Documentation Principles

This document is to provide value guidance when creating Arcade documentation. This supplements the .NET Core Engineering Services Principles.

Principles

1. Documentation is successful only if the customer is successful.
2. Documentation, not code, is the primary source of information for customers.
3. Documentation is written for the customer.
4. Documentation is up-to-date.
5. Documentation is explicit.
6. Developers, including First Responders, are able to contribute documentation easily and without disruption.
7. Features are documented. Feature documentation explains why the feature is needed, what the feature does, and how the feature is used.
8. Questions are documented. If someone had to ask, it was overlooked in the docs.

Project assumptions

- Value ease of maintenance and information quality.
- Arcade's customers are developers who understand their project's current build. Therefore there is high value in providing a mapping of existing non-Arcade build steps and concepts to Arcade build steps and concepts.
- Arcade's customers have high technical knowledge. Use this expectation to avoid over-documenting simple tasks.
- Examples should be simple but not trivial.
- Documentation starts with a single landing page (like current StartHere.md). All other documentation may be found within a few clicks of here.
- Documentation is organized by category, then by product. Categories borrow from 1ES and mirror development flow. This provides natural boundaries and flow for both customers and developers, improving understanding. Existing documentation maps well to this space. The categories are: Code, Build, Test, Deploy.
- Markdown files are sufficient for current efforts. GitHub pages provides a nice path for extending simple Markdown support, adding options and complexity. These options may be added as needed.

- The Arcade Project is itself a product, though it may also be viewed as a collection of smaller products. As such, Arcade also benefits from having product documentation.
- A "Frequently Asked Questions" section within product documentation is an easy place to address a question or provide some piece of information that otherwise does not have a home (perhaps because that part of the documentation has not yet been written, or perhaps because the question is so common it benefits from being highlighted). The Arcade repository has a FAQ document to act both as a place for questions about Arcade and to provide a central hub for links to other, product-specific FAQs.

Writing documentation

Documentation is centered around a product. Product documentation contains all knowledge necessary for a user to be successful. They may be a single page or many as best fits the needs. Documentation spanning multiple documents should be anchored from a single starting document, perhaps as a table of contents or integrated with the Overview section.

Documentation for a product may be stored in multiple places. For example, a User's Manual may reside in a Azure Repo repository alongside the code while the design documents live in a Word document on SharePoint. In this case it is important that documentation be anchored in a single starting document.

Topics typical for product documentation include:

- An *overview* describing the need the tool fulfills and how it does this.
- *Requirements* and specifications for the product's execution and operation
- *Architecture* and design details about the product's construction
- *Concepts* or in-depth explanations fundamental to understanding and use
- *How-to guides* or task-based recipes of the product's use
- *Examples* or samples illustrating the use of the product
- *Reference* information or detailed product APIs, CLI commands

An ideal documentation set would contain all of these items at an appropriate level of verbosity. When this is not possible, remember Customer Obsessed: Include information necessary to ensure a successful customer experience.

Be mindful of the differences between documentation intended for users of a product and documentation intended for developers of a product. Keeping the two separate can improve a user's experience, and providing design information to users can improve a user's self-sufficiency.

Documentation reflects the latest version of the product. The documentation should be updated whenever product functionality changes. If needed, consider document migration issues and breaking changes in a separate "Migration Issues" section.

Resources for writing

Some information to guide writers.

- The 5 Writing Principles
- Microsoft Writing Style Guide

A Template

Below is a simple template for user-facing documentation.

```
# Product++

Product++ is a tool that moves bits from Here to There, performing all of the tedious and er

This product is owned by Us. For support, see [How To Get Help](how-to-get-help.md).

## Requirements and Architecture

The Product is a web-based service. See [https://github.com/dotnet/core-eng/blob/master/Doc

## Concepts

### Speculative File Copy

Speculative File Copy is the idea that the file being copied is already present at the desti

## How to Use
Product++ is a RESTful webservice driven entirely by simple JSON statements. A typical opera

### Copying two files from remote locations.
Provide the source files in the *source* array payload and the destination files in the *des

### How do I use this with Other Project?
See also [Other Project FAQ](other-project-faq.md) for questions related to interoperabilit

### How fast does this copy files?
Really fast.

### I'm getting HTTP Error 429.
This indicates the server received too many requests from you. Use batching to send fewer re

### Migration from previous versions
#### Migrating from versions 0.2 to 0.3
See [Migrating from v0.2](Migrating-0.2.md).

#### Migrating to 1.0 from earlier versions
```

`Version 1.0 works always; there is no change needed.`

Was this helpful?