ChatGPT

Oxygen 6 RC 1 Multilingual, RTL, and CSS Logical Properties Compatibility Review

Scope, sources, and evaluation approach

This report evaluates **Oxygen 6 RC 1** (release-candidate build) specifically for two adoption-critical requirements in multilingual, Arabic/Right-to-Left (RTL) website delivery:

1) **Multilingual plugin compatibility**, with emphasis on WPML and any documented behavior for RTL languages (e.g., Arabic).
2) **Ability to use CSS Logical Properties** (e.g., `margin-inline-start`, `inset-inline-end`, `inline-size`) instead of physical properties (e.g., `margin-left`, `right`, `width`), including what Oxygen generates by default, whether logical properties can be enabled, and what this implies for RTL and responsive work.

The analysis prioritizes: - Official Oxygen 6 release notes and documentation. [1]
- WPML official compatibility and errata documentation (as the most direct ground truth for WPML support status). [2]
- Community and user reports from Oxygen's public issue tracker (GitHub) and other reputable sources where they document reproducible issues and workarounds. [3]
- Vendor documentation from multilingual plugin providers (notably TranslatePress and Weglot) where they claim/describe compatibility mechanisms with Oxygen. [4]

Because Oxygen 6 RC 1 is a pre-release, findings are framed as **"supported/unsupported/uncertain as of published RC1 documentation and public reports"**, not as a guarantee for the final/stable release. The RC 1 release announcement itself explicitly positions it as a pre-release build to be tested on staging. [5]

Oxygen 6 RC 1 context relevant to multilingual and RTL risk

Oxygen 6 RC 1 (published January 15, 2026; updated January 21, 2026) is described by its authors as a late-stage stability/security update, with a small changelog (stacking indicator improvements, a builder-access security fix, and updates to Breakdance Elements for Oxygen). [5]

Two platform-level statements in Oxygen's own materials matter for multilingual/RTL due diligence:

- Oxygen 6 is a **complete rewrite**, "from the ground up," with a new UI, native CSS variables, and other workflow changes. [6]
- Oxygen and **entity["organization","Breakdance","wordpress builder by soflyy"]** now share the same "core builder engine," and Oxygen claims that "nearly every plugin compatibility issue" addressed in Breakdance is now addressed in Oxygen (as of February 26, 2025). [7]

These statements suggest improved compatibility in general, but they do **not** constitute an official compatibility claim regarding WPML (or RTL) for Oxygen 6 RC 1. The Oxygen 6 documentation section on multilingual plugins also stops short of any guarantee and frames integrations as third-party. [8]

Multilingual plugin compatibility and RTL readiness

What Oxygen 6 RC 1 documentation does and does not claim

In Oxygen 6 documentation, the “Multilingual Plugins” integration page states:

- Among multilingual plugins used by Oxygen users, **entity["organization","TranslatePress","wordpress translation plugin"]** is “the most popular.”
- **entity["organization","Polylang","wordpress multilingual plugin"]** has been reported as successful by some users.
- The Oxygen team recommends **entity["company","Weglot","website translation service"]**.

[8]

Notably, the Oxygen 6 multilingual page **does not** claim official integration with WPML, nor does it document RTL-specific behavior (e.g., Arabic layout flipping, RTL editor UI). Instead, it emphasizes a support-policy boundary: Oxygen support covers Oxygen itself, not issues caused by third-party multilingual plugins, and Oxygen positions itself as willing to collaborate *if* multilingual plugin developers pursue integration. [8]

WPML’s official position: not compatible, with specific known failures

WPML’s official compatibility listing for Oxygen states plainly that “Oxygen is not currently compatible with WPML” and warns users they “may experience issues.” [9]

WPML further documents concrete, recurring failures in Oxygen-built content translation workflows, including: - **Completed translations not loading on the front-end** for Oxygen-built posts/pages when using WPML’s Translation Editor (WPML’s workaround is manual translation; WPML states Oxygen is not currently compatible). [10]
- **Widget/link translation issues** (e.g., inability to translate Link fields in Oxygen widgets via WPML’s Translation Editor), again with a manual-translation workaround and an explicit statement that WPML is jointly investigating with the Oxygen team. [11]

WPML support threads reinforce this: WPML staff repeatedly state Oxygen is not fully compatible and advise manual translation; they reference known issues like non-Latin characters being encoded in translations and suggest workarounds (with caution that they are not official). [12]

Implication for Oxygen 6 RC 1 evaluation: even though WPML’s public compatibility pages reference “version 4.0” in the listing context, WPML’s broader, maintained errata and support answers continue to treat Oxygen as **not WPML-compatible** in a way that impacts production multilingual delivery. There is no public WPML statement (in the sources reviewed) that asserts Oxygen 6 RC 1 resolves these incompatibilities. [2]

Oxygen Classic multilingual caveats as risk indicators for Oxygen 6

Although this report targets Oxygen 6 RC 1, Oxygen Classic documentation contains multilingual caveats that align with WPML's known issues, and therefore function as historically consistent risk patterns:

- "Template translation doesn't always work for logged-out users," and a recommended workaround is to keep one template, duplicate content inside it, translate in-place, and use conditions to show/hide by language. [13]
- Translation plugins can prevent saving changes to reusable parts, and changes in one language may break another language on some WPML sites. [13]

These match community reports in Oxygen's public issue tracker, including: - Mixed language behavior when logged out and translation coverage gaps (e.g., rich text fields not included), when using WPML and translated template post types. [14]
- Reusable part saving failures caused by WPML language-directory redirects interfering with Oxygen save requests. [15]

Because Oxygen 6 is a rewrite, these Classic-era issues are not proof of identical behavior in 6 RC 1. However, the absence of an explicit Oxygen 6 + WPML integration statement, combined with *continued WPML errata*, means these remain relevant signals when assessing risk. [16]

Compatibility comparison table

The following table consolidates the most defensible compatibility signals (official documentation first, then community reports). "RTL readiness" here refers to the likelihood that the plugin + Oxygen combination can produce correct RTL output **without extensive per-page CSS surgery**, based on whether directionality is part of the translation mechanism and whether known RTL friction exists.

Multilingual solution	Public compatibility signal for Oxygen	How language switching/ translation is handled (as documented)	RTL-related notes (Arabic emphasis)
WPML	WPML: “not currently compatible,” multiple errata; manual translation recommended as workaround. [2]	WPML Translation Editor is the problematic path for Oxygen-built pages; WPML repeatedly advises manual translation for Oxygen pages. [17]	WPML states it supports RTL languages and layouts, but only applies RTL formatting to themes that support RTL; if the theme doesn’t, WPML won’t apply RTL formatting. [18] This increases risk in “theme-less builder” workflows where RTL depends on custom CSS.
entity["organization","TranslatePress","wordpress translation plugin"]	TranslatePress documentation explicitly states it is compatible with Oxygen and provides a concrete Oxygen-side method for show/hide by language. [19]	TranslatePress can drive language decisions via locale; doc shows using Oxygen conditions with `get_locale` to show/hide elements per language. [19]	RTL output still depends on CSS/layout choices, but TranslatePress’s approach (front-end translation with locale awareness) avoids some Oxygen↔WPML Translation Editor integration points that are documented as failing. [20]

Multilingual solution	Public compatibility signal for Oxygen	How language switching/ translation is handled (as documented)	RTL-related notes (Arabic emphasis)
entity["company","Weglot","website translation service"]	Oxygen recommends Weglot (Oxygen 6 docs). [8] Weglot lists Oxygen Builder as compatible. [21]	Weglot positions compatibility at the platform/tool level (including Oxygen), implying it can translate output without deep builder-specific field extraction. [21]	RTL feasibility is strongly tied to how the site's CSS handles direction; Oxygen's open RTL issues (alignment defaults, physical properties) still matter for Arabic UX. [22]
entity["organization","Polylang","wordpress multilingual plugin"]	Oxygen 6 docs: "reported as a successful option by some users." [8] Oxygen Classic docs: community success reported; caveats exist. [13]	Polylang typically works by creating translated content entities and switching language by URL/locale; Oxygen Classic docs lean on URL-string conditions for show/hide approaches. [13]	At least one long-standing Oxygen issue reports Polylang-related editor incompatibility in a multilingual context (e.g., Gutenberg-related slug variation). [23] RTL still depends on CSS and Oxygen's own RTL friction. [22]

Bottom line on WPML: Based on WPML's maintained documentation and errata, there is no evidence (in the reviewed public sources) that Oxygen 6 RC 1 provides the integration hooks WPML needs for reliable use of WPML's Translation Editor with Oxygen-built pages/templates. Treat WPML as **not supported** for a production-grade multilingual workflow unless you accept manual translation and/or bespoke workarounds. [24]

RTL behavior in Oxygen workflows: documented friction points and what they imply for Arabic

RTL readiness has two distinct surfaces:

- **Builder/editor RTL**: whether the authoring UI behaves correctly when the admin language or environment is RTL.
- **Front-end RTL**: whether rendered sites can flip direction and maintain correct spacing/alignment/layout affordances.

Builder/editor RTL: unresolved UX issues indicated by long-running reports

Oxygen's public issue tracker contains explicit reports that editing RTL websites is "less complete" because UI elements appear in unexpected places, and that switching WordPress/admin language does not fix it; the reporter's workaround is to force the builder UI direction to LTR via CSS in a builder toolbar stylesheet. [25]

A broader, older feature request asks for "Better RTL Support" so sites can be built for Hebrew/Arabic clients, implying this limitation has been material for years. [26]

While these reports long predate Oxygen 6 RC 1, the Oxygen 6 documentation reviewed does not contain a countervailing statement such as "Oxygen 6 editor is fully RTL-aware." Therefore, for Arabic-first teams, **assume additional authoring friction** until proven otherwise in hands-on testing. [27]

Front-end RTL: where CSS defaults and "physical" styling become costly

Two specific RTL-related issue patterns are particularly relevant to Arabic sites:

- **Default alignment assumptions.** A report states that inserting a new DIV results in `text-align: left;` always being added, which is "redundant" in LTR and "incorrect on RTL languages," creating recurring alignment problems for Hebrew sites. [28]
- **Need for direction-aware spacing.** A feature request asks for the ability to "easily change padding, margin, etc. from right & left to (inline-start & inline-end)" specifically to reduce RTL effort, which directly connects Oxygen's styling UX to the CSS Logical Properties topic. [29]

These are meaningful because Arabic layouts tend to break not only from `direction: rtl`, but from physical assumptions embedded in styling primitives (left padding, left-positioned icons, "float right," etc.). If a builder routinely bakes physical directionality into generated CSS, RTL becomes a "second stylesheet" problem—costly and error-prone.

Third-party RTL patching exists but is likely obsolete for Oxygen 6 era

A WordPress plugin titled "RTL Support for Oxygen Builder" exists in the WordPress plugin directory, tagged for Arabic/Hebrew/RTL. However, WordPress.org flags that it has not been tested with the latest three major WordPress releases and may be unmaintained; it was last updated "6 years ago" and tested only up to WordPress 5.4.18. [30]

For a 2026 evaluation of Oxygen 6 RC 1 (and modern WordPress versions), this plugin is best viewed as historical evidence that RTL pain existed—not as a reliable mitigation for current builds. [31]

Diagram of the practical RTL problem Oxygen teams face

A simplified representation of why "physical" CSS becomes a multilingual RTL cost center:

```
LTR page expectation:                RTL page expectation:
[START → ..... END]                  [END ..... ← START]

Physical CSS strategy:
  margin-left, padding-right, right:0
  => must be audited and swapped per direction.

Logical CSS strategy:
  margin-inline-start, padding-inline-end, inset-inline-end:0
  => adapts automatically when direction changes.
```

The open request to support `inline-start`/`inline-end` within Oxygen's UI indicates that Oxygen users are explicitly seeking the second strategy for RTL efficiency. [29]

CSS Logical Properties in Oxygen 6 RC 1: native output vs developer-controlled output

What Oxygen 6 says about CSS generation

Oxygen 6 materials repeatedly describe a CSS-forward, developer-oriented model:

- Oxygen 6 Beta/RC communications describe a "properties panel that maps directly to CSS" and improvements to responsive design clarity, plus performance optimizations such as minified CSS output. [32]
- In Oxygen 6 Beta 2 materials, Oxygen explains that "design properties like padding, colors, or layout settings generate dynamic CSS," while content controls populate HTML structure. [33]

This supports a key inference: **Oxygen 6 is not a "theme translation system."** It is a visual authoring tool that generates HTML/CSS, and multilingual/RTL outcomes depend heavily on how that CSS is structured and emitted.

Does Oxygen 6 "generate logical properties" by default?

In the examined Oxygen 6 documentation and release notes, there is **no explicit statement** that Oxygen generates CSS logical properties (e.g., `margin-inline-start`) instead of physical properties. [34]

Conversely, the Oxygen public issue tracker includes an open request specifically asking for UI support to switch spacing controls from left/right to inline-start/inline-end, indicating that users **do not currently have** that ergonomic pathway. [29]

Additionally, the open RTL issue about DIVs defaulting to `text-align: left` indicates that at least some default output is direction-physical rather than direction-logical (where `text-align: start` would typically be direction-aware). [28]

Conclusion: Based on public sources reviewed, Oxygen 6 RC 1 does not provide evidence of **native** logical-property generation across its design controls, and community requests suggest the opposite. The most defensible position is: **logical properties are not a first-class, builder-level output mode** today. [35]

Can Oxygen 6 still *use* logical properties?

Yes—via developer-controlled CSS.

Oxygen includes a "CSS Code" element whose documentation states that CSS written there is added to the stylesheet for the specific document (post/template/archive stylesheet). [36]

This means a developer can implement logical-property strategies manually (globally via shared classes/selectors, or locally for templates), even if the visual spacing controls emit physical properties.

Example pattern (illustrative):

```
/* Direction-aware spacing */
.card {
  margin-inline: 1rem;
  padding-inline-start: 1.25rem;
}

/* Direction-aware positioning */
.badge {
  position: absolute;
  inset-inline-end: 0.75rem;
  inset-block-start: 0.75rem;
}

/* Direction-aware alignment */
.card__title {
  text-align: start;
}
```

(Implementation via Oxygen's CSS Code element or class-based CSS fits within Oxygen's described CSS workflow.) [37]

Important nuance: Manual CSS can conflict with auto-generated physical styles unless you enforce conventions (e.g., avoid using the visual "left/right" padding controls on elements that should be direction-neutral, and centralize spacing via a class framework). The existence of the "padding-inline instead of padding-right" feature request signals that this conflict is a real workflow cost for RTL teams. [29]

CSS Logical Properties evaluation matrix

CSS capability needed for RTL-friendly builds	Oxygen 6 RC 1 evidence of native support	Practical reality for teams	RTL implication
Use `*-inline-start/end` for margin/padding in visual controls	No public documentation of this; open request asks for exactly this feature. [38]	Likely requires writing CSS manually and/or adopting a class framework where logical properties live in code, not UI sliders. [37]	Without conventions, RTL becomes "duplicate styling" work; with conventions, logical properties can significantly reduce RTL deltas. [29]
Use direction-aware alignment defaults (`text-align: start`)	Open issue describes defaults that are not RTL-friendly (`text-align: left`). [28]	Requires manual correction patterns (e.g., global reset or authoring rules).	Prevents repeated "fix alignment" loops when building Arabic pages. [28]
Use logical sizing (`inline-size`, `block-size`)	No explicit Oxygen statement found; but Oxygen can inject arbitrary CSS via CSS Code. [39]	Feasible through custom CSS; unclear whether UI emits logical sizing values.	Useful when combined with direction/writing-mode changes; most impactful for advanced internationalization work. [29]
Maintain responsive styling across breakpoints while staying RTL-safe	Oxygen highlights responsive design improvements and CSS generation model. [40]	Breakpoints can coexist with logical properties; the work is governance (don't author physical overrides unless truly needed).	Good governance reduces per-language CSS branches; weak governance increases divergence. [41]

Analytical conclusions for multilingual Arabic/RTL adoption

WPML support status for Oxygen 6 RC 1

From a technical evaluation standpoint, the most reliable and current publicly documented position is still: **WPML does not consider Oxygen compatible**, and WPML documents specific failures that impact real-world multilingual delivery (translations not loading, link/widget translation problems, need for manual translation). [42]

Oxygen 6 documentation does not counter this with an official WPML integration claim, and instead highlights other tools (TranslatePress, Polylang, Weglot) while disclaiming responsibility for third-party integrations. [43]

Decision-grade conclusion: If WPML's Translation Editor workflow is a requirement (especially for large multilingual operations), Oxygen 6 RC 1 should be considered **high risk** unless you are explicitly willing to (a) translate manually, and/or (b) engineer custom workarounds and accept ongoing maintenance. [44]

Alternatives that are better-aligned with Oxygen's documented direction

Oxygen's own multilingual documentation recommends Weglot and positions TranslatePress as the most common choice among users. [8]

TranslatePress provides an explicit Oxygen integration path for language-dependent element visibility using `get_locale` and Oxygen's conditions UI, which is conceptually aligned with builder-driven output: you are not relying on WPML's extraction of builder-internal strings so much as translating or conditionally rendering front-end output based on locale. [45]

Weglot publicly lists Oxygen Builder as compatible, and Oxygen itself recommends it, suggesting it is the "least resistance" route among the options mentioned by Oxygen. [46]

RTL and Arabic: the governing constraint is still CSS directionality

Even if multilingual content is solved via a translation tool, Arabic/RTL success depends on whether your layout system is direction-aware.

WPML explicitly notes that RTL formatting applies only when themes support RTL. In builder-centered sites, this functionally translates to: **your CSS architecture must support RTL**, not only your translation plugin. [18]

Oxygen's open RTL-related issues (editor RTL friction, left-alignment defaults, lack of logical spacing controls) indicate that without a deliberate CSS strategy, RTL sites will require repeated manual fixes. [47]

CSS Logical Properties: feasible, but not "turnkey" in the builder UI

Oxygen 6 describes itself as mapping properties to CSS and generating dynamic CSS from design properties like padding. [48]

However, public community requests explicitly ask for logical-property support (`inline-start/end`) in the properties pane for RTL efficiency, implying this is not natively available as a UI-level mode. [29]

At the same time, Oxygen supports custom CSS injection through its CSS Code element, enabling teams to standardize on logical properties by policy (e.g., adopting a logical-property class framework and minimizing visual left/right overrides). [37]

Decision-grade conclusion: Oxygen 6 RC 1 can be used in a logical-property-first, RTL-friendly architecture **if** your team enforces CSS governance (class-first styling, logical properties in code, avoid physical overrides). It does not present evidence of being a “native logical properties generator” in its visual controls, and RTL-related issues suggest this remains a manual discipline rather than a built-in capability. [49]

---

[1] [5] [32] [34] [40] Oxygen 6 RC 1 Is Now Available — Oxygen
https://oxygenbuilder.com/oxygen-6-rc-1-is-now-available/

[2] [9] [42] Compatibility between Oxygen plugin and WPML
https://wpml.org/plugin/oxygen/

[3] [25] [47] Editing RTL websites · Issue #1052 · soflyy/oxygen-bugs-and-features · GitHub
https://github.com/soflyy/oxygen-bugs-and-features/issues/1052

[4] [19] [20] [45] Oxygen Builder Integration - TranslatePress
https://translatepress.com/docs/restrict-by-language/oxygen-builder-integration/

[6] [7] Launch Frequently Asked Questions — Oxygen
https://oxygenbuilder.com/launch-faq/

[8] [16] [27] [43] [46] Multilingual Plugins — Oxygen
https://oxygenbuilder.com/documentation/integrations/multilingual-plugins/

[10] [17] [24] [44] Oxygen Builder - Completed translations do not load on the front-end - WPML
https://wpml.org/errata/oxygen-builder-translations-are-complete-but-theyre-not-being-loaded-on-the-frontend/

[11] Oxygen Builder widget and link translation issues - WPML
https://wpml.org/errata/oxygen-builder-widget-and-link-translation-issues/

[12] Oxygen Builder integration with WPML - WPML
https://wpml.org/forums/topic/oxygen-builder-integration-with-wpml/

[13] Multilingual Support | Oxygen - The Visual Site Builder for WordPress
https://classic.oxygenbuilder.com/documentation/other/multilingual/

[14] WPML - Mixed languages when logged out and Rich Text ...
https://github.com/soflyy/oxygen-bugs-and-features/issues/171?utm_source=chatgpt.com

[15] WPML and Oxygen: re-usable part issue #1419
https://github.com/soflyy/oxygen-bugs-and-features/issues/1419?utm_source=chatgpt.com

[18] Does WPML have RTL support (right-to-left)?
https://wpml.org/faq/does-wpml-have-rtl-support-right-to-left/?utm_source=chatgpt.com

[21] Weglot | Compatible Platforms & Tools
https://www.weglot.com/documentation/compatibilites

[22] [28] New Divs align left by defaut · Issue #3228 · soflyy/oxygen-bugs-and-features · GitHub
https://github.com/soflyy/oxygen-bugs-and-features/issues/3228

[23] Gutenberg Editor doesn't work with Polylang multilingual ...
https://github.com/soflyy/oxygen-bugs-and-features/issues/718?utm_source=chatgpt.com

[26] Better RTL Support · Issue #30 · soflyy/oxygen-bugs-and-features · GitHub
https://github.com/soflyy/oxygen-bugs-and-features/issues/30

29 35 38 41 49 padding-inline instead of padding-right For easy multi-language support · Issue #3113 · soflyy/oxygen-bugs-and-features · GitHub
https://github.com/soflyy/oxygen-bugs-and-features/issues/3113

30 31 RTL Support for Oxygen Builder – WordPress plugin | WordPress.org
https://wordpress.org/plugins/rtl-support-for-oxygen-builder/

33 48 Oxygen 6.0 Beta 2 is Now Available
https://oxygenbuilder.com/oxygen-6-0-beta-2-is-now-available/?utm_source=chatgpt.com

36 37 39 CSS Code
https://oxygenbuilder.com/documentation/reference/elements/other/css-code/?utm_source=chatgpt.com