

AF DatePicker

Date, DateTime and Range Picker for UGUI

User Documentation

AF DatePicker is a configurable date and time picker for Unity UGUI. It is designed to work like a normal Unity component: add the prefab, choose the behavior you need, optionally assign a Theme and/or Layout, and style the surrounding UI to suit your project.

Built on familiar Unity UI: AF DatePicker is made from standard Unity UGUI and TextMeshPro components. There is no proprietary UI system or separate workflow to learn. If you already know UGUI, the prefab, hierarchy, buttons, text, input fields and layout components should feel immediately familiar.

Full C# source code is included. You can inspect, modify and extend AF DatePicker as needed for your own project.

1. Quick Start

1. Add the AFDDatePicker prefab to a Canvas.
2. Alternatively, open the included demo scene and enter Play Mode to try a preconfigured example immediately.
3. In Settings, choose the Trigger Type and picker Mode.
4. Configure date restrictions, initial values, display format and optional buttons.
5. In Design, assign a Theme and Layout preset. You can edit the assigned assets directly in the picker Inspector.
6. Position and size the trigger object as you would any other UGUI element.
7. Use the picker events or public API when your application needs to react to or control the selected value.

2. Picker Modes

Mode	Purpose
Date	Select one calendar date.
Date Range	Select a start date and an end date.
DateTime	Select one date together with a time. Time selection can use a specific time or a time range.
DateTime Range	Select a start date/time and an end date/time.

3. Trigger Types

The trigger is the UI element the user clicks to open or close the picker. AF DatePicker comes with three trigger types:

- Text - a TextMeshPro text element.
- Input Field - a TMP Input Field used as a read-only picker trigger.
- Button - a standard UGUI Button.

The selected value is formatted by AF DatePicker and shown on the active trigger. If no value is selected, the trigger returns to its original prefab-authored text.

4. Settings

Date Selection

Allow Past, Allow Today and Allow Future determine which dates can be selected. Optional Minimum Date and Maximum Date limits can further restrict the available calendar dates.

Initial Start Date controls the initial value shown by the picker. Range modes also provide Initial End Date. Initial values can be left empty, use the current date where applicable, or use a custom value.

Display Format

Built-in date formats include:

- dd/MM/yyyy
- MM/dd/yyyy
- yyyy-MM-dd
- dd.MM.yyyy
- dd-MM-yyyy
- Custom format

Time can be displayed in 12-hour or 24-hour format. DateTime modes can display Date And Time or Time And Date. Range modes also allow a custom Range Separator.

Calendar

First Day Of Week can follow the locale or be explicitly set. Weekday Display controls how weekday headings appear. Letter uses one character, Abbreviated uses three characters, and Custom uses the full text authored in the weekday TMP labels.

Behaviour

For single-value Date and DateTime modes, Close After Selection can automatically close the calendar once a selection is complete.

Options Bar

The bottom options bar can independently show or hide Today, Now and Clear where those actions are applicable. Today selects the local current date. Now uses the local current date and time. Clear removes the current selection.

5. Time Selection

Time values can be entered directly into the hour and minute fields or selected using the popup time scrollers. Hours and minutes are always stored as valid DateTime values internally.

- 24-hour mode uses hours 00-23.
- 12-hour mode displays AM/PM labels in the hour scroller.
- Minutes are available from 00-59.
- Manual input is validated and reformatted to two digits.
- In 12-hour mode, manually changing the numeric hour preserves the current AM/PM period.
- Clicking elsewhere in the calendar closes the time popup.

6. Design: Theme and Layout

Design is intentionally separated from behaviour. Theme controls appearance; Layout controls geometry. Functional picker settings remain on the individual AFDatePicker instance.

Theme

A Theme is a ScriptableObject asset shared by any picker that references it. Themes control the supplied picker's visual appearance, including header and body typography, text colours, backgrounds, selection/range/hover states, selector appearance, options bar colours and an optional rounded sprite.

The package includes several ready-to-use themes:

- Default
- Clean Light
- Slate Dark
- Arcana Blue
- Unicorn Poop
- Dark Green

Important: editing an assigned Theme edits the Theme asset itself. Every AF DatePicker using that Theme will therefore use the changed values. Duplicate an existing Theme first when you want to create a separate variation.

Layout

A Layout is also a shared ScriptableObject asset. It controls the geometry of the supplied picker without changing its colours or behaviour.

Area	Layout controls
Picker	Overall picker size.
Navigation	Navigation height, spacing and arrow width.
Weekdays	Weekday header height and spacing.
Calendar	Spacing between day cells.
Time Bars	Time bar height and spacing.
Time Selector	Popup size, spacing, item height and item width.
Month Selector	Month item spacing.
Year Selector	Year item height, item width and spacing.

As with Themes, changing a Layout changes every picker that references that Layout. Duplicate a Layout before editing when you need a separate variation.

Placement

Placement settings are stored on the picker instance rather than in the Layout asset. They control Calendar Placement, Calendar Spacing, Month Selector Width and Month Selector Spacing.

Creating Your Own Theme or Layout

Create a Theme or Layout asset using Unity's Create menu, or duplicate one of the supplied presets. Assign the asset in Design > Presets, then edit it directly in the AF DatePicker Inspector. Because these are normal Unity assets, their values are saved with the project.

7. Calendar and Selectors

The main calendar displays a six-week, 7 x 6 day grid. Navigation arrows move between months. The Month and Year controls open dedicated selectors for faster navigation.

- Month Selector - displays the twelve authored month labels in a 3 x 4 grid.
- Year Selector - provides a scrollable year list and visually highlights the selected year.
- Dates outside the current month may be shown with reduced emphasis while still participating in selection where valid.
- Disabled dates use the Theme's disabled appearance and cannot be selected.

8. Opening and Closing

Clicking the active trigger toggles the calendar. Clicking outside the picker closes it. AF DatePicker creates its outside-click blocker under the containing Canvas at runtime so the blocker can cover the surrounding scene UI.

9. Events

The Advanced tab exposes the picker events:

- OnOpened - invoked when the picker opens.
- OnClosed - invoked when the picker closes.
- OnValueChanged - invoked when the selected value changes.

10. Retrieving the Selected Value and Public API

The selected value is available directly from the AFDatPicker component as nullable DateTime properties. Use SelectedDate for Date and DateTime modes. Use RangeStart and RangeEnd for Date Range and DateTime Range modes.

For a single Date or DateTime value:

```
AFDatePicker picker = GetComponent<AFDatePicker>();

if (picker.SelectedDate.HasValue)
{
    DateTime value = picker.SelectedDate.Value;
    Debug.Log(value);
}
```

For Date Range or DateTime Range:

```
AFDatePicker picker = GetComponent<AFDatePicker>();

if (picker.RangeStart.HasValue && picker.RangeEnd.HasValue)
{
    DateTime start = picker.RangeStart.Value;
    DateTime end = picker.RangeEnd.Value;

    Debug.Log($"{start} - {end}");
}
```

HasSelection is a convenience property for checking whether SelectedDate contains a value. HasRange is true only when both RangeStart and RangeEnd are available. During range selection, RangeStart can therefore contain a value while HasRange is still false.

OnValueChanged does not pass the date as an event argument. When it fires, read SelectedDate, RangeStart and RangeEnd from the picker. This keeps one event usable for all picker modes.

Example event handler:

```
public AFDatPicker Picker;

public void HandleDatePickerChanged()
{
    if (Picker.Settings.Mode == AFDatPickerMode.Date ||
        Picker.Settings.Mode == AFDatPickerMode.DateTime)
    {
        if (Picker.SelectedDate.HasValue)
```

```

        Debug.Log(Picker.SelectedDate.Value);
    }
    else
    {
        if (Picker.RangeStart.HasValue)
            Debug.Log($"Start: {Picker.RangeStart.Value}");

        if (Picker.RangeEnd.HasValue)
            Debug.Log($"End: {Picker.RangeEnd.Value}");
    }
}

```

Available public properties:

- **IsOpen** - true while the calendar is open.
- **Settings** - access to the picker settings.
- **SelectedDate** - selected Date/DateTime value, or null when none is selected.
- **RangeStart** - start of the selected range, or null.
- **RangeEnd** - end of the selected range, or null until the range is complete.
- **HasSelection** - true when SelectedDate contains a value.
- **HasRange** - true when both RangeStart and RangeEnd contain values.

Available public methods:

- **OpenCalendar()** - opens the picker.
- **CloseCalendar()** - closes the picker.
- **ToggleCalendar()** - toggles the current open state.
- **SetDate(DateTime value)** - sets the single selected date/date-time and invokes OnValueChanged.
- **SetRange(DateTime start, DateTime end)** - sets the range and invokes OnValueChanged. If end is earlier than start, AF DatePicker automatically swaps them.
- **ClearValue(bool notify = true)** - clears the current value. Pass false when you do not want OnValueChanged invoked.
- **CloseTimeScrollerPanel()** - closes the time-selection popup if it is open.

11. Advanced References

The Advanced > References section contains the internal object references used by the supplied prefab. Normal users should not need to change these. They are exposed primarily so the prefab can be inspected, repaired or deliberately restructured by advanced users.

If you create a heavily modified picker hierarchy, preserve the required references and expected UGUI/TMP components.

Because the complete source is supplied, advanced users can also extend the picker behaviour directly. The supplied implementation intentionally stays within normal Unity C#, UGUI and TMP patterns.

12. Styling the Trigger

AF DatePicker does not force a visual style onto the surrounding trigger. The Text, TMP Input Field or Button can be styled and positioned using normal Unity UGUI/TMP controls. This keeps the picker compatible with the rest of your project's UI rather than imposing a separate trigger design.

13. Locale and Time

Locale affects the configured first day of the week when First Day Of Week is set to Locale. Weekday and month text is based on the text supplied in the prefab rather than forcing operating-system month/day names, which keeps those labels available to normal Unity localization workflows.

Today and Now use the device's local date/time. AF DatePicker does not provide manual GMT/UTC offset handling.

14. Recommended Workflow

1. Start with the supplied AFDatePicker prefab.
2. Choose the required Mode and behaviour.
3. Choose the closest supplied Theme and Layout.
4. If customization is needed, duplicate the Theme/Layout assets and edit the copies.
5. Style and position the trigger to match your own UI.
6. Connect OnValueChanged or use the public API as required.
7. Avoid modifying internal prefab references unless you intentionally change the supplied hierarchy.

15. Demo Scene

A demo scene is included as a ready-to-run reference. Open the scene and enter Play Mode to try the picker without additional setup. It also provides a known-good example of the supplied prefab and its references, and can be used as a quick reference if you customize your own instance.

16. Notes for Package Documentation

Import AF DatePicker into your Unity project and add the supplied AFDatePicker prefab to a Canvas. AF DatePicker uses standard Unity UGUI and TextMeshPro components and does not require a separate UI framework or proprietary runtime system. If TextMeshPro Essentials have not yet been imported into the project, Unity may prompt you to import them.

The included demo scene provides a ready-to-run example of the picker. Open it and enter Play Mode to verify the component and use it as a reference when configuring your own instances.

Supplied visual presets are located under Assets/Arcana Futura/AF DatePicker/Presets. Theme presets are in the Themes folder. Custom Layout assets can be created or stored in the sibling Layouts folder.

To create your own variation, duplicate one of the supplied Theme assets or create a new Theme/Layout asset, assign it under Design > Presets, and edit it normally in the Inspector. Remember that Theme and Layout assets are shared: editing an asset affects every picker that references it.

AF DatePicker includes its full C# source code. The supplied prefab and scripts may therefore be inspected, modified and extended to suit the needs of your project using normal Unity, UGUI and TextMeshPro workflows.

For supported Unity versions, refer to the compatibility information on the Asset Store listing for the version of AF DatePicker you are using.

17. Contact

Have a question, need help, or found a bug? Please use our online contact form and we'll get back to you as soon as possible.

Website: <https://arcanafutura.eu/>

Support: <https://arcanafutura.eu/asset-support>

18. License and Disclaimer

AF DatePicker is provided under the applicable Unity Asset Store licence. It may be used and modified as part of your own commercial or non-commercial projects, subject to the terms of that licence.

You may modify the supplied source code for use in your projects. Modification does not grant the right to redistribute or resell AF DatePicker itself. The asset, its source code, or modified versions of the asset may not be redistributed, resold, sublicensed or otherwise made available as a standalone asset or development tool.

AF DatePicker is provided “**as is**”, without warranties of any kind. Arcana Futura is not liable for losses or damages resulting from its use.

AF DatePicker is provided under the Unity Asset Store End User License Agreement (EULA).

<https://unity.com/legal/as-terms>