"Wallaby" SDK Documentation Content Strategy

Last updated: 1/16/2012 1:44 PM

OWNER	Marvin Martian		
PROGRAM MANAGER	Wallace Grommit	**REVIEWERS**	Samuel Marchbanks, Leonard Tolstoy, Woodrow Wilson

1 Overview & Scope

Documentation is the lifeblood of all SDKs: an SDK without good documentation is of little value to customers. Documentation for developers is more than just the words that an SDK ships – it's also the code snippets and code samples. This document describes the vision, strategy, and direction for "Wallaby" SDK documentation content. It identifies goals and explains the rationale for investing in those areas. It also identifies dependencies and risks.

The intended audience for this document is the "Wallaby" SDK team, the "Wallaby" SDK content contributors, marketing, management, and partners such as the Internet Developer Network (IDN).

This document does *not* cover the following related areas:

- Plans and processes for achieving documentation content goals and meeting dependencies. These will be described in subsequent documents.
- Confidence levels that goals will be achieved.
- Strategy and goals for some aspects of the user experience with "Wallaby" SDK content, such as navigation, discoverability, and UI design. For specifications describing navigation, discoverability, and UI design for the "Wallaby" SDK, see http://team/sites/oz/canberra%20Specs/Forms/AllItems.aspx.
- Strategy or goals for "Wallaby" SDK tools and binaries.
- Localization. The decision to localize or not (and in what languages) will be decided at the VP level.

2 Definitions

This document uses the following terms as described:

- **Code sample** – File or project-based code that demonstrates the usage of one or more technologies. Usually available in both executable and raw source form.
- **Code snippet** – Relatively small bits of code that appear in API reference topics.
- **Content** – Everything an SDK ships, including documentation content, tools, and product binaries.
- **Content viewer** – The application that enables customers to discover, view, navigate, and use SDK documentation content. For the "Wallaby" SDK, the plan of record is for the "Canberra" client to be the content viewer.
- **Cross-technology scenario** – The constraint or condition that specifies an application will span multiple technology areas, such as "Avalanche" and "Indigo".
- **Discoverability –** The ability to find documentation content that exists. This is usually done through Search, Index, TOC, and/or hypertext.
- **Documentation content** – Information that identifies, explains, and/or demonstrates the behavior of a shipping product; documentation content for developers usually consists of descriptive text, small code examples, and executable file or project-based samples. Most developer products ship some documentation and a content viewer with the product.
- **Documentation content contributor** – Individual or team who creates documentation content and supplies it to the "Wallaby" SDK.
- **Graphical code** – code native to "Wallaby."
- **How-to topic** – A documentation topic that shows how to accomplish one task. Includes code and a small amount of introductory text.
- **Navigation topic** – A documentation topic that contains primarily a list of hypertext links to other topics.
- **Non-graphical code** – Source code that targets the common language runtime and benefits from the services that it provides. Non-graphical code is compiled to an intermediate language and then JIT-compiled at installation time or as needed at run time.
- **Scenario** – A constraint or situation that exists when a developer writes code, such as a scenario where the application needs to run on Windows NT, or a scenario where the application needs to interoperate with existing non-graphical code.
- **Task** – Something that a developer is trying to accomplish in her code and is likely to try to find out how to do in the documentation. Tasks can be fine-grained, i.e., "write to a file", or they can be more complex, i.e., "configure your application."
- **User education (UE) team –** A group individuals, typically consisting of writers, editors, and production staff, whose mission is to create and publish documentation for ArtPak products. Also known as User Assistance.
- **Walkthrough** – A documentation topic that orients the reader to a feature area by providing specific instructions on how to create an application that uses the feature. Contains multiple procedures connected by explanatory text.

3 Wallaby SDK Goals, Audience, and Usage Scenarios

3.1 Goals

The “Wallaby” SDK is the set of documentation, samples, tools, binaries, and build environments that developers will use to design and build applications, libraries, and tools that target the “Wallaby” platform. The “Wallaby” SDK must enable developers to successfully use the APIs that ship in the “Wallaby” GUI. The key focus of API innovation in “Wallaby” is the set of non-graphical code libraries, called Eucalyptus™. The Eucalyptus class library extends and enhances the functionality and programming model introduced by the .PAINT Framework.

The “Wallaby” SDK is of critical and strategic importance in winning developers to the platform. The SDK is especially important to the success of early adopters because the SDK is likely to be the only (or the most stable) environment available for building applications in pre-“Wallaby” RTM releases. A “Wallaby” SDK that meets developer needs for completeness, accuracy, and discoverability is required for developers to successfully target the platform. An *outstanding* “Wallaby” SDK would help drive adoption of the platform by exposing and illuminating the breadth and depth of innovation in “Wallaby” and demonstrating ArtPak’s commitment to developers.

3.2 Audience

This section describes the developers who are expected to use the “Wallaby” SDK. These expectations are based on “Wallaby” SDK team members’ and documentation content contributors’ experience with previous SDKs and on marketing data about users of previous SDKs. The audience for the “Wallaby” SDK includes both internal and external developers.

Note that content intended only for producers will not be included in the “Wallaby” SDK. However, there will be some content in the SDK that is appropriate for both producers and developers. UE teams who own production-specific tools and functionality are responsible for ensuring that appropriate documentation for the producer audience ships with the GUI or on the web, as appropriate. The primary reason for excluding production-specific content from the SDK is that producers should not have to install the SDK and search through it to find the content they need.

Developer roles

Based on experience with previous SDKs, we believe that the majority of developers who use the SDK *outside of Artists Studio* are likely to be in one of the following roles:

- *“bleeding edge” technology investigators and early adopters* - use the SDK to evaluate the new technologies and determine whether they want to adopt them. If they become early adopters, they use the SDK to develop applications that can serve as early examples of success with the product. Early adopters often identify issues with the product, and product team members can use this information to decide which issues need to be fixed before release.
- *academic researchers* - use the SDK as part of their research into computer graphics topics.
- *high school teachers and college professors* - require or encourage the use of the SDK in the courses they teach.

- *independent software vendors (ISVs)* - use the SDK to learn the technology and develop applications that use it.
- *students* - use the SDK in their coursework.
- *web developers*, who often prefer a streamlined development environment
- *tools developers* – use the SDK to design compilers, pre-production renderers, performance profilers, and other tools that support graphical application development using "Wallaby" technologies.
- *graphic designers and other professionals who traditionally work on web pages*.
- *hobbyists* - use the SDK to develop graphical applications for their own personal use and to learn about programming.

Developers who use the "Wallaby" SDK *within Artists Studio* could come from any and all professional developer roles, such as enterprise developers and ISVs.

Developers who use the "Wallaby" SDK documentation *on IDN* could be in any and all developer roles.

Programming expertise

We expect that developers who use the "Wallaby" SDK (in one of its forms) will have expertise that runs the gamut of programming knowledge and experience, as follows:

- Highly skilled computer scientists or researchers, aka "rocket scientists".
- Corporate developers, who usually have computer science training, such as Certified Software Development Engineers.
- Enterprise developers, many of whom are self taught and have no formal computer science training.
- Beginning programmers, such as students.
- Some ISVs and enterprise developers will have experience with non-graphical code because they have used the .PAINT Framework 1.0/1.1 and/or Artists Studio 4.0. Some will be familiar with the Kangaroo API and/or DHTML and have little to no experience with non-graphical code. Others will be familiar with Easel programming only to the extent that they called built-in language functions to create their Easel-based applications.

Technology usage and application type

Developers using the "Wallaby" SDK will be able to write all types of piggyback wireless applications. There is likely to be heavy usage of "Wallaby" presentation technologies since "Sydney" is a key part of "Wallaby" innovation and the initial "Wallaby" release is a Client release. However, "Wallaby" developers will be able to build components for both clients and servers, and they can design distributed applications that use a service-oriented approach.

We expect that non-graphical code will be used to develop most "Wallaby" applications. However, some applications will require the use of graphical APIs, such as "twitch" games and CAD applications and other scenarios where high-performance 3D graphics are required. New graphical APIs are being developed for "Wallaby", and we expect that developers will use these graphical APIs to extend existing graphical applications. Some developers will ensure that their existing applications benefit from "Wallaby" features by developing non-graphical components that can interoperate with existing graphical components.

Programming/markup language preference

Art Basic programmers comprise the largest audience in terms of the number of developers using the Art Basic language. However, we expect that many AB developers will use the SDK only in the context of Artists Studio. Developers who use the SDK without an IDE tool are likely to be AB++, AB#, J#, and advanced AB developers. Developers who previously used DHTML are likely to use XAML and perhaps JScript.

Additional Resources

ISV archetypes: http://wallaby/audience/ISV%20Archetypes-Overview.htm

Artists Studio developer personas: http://wallaby/teams/usability/personas/

"Wallaby" Developer Scenarios: http://wallaby/scenarios/default.htm#Developer%20section

3.3 Usage Scenarios

There are 4 key usage scenarios for the "Wallaby" SDK:

- online (IDN)
- Artists Studio
- offline (non-Artists Studio) SDK standalone
- SDK used with 3rd party development environments

Each of these scenarios has certain unique characteristics by design. Yet, to help ensure there is no learning curve or disconnect for customers moving from one scenario to another, the customer experience with "Wallaby" SDK content needs to be consistent in approach, design, and basic function between online, offline, and Artists Studio, while still allowing for appropriate differences. Here are some of the customer expectations for the above scenarios that we should keep in mind as we try to keep the customer experience consistent:

- The IDN online library should contain all of the content in the "Wallaby" SDK, and IDN should have more up-to-date content than the offline SDK, when such content is available.
- IDN should expose a superset of the offline content (such as whitepapers, columns, etc.).
- The IDN online library and the "Wallaby" SDK default TOC need to be organized using the same overall approach.
- Because Artists Studio leverages the context of the customer's development environment to target specific kinds of documentation content and community connections, in the Artists Studio documentation, there is likely to be functionality not available in the standalone SDK, such as IntelliSense and F1.
- Customers should be able to choose for the presentation of the content to be consistent between the standalone SDK and Artists Studio.

4 ArtPak SDK Background & History

Previously shipped ArtPak SDKs have paved the way for the “Wallaby” SDK by setting developer expectations for documentation content coverage and quality. This section discusses some existing ArtPak SDKs and the developer expectations they have set. It also describes some of the key takeaways from customer feedback for these SDKs. Content for all of the SDKs listed below is available on IDN.

Easel SDK

The Easel SDK (ESDK) is the closest living relative of the “Wallaby” SDK. In fact, the “Wallaby” SDK is an evolution of the ESDK. Both SDKs are tied to graphical user interface (GUI) releases and help ensure that developers can build applications that target the GUI. The ESDK includes documentation for the Canvas APIs, the Palette APIs, ArtMedia APIs, and many others, to support development on multiple graphics platforms. It contains about 30,000 API reference topics, as well as technology overviews and code samples. Its primary audience has been developers who write graphical AB and AB++ code. Easel SDK contributors have the autonomy to determine the focus, format, and organization of their documentation content. The documentation content for each technology is located in separate Table of Contents nodes. The ESDK contains little documentation content that explains how to use one technology with another or how to decide between various technologies. There is little consistency between the design of the various kinds of documentation topics, and no consistent approach to organizing them.

ESDK customers have contributed feedback on what they need and expect in documentation. They have indicated that they appreciate being able to get information on all features in the GUI in one package. They have asked for the ability to easily get the updated docs and specific technologies that they are looking for. Customers frequently use the ESDK feedback mechanism to report such issues as incorrect information in the documentation as well as difficulties with the content viewer. Customers also ask for more code or code in a different language, both in API reference topics and in task-oriented topics. They also report that they need information on the various connection points between technologies and on how to decide between technologies. Customers appreciate the thoroughness of the API reference topics, and they sometimes ask for undocumented APIs to be documented. Customers expect to get the latest and greatest documentation content on IDN. Many customers complain that they have difficulty finding the documentation content they need, even when they know it exists, to the extent that many customers use Google to search IDN.

The ESDK has an email-based feedback mechanism, to which ESDK contributors respond. The ESDK publishes an update whenever there is a new GUI release. In addition, some ESDK contributors push documentation fixes to IDN online as new information is available, and to the IDN subscription CD approximately every 3 months.

ArtNet SDK/ArtWeb Workshop

The ArtNet SDK/ArtWeb Workshop was a standalone, web-based collection of documentation content and tools designed to help developers create web-based solutions using HTML, DHTML, cascading style sheets, and scripting. It also provided a Web Publishing SDK and content management tools. The audience for the Web Workshop content included website developers and ISVs who were integrating Internet functionality into their applications. The SDK’s primary mission was to help drive adoption by documenting and demonstrating the capabilities of Internet Explorer. Web Workshop content was merged with the IDN library two and one half years ago.

This SDK collected a variety of API reference documentation topics from various contributors and presented them in a unified way that was optimized for web developers. Primary content types include reference topics (DHTML objects, COM interfaces, Canvas functions, and XML elements), tutorials/how-tos, and overviews.

The TOC for Web Workshop was shallow by design to help improve discoverability. A leaf node in the TOC typically consisted of a list of links to remaining content (aka, a portal). Customers could run code samples directly from the points at which they were referenced via a clickable button. Tables were used to minimize the visual area needed to capture the categories of members supported by interfaces.

Some of the takeaways from customer feedback on the Web Workshop include:

- "Cool" UI features do not always lend themselves to a good user experience. Care needs to be taken during the design phase to consider potential accessibility issues, and usability testing needs to take place.
- The ability to run a code sample from within the documentation topic can be a very helpful and compelling feature.
- Code examples need to clearly demonstrate the feature being documented.
- Everything in the public header files needs to be documented, and we need to clearly indicate when an API isn't implemented yet.

.PAINT Framework SDK

The .PAINT Framework SDK is a relative newcomer to the SDK scene. It includes documentation for the .PAINT Framework redistributable, which first shipped in the GUI in ArtPak 3.0 and ArtWeb 3.1. The .PAINT Framework SDK and the "Wallaby" SDK are similar in that they both document a large number of non-graphical APIs, and the "Wallaby" SDK documentation will include the .PAINT Framework documentation. The primary focus of the .PAINT Framework SDK has been on API reference topics, the vast majority of which are non-graphical APIs. In version 1.1, the .PAINT Framework SDK contained documentation for about 7,000 API reference topics, and 33% of API reference topics contained code examples. The descriptions in the API reference topics shipped in the .PAINT Framework versions 1.0 and 1.1 are sometimes less complete than the core Canvas API reference topics in the ESDK.

Customers have used the .PAINT Framework SDK feedback alias to report documentation errors and to request additional code examples and more complete descriptions in API reference. Additional how-to topics and improvements to discoverability have also been requested. .PAINT Framework documentation content contributors respond to customers who use the feedback alias. The .PAINT Framework documentation content has been updated once in downloadable form since Version 1.1 shipped. Additional updates have been made via IDN online and the IDN quarterly.

5 “Wallaby” SDK Documentation Content Needs

In addition to addressing customer feedback on previously shipped ArtPak SDKs, the “Wallaby” SDK has some unique documentation content needs. These needs derive primarily from the huge wave of innovation being exposed in “Wallaby” and the overall size and scope of the “Wallaby” SDK. A description of key “Wallaby” documentation needs follows.

Developers targeting “Wallaby” will need to understand the benefits of the new programming model (non-graphical code) and how to build “Wallaby” applications with Eucalyptus from the ground up. They also need to understand the migration path to “Wallaby” and how they can extend or modify their existing applications to benefit from “Wallaby” innovation. Developers will also want to know the non-graphical way to do something that they already know how to do in graphical code. The “Wallaby” SDK must provide API reference, how-to content, migration guidance, and overviews of Eucalyptus to support these needs.

In “Wallaby”, all of the Canvas APIs that shipped in ArtPak 1.0 will be available, plus a significant number of new graphical APIs. Therefore, “Wallaby” will contain APIs that enable developers to accomplish the same things using two different programming models (non-graphical and graphical). There will also be multiple technology choices for a given scenario. The “Wallaby” SDK must provide documentation content that enables developers to determine which programming model they should use and which technology is most appropriate for their scenario. In addition, the SDK needs to explain the connection points between the various technologies, such as which technologies require the use of others or work better together, and provide performance tips where appropriate.

Because “Wallaby” includes APIs for both non-graphical and graphical development and because the scope of innovation in “Wallaby” is large, the number of APIs that need to be documented and published in the SDK is likely to be well over 100,000. This high volume of documentation content requires the SDK to ensure that developers can work with one or more subset(s) of the documentation they are interested in (instead of dealing with all APIs at once in the Table of Contents, Search, Index, and navigation topics). Developers should also be able to customize the documentation to the extent that they can build a custom view and organization of the documentation, based on their own preferences and usage patterns.

6 “Wallaby” SDK Documentation Content Goals

To address customer expectations and needs for “Wallaby”, the “Wallaby” SDK needs to deliver documentation content that meets a set of goals that no ArtPak SDK of similar complexity and scope has ever achieved. In other words, the “Wallaby” SDK documentation content needs to be of higher quality and more complete than any SDK content ArtPak has ever shipped.

The high level documentation goals for the "Wallaby" SDK are as follows. The first three goals are focused on the content itself, while the remaining goals focus on the user experience with the content:

Accurate, complete, and relevant documentation content

Provide accurate, complete documentation content that is relevant to real-life development tasks and scenarios.

Multi-language code samples, snippets, and syntax

Supply abundant, compile-tested code samples, which demonstrate how to use key "Wallaby" technologies, in a variety of programming languages (i.e., AB.NET, AB#, J#, and AB++). API reference syntax must be multi-language as well.

Consistent and cohesive content

Present the "Wallaby" development platform as a consistent, cohesive, well-integrated whole.

Seamless online and offline, with community access

Give developers access to the latest and greatest documentation content from ArtPak and from the community.

Updatable documentation content

Create, package, and deploy documentation content so that it is easily updated and documentation content teams can respond to customer requests for bug fixes or enhancements.

Accessible user interface and navigation

Ensure that the SDK can be viewed and navigated successfully.

Single feedback mechanism

Provide one SDK feedback mechanism leveraged by all teams shipping documentation content in the SDK.

Customizable views

Enable developers to create custom views and subsets of the documentation.

6.1 Documentation Content Goal Drilldown

This section provides a list of sub-goals for each of the high level documentation content goals described previously. It also provides scenarios to help describe the resulting user experience.

6.1.1 Accurate, complete, and relevant documentation content

These are the primary documentation content goals. The extent to which we achieve them will significantly influence the success of the SDK. If we do not achieve these goals, the other goals described in this document are not compelling. The following list provides details:

- *Document all publicly callable APIs.* For non-graphical APIs, this includes everything that developers outside ArtPak can call. For graphical APIs, this includes all APIs in public headers. This goal is intended to ensure completeness of the API reference content.

- *Ensure that the documentation content is correct.* Content contributors will involve product developers, testers, and PMs in technical reviews for authored documentation content, and the "Wallaby" SDK will use documentation tools to auto-generate syntax. Contributors will need to be able to track whether or not each of their topics has been reviewed for accuracy. "Correct" means that code snippets compile, code samples compile and run with the expected output, all code demonstrates good coding practices, and all text descriptions are accurate.
- *Cover the key product usage scenarios, including cross-technology ones.*
- *Provide migration content.* This includes, but is not limited to, hypertext links from graphical APIs to their non-graphical counterparts and topics that explain why and how to migrate graphical applications to non-graphical code.
- *Provide navigational portals that serve as roadmaps to guide users in their efforts to learn about the various technologies in Wallaby.*
- *Explain connections between technologies and identify choices and tradeoffs.*
- *All documentation content should receive a substantive language edit as well as a copy edit.*
- *Meet consent decree and other legal requirements.*

The following table describes relevant documentation content goals by milestone:

	Beta 1	Beta 2	RC/RTM
Documentation completeness	*Most documentation content in place. *30% of key scenarios documented.	*Migration content complete. *Ref docs 90% complete. *Task docs 80% complete. *60% of key scenarios documented.	*All documentation content complete. *All key scenarios documented. Plan to update/improve documentation content as needed on web.
Tech review (technical accuracy)	All existing documentation content tech reviewed.	All existing documentation content tech reviewed.	All documentation content tech reviewed.
Edit	*100% of documentation content has legal/copy edit. *70% of documentation content has substantive edit.	90% of documentation content has substantive edit.	100% of documentation content completely edited.

Some content contributors might be unable to achieve the goals described in this section, due to time or resource constraints. Those contributors should prioritize their efforts in the following way to help ensure that the highest priority “Wallaby” SDK goals are met:

- *Prioritize effort on documentation content for API reference topics over all other kinds of authored documentation content.*
- *Prioritize effort on documentation content for tasks that include code or procedures over feature overviews.*
- *Prioritize work on documentation for non-graphical APIs over graphical APIs.*

Scenario: Joe, a developer, uses the “Wallaby” SDK documentation to learn how to develop an application that uses Eucalyptus. When he uses the API reference topics, he never finds one without summary descriptions, parameter descriptions, and return value descriptions, and he never finds erroneous content. Neither does he find typos, grammar errors, nor sentences that are unclear. Joe has never before seen such thorough, well-written documentation! Joe realizes that ArtPak has made a significant commitment to developers in “Wallaby”, and he’s more convinced than ever that he should start building “Wallaby” applications.

6.1.2 Multi-language code samples and syntax

This is the second most important group of goals. They are based on the expectation that developers using the “Wallaby” SDK with a variety of programming languages and address customer requests for abundant code in SDK documentation.

The “Wallaby” SDK documentation must:

- *Provide syntax in all of the following languages on all API reference pages: AB.NET, AB#, and AB++.*
- *Provide XAML syntax in topics for APIs that support XAML.*
- *Provide abundant compilable code snippets in API reference topics.* The following code snippet coverage goals apply, by milestone:

	Beta 1	Beta 2	RC/RTM
Code coverage	45% of all non-graphical and “Wallaby”-specific graphical APIs have code samples in at least one language.	65% of all non-graphical and “Wallaby”-specific graphical APIs have code samples.	80% of all non-graphical and “Wallaby”-specific graphical APIs have code samples.

- *Provide code snippets in the programming language(s) appropriate for the API.* The plan for which language(s) the snippets for a given technology are written in will be based on the scenarios in which the API is expected to be used and the programming language(s) that are the most appropriate targets for those scenarios. Such plans will be created by UE teams in collaboration with their product teams, and the plans will be reviewed by a set of programming language and “Wallaby” technology team reviewers.

Scenario: Jill is an AB# developer who designs web services. Whenever she uses the web services API reference pages, she finds plenty of code in AB#. She uses the code snippets more than the written content, and often cuts and pastes code from the documentation into her applications.

- *Provide executable samples that demonstrate key "Wallaby" functionality, including cross-technology samples.*

  Scenario: Fred, an ISV, is browsing the "Wallaby" SDK content online. He wants to find out what "Wallaby" is all about and whether he should start building "Wallaby" applications. He notices that the UI in the content viewer is the best he's ever seen, and he's amazed by the performance and flexibility of the search and query mechanisms. In a topic that lists all of the code samples in the SDK, he notices samples for "Eucalyptus", "Sydney", and "Canberra", among others. He also sees a list of cross-technology samples and finds one called "Van Gogh content viewer." He reads the description of the sample, runs it, and realizes that he could use the source code to build his own art viewer. He's so impressed with the functionality of the viewer that he downloads the SDK and starts experimenting with "Wallaby".

- *Provide in-topic filtering to enable customers to see code examples and syntax in only one programming language instead of all languages, if/when they choose to do so.*

  Scenario: Joe is an Art Basic developer. He navigates to an API reference topic that has syntax and code in AB, AB#, and AB++. Joe doesn't want to see code or syntax that isn't in AB. He realizes that to get the result he wants, he needs to set his preferences, so he chooses a preference for AB, which results in the non-AB syntax and code going away. Joe is ecstatic that ArtPak makes targeting "Wallaby" so easy.

6.1.3 Consistent and cohesive content

The "Wallaby" SDK enables development with multiple programming models and includes technologies contributed from many teams at ArtPak. The "Wallaby" SDK needs to ensure that customers understand that all of the technologies fit together in a cohesive way. In addition, the SDK needs to convey the fact that existing technologies are being supported, as well as new ones. Therefore, the "Wallaby" SDK documentation content needs to be consistent in approach, format, and organization across the documentation set.

To address these goals, the documentation for all technologies will be based on a well-specified set of topic types, as follows:

- Class library reference
- General reference
- How-to
- Conceptual (technology or feature overview)
- Sample
- Quickstart
- Glossary
- Navigational portals

Note: the unified set of key topic types already exists for non-graphical code documentation. For more information, see http://Documents/Forms/RecommendedTemplates.htm.

Documentation for all "Wallaby" technologies will use one set of style and authoring guidelines to help ensure consistency. Non-graphical API reference documentation will be organized according to the namespace hierarchy.

The default Table of Documentation contents (TOC) for the "Wallaby" SDK will be organized in a consistent, predictable way across technologies. The organization will be consistent with the approach and organization of the IDN online library TOC. Instead of being organized along ArtPak team lines, the default "Wallaby" SDK TOC will be organized around technologies. For example, all presentation technologies will be organized together. Details on the default TOC organization will be specified in detail in a separate document. Note that customers also need to be able to create their own customizable TOC. For more information, see the Customizable Views goal.

Scenario: Fred, an AB++ developer, frequently uses a search mechanism to find a topic relevant to the technology area he is working with, and then synchronizes to the TOC to browse for related topics. He recently used Search and the TOC to discover topics and learn about web services, and then he searched for content on graphics programming. He discovered that the TOC node for graphics programming is organized the same way the web services node is organized. He likes how-to topics, and he was delighted to discover that there are how-to topics for graphics programming, and he can quickly find them by simply browsing in the TOC.

6.1.4 Seamless online/offline with community

The "Wallaby" SDK documentation must give developers access to the latest and greatest documentation content that is available. The customer should be able to choose whether to view documentation content that lives online or offline (or both), and he/she should be able to access not just ArtPak documentation content, but also 3rd party and community sites.

Scenario: Jill, an AB# developer, uses the standalone "Wallaby" SDK. Jill configures the content viewer to search both online and offline for content and to download new content on web services to her local machine when it is available. While using the SDK when connected to the internet, Jill finds a topic on art web services. She doesn't notice where the content resides because it all looks the same to her (in fact, it is online and is downloaded quietly to her local cache when she views it). Later, when working offline, Jill searches for the same topic and finds it.

6.1.5 Updatable documentation content

Updates to "Wallaby" SDK documentation content must be readily and frequently available. Developers should be able to choose whether and when to download updated documentation onto their machines. Documentation content must be packaged and published in units that make sufficiently granular downloads feasible.

Scenario: Jack, an AB++ developer, works online during the day when at the office, but works offline at night when at home. He is delighted with the quality of the "Wallaby" SDK documentation and uses it nearly every day. He wants to make sure he always has the latest and greatest content on his local machine since he frequently works offline. So, Jack configures the content viewer to always download new "Wallaby" SDK content, when available, to his local machine. When Jack works offline, he finds all the same content that was available on IDN the last time he was online.

6.1.6 Accessible UI and navigation

The "Wallaby" SDK documentation content must be viewable, discoverable, and navigable regardless of the user's abilities or preferred navigation choices. The content viewer needs to follow accessibility guidelines, such as the following:

- The viewer needs to support the user's preference for color, contrast, sizes and fonts.
- All UI elements should support assistive technology software such as screen readers and magnifiers.
- When accessing UI elements using the TAB key, the tabbing order should make sense. Generally left to right, top to bottom.

For more information on accessibility, see http://Eucalyptus/writers/Presentation%20Task%20Force/Accessibility/AccessibilityRequirements.doc.

Scenario: Jill, an AB# developer, suffers from repetitive stress injuries and, as a result, never uses a mouse. Instead, she uses the keyboard to navigate. Jill is able to find and navigate the "Wallaby" SDK documentation online as well as offline. Without using the mouse, she can navigate down the page, move to the next topic, choose hypertext links, and access tooltips.

6.1.7 Single feedback mechanism

There has been some developer confusion about how to send feedback on the various ArtPak SDKs. If the customer used the ESDK, they could click a feedback link at the bottom of those pages. The same mechanism exists for the .PAINT Framework SDK. In addition, IDN places a feedback link at the bottom of topics in the online library. The "Wallaby" SDK should make it easy for developers to send feedback by offering a single, easy to use feedback mechanism. And, no matter which "Wallaby" technology the customer sends feedback on, the feedback experience should be consistent and positive. Documentation content contributors should leverage customer feedback when determining which documentation fixes to make.

Scenario: Joe, an AB developer, is looking through the documentation for a particular topic but doesn't find it. Joe wants to tell ArtPak that he needs the content, so he clicks a feedback link at the bottom of the topic he is on. A form pops up, which he fills out; he clicks Submit and the information is sent to ArtPak the next morning when Joe is online. Joe gets an auto-response email thanking him for his feedback. The programmer/writer that owns the documentation for that technology area checks the feedback database that day and discovers that a new topic is needed. She prioritizes her work accordingly, writes the new content, and turns it over to her editor. Soon, the content is completed and published. Joe gets an email saying that the content he requested is available.

6.1.8 Customizable views

Because the "Wallaby" SDK documentation set will be very large and will encompass many technologies, developers need to be able to create and view easily manageable subsets of the documentation. They should be able to choose from content subsets the SDK team has identified or create their own subset on the fly. Their choices should affect the various discoverability and navigation mechanisms (such as TOC, Search, and Index). Developers should be able to rearrange a TOC to suit their preferences, and they should be able to create navigational portals on the fly that will identify topics that could help them learn to use technologies they are interested in.

Scenario: Jill, a developer, is building a non-graphical application and is not interested in seeing documentation for graphical APIs. Jill configures the content viewer to only show her the non-graphical documentation. As a result, the TOC, Search, and Index are scoped to include only non-graphical content. Then, Jill discovers that she doesn't like the arrangement of the TOC that appears for non-graphical content, so she rearranges the TOC to suit her preferences. She then realizes that the TOC still contains more content than she wants to see. She chooses to build her own TOC on the fly from the ground up by indicating exactly which kinds of content she wants to see. Jill never uses the API reference or technology overviews, so she chooses to only have How-tos and code samples in the TOC; she also decides to include only content for the subset of non-graphical technologies that she currently uses. The old TOC is replaced with the new TOC that is based on her preferences.

7 "Wallaby" SDK Documentation Content Contributors

The creators of the documentation content for the "Wallaby" SDK are, in most cases, members of ArtPak User Education teams. The User Education (UE) teams that create documentation content for developers are typically composed of programmer/writers, editors, managers, and production staff. It is the responsibility of each UE manager to ensure that the documentation content created by their team meets the requirements of the "Wallaby" SDK and to ensure that they create and publish any necessary documentation plans. Following is a list of some of the UE teams for key components of the "Wallaby" SDK. The technology is listed, along with the UE manager's name:

- "Eucalyptus"/Shell – C. Dundee
- "Sydney" – Sidney Melbourne
- Kangaroo – Tom Outback
- .PAINT Framework – Piet Mondrian
- "Canberra" – Bob Ayers

8 Dependencies and Risks

To be successful in meeting the "Wallaby" SDK documentation content goals, several dependencies must be met:

- UE teams must be adequately staffed, with qualified programmer writers, editors, tools developers, release management, and builders. Typical programmer/writer-to-developer ratios at ArtPak vary from 8:1 to 13:1. The appropriate ratio depends on the complexity of the API, the timeframe available to document it, the desired quality, and the stability of the API.
- The "Wallaby" release cycle needs to be of sufficient length for writers and editors to have adequate time to do their work before lockdown.
- The code needs to be stable long enough for writers and editors to accurately document the shipping API behaviors.
- UE teams will need tech reviews from PM, dev, test, PSS, and architect subject matter experts. They will also need PM, dev, and test participation in planning and prioritizing documentation coverage.

- The “Wallaby” SDK will need to be allowed to drop documentation late enough to the product to ensure that the API reference documentation matches the shipping APIs.
- UE teams will need infrastructure and processes from the “Wallaby” SDK team, such as a GUI tool to enable documentation topics to be appropriately included in the TOC, a tool to track programming language coverage in code snippets, a tool and test infrastructure for snippet compilation testing, guidance and coordination of terminology issues, a set of reviewers and a review process for plans for programming language coverage in code snippets, and a tool for tracking and publishing progress towards goals, such as coverage of key scenarios, and tech review completion.
- UE contributors will need support from the “Wallaby” SDK team in their efforts to collaboratively create cross-company standards, such as topic templates, authoring guidelines, and schemas.

The “Wallaby” SDK team will need guidance from UE leadership bodies, such as the UART developer documentation content subgroup, on issues that affect documentation content standards

9 Related Documents

- Wallaby SDK Team site
- Development documentation: <<INSERT LINK>>
- Test matrix & documentation: <<INSERT LINK>>
- API documentation: <<INSERT LINK>>