Guidelines for Auto-generated Documentation for Platform APIs

About this Guide

This document provides guidelines and outlines best practices on:

- Marking up source code comments for auto-generated platform API reference documentation for a given programming language.
- Creating conceptual material that is easy to read and understand for the target audience and conforms to the *Extreme Web Style Guide* for HTML outputs (published on GitHub or using a static site generator).

Goals and Audience

The primary goal of this guide is to provide editorial assistance to Extreme developers on structuring and writing platform API documentation; to help them keep their documentation consistent with other Extreme documentation and meet the needs of their target audience (external developers using their APIs).

Python API Documentation (reStructuredText + Sphinx): Writing Guidelines

The ExtremeXOS team uses the Sphinx tool to document Python APIs/Apps. Sphinx uses reStructuredText as its markup language and can generate output in multiple formats including HTML, LaTeX (for printable PDF versions), and ePub.

The Sphinx HTML output style (colors and font) should be modeled based on the EXOS Python API page - http://documentation.extremenetworks.com/python/index.html.

Extreme Colors

Name	RGB	CMYK	Hex (Web)	Pantone (Print)
Extreme Purple	R64 G0 B153	C90 M100 Y0 K0	440099	Violet C
Extreme Gray	R119 G119 B123	C55 M47 Y43 K10	77777b	Cool Gray 9 C
Extreme Orange	R255 G107 B0	C0 M72 Y100 K0	ff6b00	1505 C
Extreme Blue	R9 G113 B206	C84 M54 Y0 K0	0971ce	285 C
Extreme Green	R59 G175 B41	C76 M3 Y100 K0	3baf29	361 C
Extreme Yellow	R245 G195 B0	C4 M22 Y100 K0	f5c300	7406 C

Graphics

- To create a diagram, use any drawing tool.
- Save your image as a .png or .svg file.
- Include a caption for the figure in the HTML output.
- Do not use shading, shadows, borders, highlights, or other post-production techniques.

Content Organization

Information should be organized based on core tasks that your users will want to do.

- A Table of Contents should be provided. It should link to headings at levels 1 and 2 at a minimum, but should primarily help the user navigate to most desired sections.
- Write simple, concise, declarative sentences in present tense.
- If your document generation tool provides a mechanism for adding search, add it in a conspicuous place (such as above the TOC or at the top-right of the content area).
- Consider including the following sections:
    - Overview of the API

      The overview explains what you can do with the API and who the API is for. List the API features. Any visual information such as architecture or data model diagrams can be provided here.
    - Getting Started

      This section details the first steps users must take to start using the API. They might involve the following:
        - System Prerequisites/Requirements – what OS, SDKs, IDEs, etc. are required. List version numbers and release dependencies.
        - Installation steps
        - "Hello World" tutorial – Show the user how to use your API to get the simplest possible output with the system. Use screenshots wherever possible.
    - Tutorials
        - This section contains frequently used scenarios/task-based examples with code samples (3 to 5 are sufficient).
        - Make code samples copy-paste friendly.
        - Clarity is more important than efficiency: Long, unwieldy names for variables, classes, members in the sample code are fine.
        - Use hard-coded values for sample code.
        - Incorporate syntax highlighting.
        - Use consistent white space to make the code snippets easily readable.
        - Add both code comments and before-and-after explanations.
        - Focus on the parts of the code unique to the company. Link to other resources that developers can rely on for general coding.
        - Each code sample needs to be tested and maintained.
    - API Reference
        - This section is auto-generated from source code markup and describes the modules, objects, classes, and functions.
        - Document all modules, classes, functions, and public methods using the PEP-257 conventions. For example, in the first sentence

of a class description, briefly state the intended purpose of the class with information that cannot be obtained from the class name.
        - For method descriptions, document any dependencies that are needed to call the method, and how the method behaves if such a dependency is missing.
    - Troubleshooting
      Include tips for common problems when using the API (authentication errors, path errors, exceptions, etc.).
    - License/Copyright Information
    - Support
      Link to the Extreme support portal & The Hub

GitHub Readme File Guidelines

Always provide a readme file when hosting code in GitHub. The readme file should follow the same general structure and content guidelines as above. There are no style recommendations except to follow the Markdown formats and make headings, lists, and code examples clear to readers. Provide links to the Extreme Documentation page and the support portal if possible.