Section stacked chart

Section Power BI subpattern

Article • 01/03/2025

This article provides information about the Section PowerBI subpattern. This subpattern is used as part of the Operational Workspace pattern, specifically for the panorama section that contains a PowerBI control.

Usage

The Section PowerBI subpattern is used as part of the Operational Workspace pattern, specifically for the panorama section that contains the PowerBI control.

Pattern changes for Microsoft Dynamics AX

This pattern didn't exist for Microsoft Dynamics AX 2012.

Model

High-level structure

TabPage PowerBI (PowerBI)

Core components

Apply Section PowerBI to the appropriate tab page in the workspace.

Related container patterns

- Operational workspace

UX guidelines

None

Appendix

Section Related Links subpattern

Article • 01/03/2025

This article provides information about the Section Related Links subpattern. This subpattern is used as part of the Operational Workspace pattern, specifically for the last panorama section that contains a set of links to other forms.

Usage

The Section Related Links subpattern is used as part of the Operational Workspace pattern, specifically for the last panorama section that contains a set of links to other forms.

Pattern changes for Microsoft Dynamics AX

This pattern didn't exist for Microsoft Dynamics AX 2012.

Model

High-level structure

- TabPage
  - *LinkButton ($Button) [0..N]*
  - *ButtonGroup (Group) [0..N]*
- LinkButton ($Button) [1..N]

Core components

Apply Section Related Links to the appropriate tab page in the Operational Workspace.

Related container patterns

- Operational Workspace

UX guidelines

Section Stacked Chart subpattern

Article • 01/03/2025

This article provides information about the Section Stacked Chart subpattern. This subpattern is used as part of the Operational Workspace pattern when a panorama section contains one or two charts.

Usage

The Section Stacked Chart subpattern is used as part of the Operational Workspace pattern, specifically for a panorama section that contains one or two charts.

Pattern changes for Microsoft Dynamics AX

This pattern didn't exist for Microsoft Dynamics AX 2012.

Model

High-level structure

- TabPage
  - *ChartPart (FormPart) [0..N]*

Each Form Part points to a form that contains a single chart. Each of these forms should use the Section Chart form pattern.

Core components

Apply Section Stacked Chart to the appropriate tab page in the workspace.

Related container patterns

- Operational workspace
- Section Chart

UX guidelines

Section Tabbed List subpattern

Article • 01/03/2025

This article provides information about the Section Tabbed List subpattern. This subpattern is used as part of the Operational Workspace pattern, specifically for a panorama section that contains a set of vertical tabs, each of which contains a filtered list of data.

Usage

The Section Tabbed List subpattern is used as part of the Operational Workspace pattern, specifically for a panorama section that contains a set of vertical tabs, each of which contains a filtered list of data.

Pattern changes for Microsoft Dynamics AX

This pattern didn't exist for Microsoft Dynamics AX 2012.

Model

High-level structure

- TabPage
- TabbedList (Tab) (Style=VerticalTabs)
    - *TabbedListPage (TabPage) [0..N]*
        - TargetForm (FormPart)

Each Form Part points to a form that contains the content for the section. Each of these forms should use one of the Form Part Section List form patterns.

Core components

Apply Section Tabbed List to the appropriate tab page in the workspace.

Related container patterns

- Operational Workspace

Section Tiles subpattern

Article • 01/03/2025

This article provides information about the Section Tiles subpattern. This subpattern is used as part of the Operational Workspace pattern, specifically for the first panorama section (the Summary section) that contains a set of tiles, charts, and singleton cards.

Usage

The Section Tiles subpattern is used as part of the Operational Workspace pattern, specifically for the first panorama section (the **Summary** section) that contains a set of tiles, charts, and singleton cards.

Pattern changes for Microsoft Dynamics AX

This pattern didn't exist for Microsoft Dynamics AX 2012.

Model

High-level structure

- TabPage
  - *TileButton (TileButton) [0..N]*
  - *TargetForm (FormPart) [0..N]*

The Form Parts are used to embed Charts or singleton Cards into the **Summary** section of the workspace. Each form that represents a Chart should use the Section Chart form pattern.

Core components

Apply Section Tiles to the first tab page in the Operational Workspace.

Related container patterns

- Operational Workspace
- Section Chart

Tabular Fields subpattern

Article • 01/03/2025

This article provides information about the Tabular Fields subpattern. This subpattern is used to show information efficiently in a tabular format.

Usage

This subpattern is used to show information efficiently in a tabular format. The fields are arranged in a table that contains rows and columns, and that optionally contains column headers, row labels, a caption, and a footer. The Tabular Fields subpattern can be applied on the following controls:

- TabPage control
- Group control

Pattern changes

In previous releases of Microsoft Dynamics AX, there was no formally accepted way to model this pattern. Therefore, this pattern was modeled in many inconsistent ways that must be modified to match the current pattern. The most common way to model this pattern was to use groups for columns. However, groups are now used for the rows. The primary reason for this change was to better match the HTML/CSS constructs, and it also helps keep the tab sequence and semantics of a table.

Model

High-level structure

- TabularFields (Group*)
    - CaptionGroup (Group)
        - *TableCaption (StaticText) [Optional]*
    - TableHeaderRow (Group)
        - *Column0Label (StaticText) [Optional]* – **Note:** This static text fills col0, row0 with a blank.
        - ColumnLabels (StaticText) [1..N] – **Note:** These are the normal column headers.

Toolbar and Fields subpattern

Article • 01/03/2025

This article provides information about the Toolbar and Fields subpattern. This container pattern is used to show actions above a subpattern of data fields. The toolbar should contain fewer than 10 actions.

Usage

This container pattern is used to show actions above a subpattern of data fields. The toolbar should contain fewer than 10 actions.

Model

High-level structure

- [Container]
  - Toolbar (ActionPane, Style=Strip)
  - ContentGroup (Group) – **Note:** A fields subpattern is used.

Core components

- Apply the ToolbarFields subpattern to the container control.
- Address BP Warnings:
  - No additional BP checks are required beyond the AX6.3 BP checks that were carried forward.

Related patterns

- Toolbar and List

Commonly used subpatterns

- Fields and Field Groups
- Tabular Fields
- Dimension Expression Builder

Toolbar and List subpattern

Article • 01/03/2025

This article provides information about the Toolbar and List form subpattern. This subpattern is used to show child collections for the parent entity as either a tabular grid or a tree.

Usage

This subpattern is used to show child collections for the parent entity as either a tabular grid or a tree. The toolbar contains fewer than 10 actions. If a grid is used, it contains fewer than 10 fields. This article describes two patterns:

- **Toolbar and list –** This is the basic version of the pattern and should be used by default.
- **Toolbar and list (double) –** This variant includes two lists and an optional toolbar above each list.

Model

Toolbar and list – High-level structure

- [Container]
  - *Toolbar (ActionPane, Style=Strip) [Optional]*
  - *CustomFilterGroup (Group) [Optional]*
  - Grid | Tree | ListView | Table
  - *Footer (Group) [Optional]*

Toolbar and list (double) – High-level structure

- [Container]
  - *Toolbar1 (ActionPane, Style=Strip) [Optional]*
  - *CustomFilterGroup1 (Group) [Optional]*
  - Grid | Tree | ListView | Table
  - *Toolbar2 (ActionPane, Style=Strip) [Optional]*
  - *CustomFilterGroup2 (Group) [Optional]*
  - Grid | Tree | ListView | Table
  - *Footer (Group) [Optional]*

Workspace Page Filter Group subpattern

Article • 01/03/2025

This article provides information about the Workspace Page Filter Group subpattern. This subpattern is used as part of the Operational Workspace pattern when a workspace must expose a single workspace-wide filter on the form.

Usage

The Workspace Page Filter Group subpattern is used as part of the Operational Workspace pattern, specifically when a workspace must expose a single workspace-wide filter on the form.

Pattern changes for Microsoft Dynamics AX

This pattern didn't exist in Microsoft Dynamics AX 2012.

Model

High-level structure

- Group
  - Field ($Field)

Core components

Apply Workspace Page Filter Group to the appropriate group in an (operational) workspace.

Related container patterns

- Custom Filter Group
- Operational workspace

UX guidelines

The verification checklist shows the steps for manually verifying that the form complies with UX guidelines. This checklist doesn't include any guidelines that will be enforced

Build extensible controls

Article • 01/23/2025

This article describes how to create new application controls that have a property sheet in Visual Studio and have server-side business logic.

Prerequisites

For this tutorial, you must access the environment by using Remote Desktop, and you must be provisioned as an administrator on the instance. For more information, see Deploy and access development environments.

Overview

The Control Extensibility Framework lets you create new application controls. You can use the same tools that Microsoft uses to build controls that are already present in the program, such as the chart control. Three important artifacts are involved in the process of developing an extensible control:

- **The X++ build class –** The build class lets a developer define the properties that appear in the Microsoft Visual Studio property sheet for the control. The developer can also define the modeling behavior for the control when it's used in the form designer. The build class is consumed by the run-time class to initialize the state of the control based on the value of properties in the property sheet.
- **The X++ run-time class –** The run-time class lets a developer define server-side business logic and data access patterns for an extensible control. Two concepts that are specific to building extensible controls are the *properties* and *commands* that the X++ class defines. Each property and command that is defined is serialized into a JavaScript view model at run time, and can be consumed by the client parts of the extensible control (the HTML and JavaScript). These properties and commands are the main channels for moving information between the server-side and client-side parts of the control.
- **The control HTML and JavaScript –** Each control uses HTML, JavaScript, and CSS files to define control visualization and client-side interaction patterns. By using the Microsoft Dynamics HTML binding syntax together with jQuery, a developer can consume the properties and commands that are defined in X++ to design powerful data-driven UI.

All three artifacts of extensible control development are explained in more detail in the following sections.

Keyboard shortcuts for extensible controls

Article • 12/31/2024

Keyboard shortcuts are an important consideration when you create any extensible control. This article provides information that will help you choose keyboard shortcuts for your extensible controls. It also outlines the recommended method for implementing keyboard shortcuts for extensible controls.

Overview

For accessibility, it's essential that keyboard-only users be able to use controls. Therefore, keyboard shortcuts are an important consideration when you create any extensible control. This article provides information that will help you choose key combinations to use as keyboard shortcuts. It highlights the shortcuts that are currently used by finance and operations apps and supported browsers, shortcuts that are planned for implementation, and shortcuts that one or more browsers don't allow to be overridden. This article also outlines the recommended way to implement keyboard shortcuts for extensible controls.

Choosing a key combination

When you're trying to choose a key combination to use as a keyboard shortcut, it's important that you're aware of other existing shortcuts. In this way, you help guarantee that your shortcut won't overlap an existing shortcut. If you try to collide with an existing shortcut, one of the following outcomes might occur:

- The new keyboard shortcut might not work, because a browser doesn't allow that key combination to be overridden, or a framework-provided shortcut takes precedence over the new shortcut.
- The new keyboard shortcut might remove expected keyboard functionality, because users expect specific key combinations to perform specific functions in a browser. Alternatively, you might override framework-provided shortcuts or other control shortcuts, so that keyboard-only users can't use them.

Because of these potential issues, we recommend that you adhere to this guidance when you choose a key combination:

Extensible control programming reference

Article • 12/31/2024

This article provides reference content for extensible control programming.

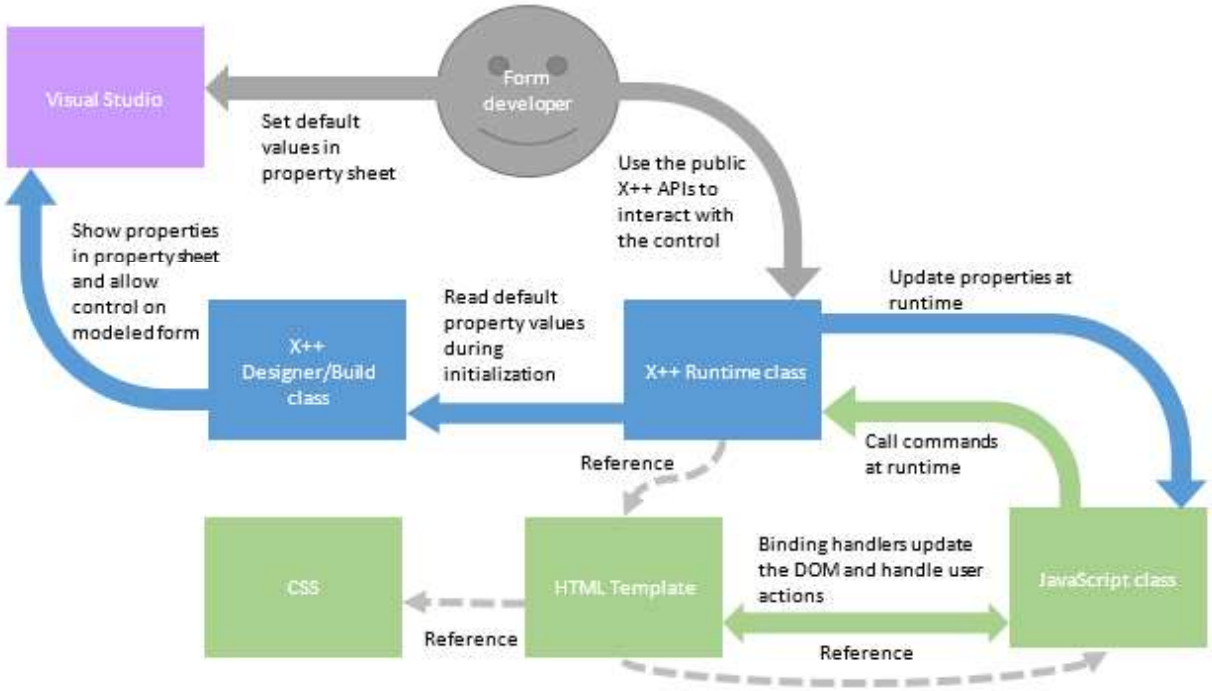
This document describes the API, HTML, and JavaScript support for creating extensible controls.

Examples

This document contains small code snippets that show how to use each API that is documented. More complete examples of finished controls that leverage many of these APIs can be found on GitHub. Extensible Control Examples on GitHub

Control block diagram

This high-level diagram illustrates the key components of an extensible control and how they interact with each other. Your extensible control solution will contain two X++ classes that implement your control. The runtime class implements the runtime data, presentation, and behavior of your control. The build class defines how your control is displayed in Form Designer, Property Window, and Application Explorer.



X++

Control extensibility

Article • 08/12/2022

This article describes the architecture that lets developers extend the user interface and also define new user interface patterns.

You can extend the existing application user interface (UI) and can also define entirely new UI patterns to create compelling new user experiences. By using modern tools such as HTML5, CSS3, and jQuery, developers can define customized visualizations of business data and drastically enhance the program's interaction patterns.

Server-side architecture

The Control Extensibility Framework takes advantage of the existing and familiar X++ language for developing server-side data access and business logic. There are no artificial restrictions on the code that developers write to build extensible controls. Instead, developers can declaratively define the modeling experience and the run-time behavior through a set of X++ class and method attributes. A developer defines one class for the design-time behavior (the X++ Build Class) and one class for the run-time behavior (the X++ RunTime class).

- In the X++ Build Class, attributes enable the definition of design-time behaviors such as custom properties in the property sheet, the addition of child controls, and extra modeling components.
- In the X++ Runtime Class, attributes are used to define the run-time properties and commands that the extensible control will access from the client. The X++ Runtime Class consumes the X++ Build Class to initialize the run-time properties, based on the values and data bindings that are specified in the property sheet.

Client-side architecture

The client-side behavior for the control is defined by using HTML and JavaScript. In the context of a Model-View-ViewModel architecture, the HTML for the control is the View, and the JavaScript is the viewmodel. The Control Extensibility Framework provides an HTML-based binding syntax that enables elements in the HTML View to be bound to data fields and properties in the JavaScript viewmodel. In addition, the framework enables visualization behavior to be defined based on conditional expressions or logical evaluations that can react to changes in viewmodel properties or business data. The JavaScript viewmodel is automatically generated at run time, based on the properties and commands that are defined in the X++ runtime for the control. This automatically

Create localizable labels

Article • 08/12/2022

This article explains how to create localizable labels for client components and HTML/JavaScript controls.

This article details the process for creating **localizable** labels for client components and HTML/JavaScript controls. This process uses the existing localization tools and process for labels to bring localization support to client components and HTML/JavaScript controls. The following process relies on the label resource controller that can serialize label files into their JavaScript equivalents so that the labels can be used by the client components and HTML/JavaScript controls. The label resource controller is deployed automatically. It is an MVC service that is located at the /Resources/Labels endpoint.

1. Create a label file

Use the developer tools to create a new label file for your control's area, or use an existing label file for your control's area. A control's area is determined by the owning team.

- For extensible controls, your goal should be to create one label file for each HTM resource file. If multiple HTM resources share the same set of labels, only one label file should be required for the set of HTM resource files.
- For client controls and components, in general, controls that share a lot of the same functionality (for example, the Input controls: StringEdit, ComboBox, CheckBox, and so on) should also share the same label file.

Don't use a label file that also contains labels that are only used in X++. The whole label file is serialized when it's loaded by the client, so be sure to keep the labels that aren't required by the client components/controls in a separate label file.

2. Add label strings to the label file

Use the developer tools to add label strings to the label file. **Example for extensible controls:**

- **Label file name:** ClockControl
- **Label ID:** Seconds
- **Label string:** seconds

Extensible control layout guidelines

Article • 12/31/2024

This article provides guidelines that you should follow when you specify the layout and sizing of extensible controls.

Dos and don'ts for achieving the desired layout

- Don't use the layout classes on your control directly. (For example, **layout-container** and **layout-horizontal** are classes that you might see on controls in the DOM.) Instead, use the layout binding handlers to apply these classes. Internet Explorer uses a different layout framework, and to add some inline styles to elements, this framework requires the extra binding information that the handlers provide. Therefore, make sure that the classes are **not** hard-coded into the controls.
- Don't use absolute positioning (**position: absolute** and **top/bottom/left/right** positions) for elements that are children of a container that uses the layout binding handler. Absolute positioning of these elements prevents the CSS classes that are applied from laying things out correctly.
- Be careful about using **width: 100%** and **height: 100%**. These settings might not always work when they are combined with our layout CSS classes. If you want filling behavior, it's a good idea to use the **$dyn.layout.Size.available** option of the sizing binding handler instead.

Layout binding handlers

Layout

Used to apply **ArrangeMethod** and **Columns** attributes to containers Options: **arrangeMethod**, **Columns**, **Children**

ArrangeMethod

- $dyn.layout.ArrangeMethod.vertical
- $dyn.layout.ArrangeMethod.horizontalLeft
- $dyn.layout.ArrangeMethod.horizontalWrap
- $dyn.layout.ArrangeMethod.horizontalRight

Control the text that Task Recorder generates for a control

Article • 12/31/2024

This article describes how Task recorder determines what instruction label to generate for controls. It then explains how you can make sure that these labels are meaningful for the user.

Every control must have useful and meaningful instruction labels, so that the task guide, Microsoft Word document, and Help content meet Content Publishing standards for readability. We must first define two terms:

- **Control label** – The value that comes from the label property on the control.
- **Instruction label** – The label that a control instructs Task recorder to use when it's describing how to use that control (for example, “Click OK” or “In the First name field, enter ‘John’”).

When a control logs an event to Task recorder, three methods can be used to determine the instruction label that is shown to the user:

- As part of logging the Task recorder event, the control might specify an exact instruction label ID to use. As a best practice, here is how label IDs should be named: [client control type name]_[property or command name] For manual specification of an instruction label ID, see the code example later in this article (**OptionalInstructionLabelIDOverride**).
- If the control doesn't explicitly specify an instruction label ID, Task recorder looks in the SysTaskRecorderLabel file to try to find an existing instruction label ID that fits the following naming syntax: [client control type name]_[property or command name] If an instruction label ID of this type is found, Task recorder uses it.
- If a label can't be determined by using the preceding methods, Task recorder falls back to a more general-purpose instruction label. The general purpose instruction labels are in the SysTaskRecorder label file. There is one general-purpose instruction label for commands and another for properties.
  - Here is the general purpose instruction label for commands:
    - **Label ID:** CommandUserAction
    - **Label string:** %2 the %1
  - Here is the general purpose instruction label for properties:
    - **Label ID:** PropertySetValue
    - **Label string:** In the %1 field, enter %2

Build operational workspaces

Article • 01/03/2025

This article provides detailed information about workspaces and the patterns and subpatterns that are used to build operational workspaces.

A workspace is defined as...

- Part of the primary navigation mechanism.
- A form that supports a business activity (a logical group of tasks that make up the work of a target persona).
- A way to provide an initial overview and to increase productivity in the activity by allowing simple tasks to be completed directly in the workspace.

Workspaces have the following goals...

- Enable the user to understand the current state of the activity to support informed decisions.
- Let users navigate to deeper pages by selecting data, which avoids round-trips to pages with no information.
- Let users perform light tasks in the workspaces to avoid round-trips to deeper pages.
- Complete an activity without leaving the workspace.
- Reduce the need for navigation.
- Provide visual impact.
- Be constructed using prescriptive patterns and best practices that lead to minimal COGS and fast response times.

To accomplish these goals, the operation workspace pattern was developed.

Examples

An example of a workspace is the **Reservation management** workspace in **Fleet management**. You can get to it by accessing the menu item **FMClerkWorkspace**. The workspace, shown above, has the following items:

- **Summary -** Contains tiles and a chart.
- **Rentals -** Contains a vertical tab control having three pages - the first is selected, and you can see the corresponding content on the rightmost side.

Tile and list caching for workspaces

Article • 12/31/2024

It's important that workspaces perform well, and that they be responsive (that is, the data that appears in a workspace is refreshed as expected and kept up to date). This article discusses framework support for caching data that is used for tiles and lists.

Introduction

Workspaces are intended to be the hub of activity for most users. They display a wealth of information that is collected from various sources. Therefore, you must make sure that workspaces perform well and are responsive (that is, the data shown in the workspace is refreshed as expected and/or kept up to date). Caching can help guarantee great workspace performance when data comes from poorly performing queries, and is especially useful when multiple users require access to the same set of data at the same time. For example, if a grid on a form uses a complex (that is, low-performing) query, you can cache the results so that they are available for subsequent loads of the form. This article discusses framework support for caching data that is used for tiles and lists. In terms of a responsive workspace, when users navigate to a workspace (or return to it), they expect that the data in the workspace will be relatively up to date. For example, if a user takes an action that changes the data for a list or tile in a workspace, the workspace should reflect that change immediately after the action is performed (if the action was taken directly from the workspace) or when the user returns to the workspace form (if the action was taken on a different form). This article describes techniques for making sure that this type of data refresh occurs (both metadata and code) for both tiles and lists.

Count tiles

Automatic data caching

The framework automatically sets up data caching behind any defined count tile. Therefore, no extra code is required in this case. A **Refresh Frequency** metadata property on Tiles determines how often the count on the tile is automatically updated. The following table describes the options and guidance for this property.

Expand table

Task recorder resources

Article • 12/31/2024

> **Important**
>
> Customers using Human Resources, the functionality noted in this article is currently available in both the stand-alone Dynamics 365 Human Resources and the merged Finance infrastructure. Navigation might be different than noted while we make updates. If you need to find a specific page, you can use Search.

This article describes how to use Task recorder to record business processes.

Overview

Task recorder

Task recorder for finance and operations apps is a utility that lets users record business processes for several different use cases. Here are some examples:

- Step-by-step guided tours of a specific business process in the application itself
- Documentation of a business process as a Microsoft Word document that can optionally include screenshots
- Regression tests for a business process
- Automatic playback of a business process in the application

Task recorder for finance and operations apps boasts high responsiveness, a flexible extensibility application programming interface (API), and seamless integration with consumers of business process recordings. Task recorder is also integrated with the Business process modeler (BPM) tool in Microsoft Dynamics Lifecycle Services (LCS), so that users can continue to organize their recordings. However, users can no longer produce business process diagrams from recordings.

Task recorder can automatically generate application regression tests from business process recordings and play back previously recorded processes. These features also include test-specific gestures that let users take full advantage of Task recorder.

Architecture

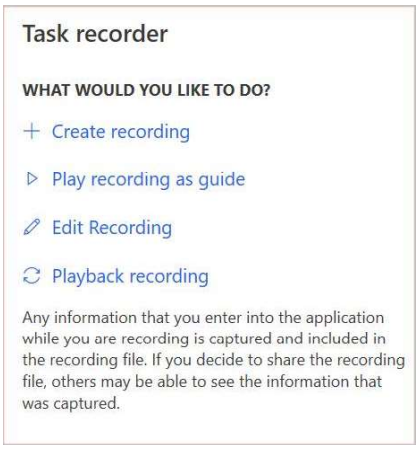
Task Recorder quick reference

Article • 12/31/2024

This article provides a quick reference sheet that explains what each button in the Task recorder menus does.

Main menu

Expand table



Create recording

Choose this option to begin creating a new recording.

Play recording as guide

Choose this option to see what your recording looks like when viewed as a Help article or played as a Task guide.

Edit recording

Choose this option if you need to change the recording's name, description, or the text that is displayed in the steps.

Playback recording

Choose this option if you need to add or remove steps. You can also use this mode to automatically play a recording.

Open and save options

Expand table

Create documentation or training with Task Recorder

Article • 01/24/2025

> **Important**
>
> Customers using Human Resources, the functionality noted in this article is currently available in both the stand-alone Dynamics 365 Human Resources and the merged Finance infrastructure. Navigation might be different than noted while we make updates. If you need to find a specific page, you can use Search.

This article explains what Task recorder and task guides are, how to create task recordings, and how to customize Microsoft task guides and include them in your Help.

> **Important**
>
> You can record your own task guides for Dynamics 365 Human Resources, but you won't be able to save them to a Business Process Modeler (BPM) library or open them from the Help pane. You can save them locally or to a network location, and then open and replay them using Task recorder.

Learn about Task recorder

Task recorder is a tool that you can use to record actions that you take in the product user interface (UI). When you use Task recorder, all of the events that you perform in the UI that are executed against the server—including adding values, changing settings, removing data—are captured. The steps that you record are collectively called a *task recording*. Task recordings can be used in many ways:

- **Task recordings can be played as task guides -** Task guides are an integral piece of the Help experience. A task guide is a controlled, guided, interactive experience through the steps of a business process. The user is instructed to complete each step by way of a pop-up prompt, which animates across the UI and point to the UI element that the user should interact with. The pop-up prompt also provides information about how to interact with the element, such as **Click here** or **In this field, enter a value.** A task guide runs against the user's current data set and the data that is entered is saved in the user's environment.