# Defaults and Arcade

It is often challenging to reconcile differences between a given repo build and Arcade - which represents .NET overall. It's sometimes tough to determine when to conform to how Arcade expects settings/properties/patterns/targets/conventions to be and when to 'override' (fork) for a given repo. This document attempts to provide some guidance which hopefully gives at least a place to start - as well as a 'frame' for discussions.

## Reminders/Principles

- There is significant value in sharing a common engineering infrastructure for .NET.
- Buying into (not necessarily agreeing to always….grin) a shared/common infra is always hard - and expensive.
- The 'how' Arcade does something is always open for discussion. We should all be working together in an attempt to have the right implementation that works best for .NET.
- We're all in the same boat - we all ship as part of .NET. In other words, no 'privileged' repos here… (the egalitarian sword cuts both ways though…)
- We prefer defaults that match the defaults used by 'vanilla' Visual Studio/C# code formatters/MSBuild/.NET SDK.
- Weigh the costs of adopting different alternatives by looking at impact on existing Arcade repos across the product (.NET) as a whole.
- Give consideration to how this affects new repos onboarding to Arcade.
- We try and avoid blocking people based on differences, but still strongly encourage consistency.

## Categories of Differences

- Category 1 - Company, Divisional, Visual Studio, or .NET requirements.
- Category 2 - Differences which are truly exceptions to what is otherwise correct for .NET overall.
- Category 3 - Differences which are excessively expensive or risky to change.
- Category 4 - Differences which are the result of team/individual preferences and/or history.

## Category Criteria ("bar" for each) and Mitigation Guidance

### Category 1 - Company, Divisional, or .NET requirements

- Hard requirements (like SDL or certs) that .NET must confirm to.
- These differences **should have no defaults** and should have the required parameters explicitly provided by the repo based on supported options presented by the Arcade SDK.

- This might also result in cases where the implementation choice Arcade took proves excessively expensive or unreasonable for a specific product team to adopt. (see other category for more on this)
- Obviously, any differences here need to be reconciled - but there's always room for the "how" discussion.

**Category 2 - Differences which are truly exceptions to what is otherwise correct for .NET overall**

- True exceptions mean that there's a concrete business case driving the difference.

- These differences have defaults (or preferences) that are set by Arcade which can be overridden **only** for this category of differences.
- Preference or history don't fall into this category.
- These differences should always be the *exception*. If multiple repos are experiencing the same differences, this likely points to a broader .NET (Arcade) change.

**Category 3 - Differences which are excessively expensive or risky to change.**

- Sometimes, the 'right' thing to do is simply too expensive or risky to change right away.
- *Expensive* here is defined as taking more than 2 man-weeks of dev time to reconcile.
- *Risky* here is defined as likely destabilizing .NET for more than 2 days.
- The mitigation for this category is to create a plan for moving to the 'right' place over time - and then executing on that plan. As with most things, 'scope' is usually where the magic of goodness lives.
- This mitigation plan should take into account the cost of moving, and the end value over time as well.

**Category 4 - Differences which are the result of team/individual preferences and/or history**

- In a perfect world, this category would not exist.
- It is **strongly** recommended that the Arcade behavior is adopted.

- In most cases, repo owners can override the Arcade defaults. It should be understood that this will often result in additional risk and cost over time. Regardless, it should continue to be the repo team's 'call' what is done.
- Eliminating (or reducing at least) this category is part of the cost of getting the value of a shared infra.
- It's hard to do….as it often involves team culture change. The idea though is that the effort (and pain) is worth it.

Was this helpful?