Asciidcotor
サンプルファイル
(analog向け)

第1.0.0版

目次

1. 表現の統一について

1.1. 統一した文言など表現ゆれを防ぐための利用

define-{定義対象}.adocに定義ずみの項目については、本文中に直書きすることなく定義を参照して利用するようお願いします。
利用方法は「3.1, “文言や画像、リンク等の定義と利用”」に記載があります。
対象の定義ファイルは以下の通りです。

define-literal.adoc
: 表現ゆれを防いだり、記載を容易にするための共通的な表現を定義しています。
こちらもindex.adocに記載すると盆雑になるため外部定義となっており、編集中のプレビューもできるように、各ファイルの最上部で条件インクルードされています。

define-imageNicon.adoc
: アイコンやイメージの画像を定義しています。

1.2. 文体の整理

本ドキュメントでは、原則として"…です。…ます。"調の敬体（丁寧体）で記載し、常体（普通体）との混用は避けてください。 ただし、文の途中や、説明のための図の中、箇条書にする部分だけを常体にするなど状況に合わせて整理する必要があります。

その他、記載について迷う場合や統一的な記載を確認する際は以下を参照してください。 Microsoft ローカリゼーション スタイル ガイド

1.3. 画像について

画像はその対象やフォーマット毎のルールや、利用時のルールがあります。

利用する画像の品質を下げないでください

画像はドキュメントで見栄えの大きな要素を占めています。
このため、画像の品質が低下するとドキュメントの品質が低下したように感じられます。
このような印象はエンドユーザーが感じる印象だけでなく、記載に関わる各ステークホルダーにも影響します。
結果として「割れ窓理論」的な状況が生まれ、ドキュメント全体の品質を確保できなくなります。
もし、今後も記載の有無のラインを一定に保ち、表現ゆれを抑え、仕様網羅可能で、統一感のあるドキュメントを維持したいのであれば、画像の品質を落とすべきではありません。

1.4. セクションの切り方

「3.2, “セクションタイトルのレベルについて”」を参考に、他の画面と可能な限り統一感を保って表題をつけてください。

1.4.1. セクション内の記載方法

- 原則として手順として必要なものにカウンター機能を付ける
- 項目内の構成について「説明」「画像」の順で記載する
  (画面ページの最上部は概要がない場合「画像」が先行するが概要が上に入るものとする)
- 指摘事項で良く現れる()表記のタイトルやセクション表記は残さず文章になるように変更
  難しいものは見出しとしてそのままにする

1.5. コントロール名

画面のパーツやコントロールは以下の文言で統一します。

コントロール名称の一覧

利用する名称	説明	利用しない名称
ドロップダウンリスト	押すと選択肢がリストで表示されるもの	プルダウンリスト
ラジオボタン	主に丸いボタンで複数の選択肢の内一意で選択するもの	オプションボタン
チェックボックス	主にチェックいれる四角いボタンで複数の選択肢の内、複数を選択可能なもの	
コンボボックス	テキスト入力とドロップダウンリストが同一のコントロールで成り立つもの	
アコーディオン	表示や入力枠が閉じたり開いたりできるコントロール	

1.6. 文中のカッコの扱いについて

本画面の参照のうち、表外の項目については原則「項目名」を利用します。
【〇〇】ボタンのようにコントロール上に表記された項目名は選択肢を除き【項目名】を利用します。
（選択肢は現状バラツキあり）
その他の画面、ロールなど原則として用語定義を利用するものは「」を利用しません。

1.7. OR条件AND条件を日本語で記載する場合

1.7.1. OR条件

```
[α]または[β]
[A]、[B]、[C]、もしくは[D]等の[α]、または[E]、[F]、[G]、もしくは[H]などの[β]
```

は

```
α or β
α[A or B or C or D (or etc.)] or β[E or F or G or H (or etc.)]
```

となります。

1.7.2. AND条件

```
[α]および[β]
```

```
[A]、[B]、[C]、および[D]等の[α]、並びに[E]、[F]、[G]、および[H]などの[β]
```

は

```
α and β
α[A and B and C and D (and etc.)] and β[E and F and G and H (and etc.)]
```

となります。

1.8. 時刻フォーマットの表記

データの項目一覧のフォーマット表記や、スクリーンショット内を置き換える際の日時表記は以下で統一します。

書式指定子	説明	使用例	表記可否
d	月の日にち (1 ～ 31)	2009-06-01T13:45:30 → 1 2009-06-15T13:45:30 → 15	○
dd	月の日にち (01 ～ 31)	2009-06-01T13:45:30 → 01 2009-06-15T13:45:30 → 15	○
ddd	曜日の省略名	2009-06-15T13:45:30 → Mon (en-US)	-
h	12 時間形式の時間 (1 ～ 12)	2009-06-15T01:45:30 → 1 2009-06-15T13:45:30 → 1	-
hh	12 時間形式の時間 (01 ～ 12)	2009-06-15T01:45:30 → 01 2009-06-15T13:45:30 → 01	-
H	24 時間形式の時間 (0 ～ 23)	2009-06-15T01:45:30 → 1 2009-06-15T13:45:30 → 13	○
HH	24 時間形式の時間 (00 ～ 23)	2009-06-15T01:45:30 → 01 2009-06-15T13:45:30 → 13	○
m	分 (0 ～ 59)	2009-06-15T01:09:30 → 9 2009-06-15T13:29:30 → 29	○
mm	分 (00 ～ 59)	2009-06-15T01:09:30 → 09 2009-06-15T01:45:30 → 45	○
M	月 (1 ～ 12)	2009-06-15T13:45:30 → 6	○
MM	月 (01 ～ 12)	2009-06-15T13:45:30 → 06	○
MMM	月の省略名	2009-06-15T13:45:30 → Jun (en-US)	-
s	秒 (0 ～ 59)	2009-06-15T13:45:09 → 9	-
ss	秒 (00 ～ 59)	2009-06-15T13:45:09 → 09	○
tt	AM/PM 指定子	2009-06-15T13:45:30 → 午後 (ja-JP)	-
yy	年 (00 ～ 99)	2019-06-15T13:45:30 → 19	-
yyyy	年 (4 桁の数値)	2009-06-15T13:45:30 → 2009	○
YYYY	暦週の基準年	週の開始が月曜日 : 1997-12-29T01:09:30 → 1998 週の開始が日曜日 : 1998-01-03T13:45:30 → 1997	-
zzz	UTC を基準とする時間および 分単位のオフセット	2009-06-15T13:45:30-07:00 → -07:00	○

詳しくはMicrosoftの 「カスタム日時形式文字列」を参照してください。

2. AsciiDocについて

AsciiDocは軽量マークアップ言語の一つで可読文書記述形式です。
出力もHTML, PDF, ePubなどが行えるため出版や開発環境でのナレッジの共有まで様々な用途で利用することが可能です。

- AsciiDoc 公式サイト
- AsciiDocをドキュメント作成に利用している例
  - YAMAHA様事例
- AsciiDocを執筆に利用している例
  - O'Reilly Mediaの書籍の一部(Welcome kitとして公開していました)
- AsciiDocをナレッジに利用している例
  - GitHub(README&Wikiに利用するオープンソースマークアップライブラリに採用されています)

2.1. AsciiDocの記述方法

AsciiDocは軽量マークアップ言語でメジャーなフォーマットであるmarkdownと比較し、 同様の記述容易性を供えつつもより高度な文書の論理構造を構築することが可能です。
記載方法の簡単な説明は以下のリンクをご参照ください。

- Asciidoctor 文法クイックリファレンス(日本語訳)

また、ヘルプデスク内で利用している記載方法は「3, AsciiDocの記載方法」をご覧ください。

2.2. Asciidoctorの利用と環境

Asciidoctorは、AsciiDoc形式のための包括的なツールで、AsciiDoc形式をHTML5、DocBook、PDFなどの形式に変換します。

2.2.1. Asciidoctorの歴史

Asciidoctorは2012年にGitHub開発者であるNick HengeveldがGitのウェブサイトに示されているgitのマニュアルページを生成するために作成したプロトタイプをベースに、同じくGitHub開発者であるRyan Waldronがプロジェクトを開始しました。
2013年にAsciidoctorがGitHubにおいて発表され、その後はオープンソースプロジェクトとしてGitHubでホストされています。
このため、GitHubではサイト上で.adocファイルのプレビューすることができ、README&Wikiをを AsciiDoc形式で記述することが可能です。

- Asciidoctor 公式サイト

2.2.2. Asciidoctorの利用方法と本環境

詳しいマニュアルもありますが、本環境については以下のBlog記事をご参照ください。本環境はBlog記事

の構成を発展させたものとなります。

- 公式マニュアル
  - Asciidoctor Users manual
- Blog記事
  - AsciiDocを利用してエンジニアのナレッジを上手に集約しよう
  - GitHub Actionsを使って気軽にAsciiDocを出力しよう(1)
  - GitHub Actionsを使って気軽にAsciiDocを出力しよう(2)
  - GitHub ActionsのAsciiDoc用アクションの仕組み(1)
  - GitHub ActionsのAsciiDoc用アクションの仕組み(2)
  - GitHub Actionsで利用したいAsciidoctorの環境
  - GitHub Actionsで利用したいDockerコンテナの基礎
  - GitHub ActionsでAsciiDocをPDF出力しよう(1)
  - GitHub ActionsでAsciiDocをPDF出力しよう(2)

2.3. なぜAsciiDocを利用するのか？

2.3.1. バージョン管理の必要性

システムに紐づくドキュメントはシステムの更新と共に、更新され共存する必要があります。
また、お客様の不明点や、問題解決に必要なTipsの記載も欠かせません。
このため、いつ何のためにどんな修正が行われたか、Gitによるバージョン管理で追跡し、
必要であれば過去のバージョンを取り消すことも必要です。

2.3.2. スタイルの統一

従来のドキュメントエディタの多くは、更新が重なると適用したスタイルを維持することが難しくなります。
Asciidoc形式は記載方法がテキスト形式に限られるため、記載時にスタイルを切り離して記載することができます。
これにより、より大きな文章でも統一されたスタイルで提供することが可能です。

2.3.3. 表現の統一

従来のドキュメントエディタの多くは、文字列の定義や、繰り返し利用される記述の一括定義が行えず、
記載箇所毎の大量の表現ゆれ、表記ゆれに対応する必要がありました。
また、表記ゆれを確認するために検索を行うと大量の検索結果が表示されてしまい
あえて別表現を使っている場合の抽出が困難でした。
Asciidoc形式は文字列の定義や、文書の参照取り込みが可能なため、これらの問題に対し適切な運用が可能です。

2.3.4. 各権限毎の出力

全ての仕様を網羅し的確な説明がなされたナレッジがあれば、それから各権限向けのマニュアル等ドキュメントを作りたいと考えるのは自然な発想です。
従来のドキュメントエディタの多くでも、そのような運用がなされていますが、更新が進むに連れて対象権限毎に表現を修正したドキュメントと、
最新の仕様が記載された網羅的なドキュメントのマージが難しくなり、表現ゆれや記載ミスの原因となっています。
Asciidoc形式はifdefディレクティブの利用により簡易的な出力制御が行えるため、
網羅的なドキュメントに全ての権限に対する記載することができ、安定的に各権限のマニュアル等ドキュメントをリリースすることが可能です。

2.3.5. リリース向け出力形式の最適化

現在、多くのシステムではPDFのドキュメントが配布されますが、
WebシステムではHTMLのドキュメントも多用されています。
HTMLのドキュメントは目次もサイドバー等に表示可能になりブラウザの機能と合わせて目的検索生が向上するためです。
従来のドキュメントエディタもほとんどの場合でPDF出力が可能になっていますが、
HTMLでの出力を行うと、HTMLの利便性を享受できないことが多く、最悪の場合レイアウト崩れ等が発生することが多々あります。
Asciidoc形式はPDF出力とHTML出力に対して、それぞれ最適な出力を提供します。

2.4. AsciiDocに移行するデメリット

ここでは従来のドキュメントと、本ナレッジを比較した場合のデメリットを記載します。

2.4.1. 記載方法に慣れていない

今までOffice suite等のドキュメントエディタを利用していた場合、
多くの方が編集方法に戸惑うと考えられます。
ただしこれらの問題はドキュメントエディタが更新された場合にも当てはまります。
Asciidoc形式はMarkdown形式の様な簡易的な記載ができる学習コストが低い記載方法の一つです。
従来のドキュメントエディタと異なり、スタイルを壊してしまうこともないため大きな問題にならないと考えています。

2.4.2. 編集履歴や校閲機能がない

Asciidoc形式はテキスト形式のため、編集履歴や校閲機能はありません。
これらを同じドキュメントエディタ上で利用していた場合は当初不便を感じると思われます。
Asciidoc形式では、本文中に出力されないコメントを記載することは可能です。
また、より確実なバージョン管理を行うためにGit-hubのプライベートリポジトリを利用することで、
Issueの管理による議論や、それに紐付けた更新など、多彩なバージョン管理等を行うことでカバー可能です。

2.4.3. 記載中にイメージがわかない

Asciidoc形式はテキスト形式のため、従来のドキュメントエディタと異なり、記載中はスタイルが適用されません。
しかし、現在は無料のテキストエディタや統合開発環境に(Atom, VS Code, IntelliJ IDEA, Android studio)等にプラグインをインストールすることで、
簡単にプレビューを表示しながら編集することが可能です。

2.4.4. 特殊な記述ができない

Asciidoc形式は原則として技術本やドキュメントを記載するための汎用的な構成を重視されているため、
画像の上に画像を貼り付けたり、ある一箇所だけ特別なスタイルを変更することができません。
これらについては、更新が進むたびに表現方法がバラついてしまうことがなくなるため、
一長一短だと考えられます。

2.4.5. ドキュメントエディタで校閲や添削できない

Asciidoc形式は出力を行うとPDFやHTMLにて出力されるため、その状態での校閲や添削はできません。
特にPDFではページング制御とうがあるため、出力イメージ上で校閲や添削を行いたいと考えられます。
この点についてPDFは、Adobe Acrobat ReaderやAppleのプレビューアプリには注釈機能があるため、
直接の編集ではない校閲や添削は可能です。
また、HTMLについてもスクリーンショット等にマークアップすることが多くのブラウザやOSで可能になっているため、
編集者以外の校閲や添削に問題はありません。

3. AsciiDocの記載方法

3.1. 文言や画像、リンク等の定義と利用

Asciidocでは定義済み項目を本文中に直書きすることなく簡単に利用することができます。

define-literal.adocに定義された情報は「{name-role-sample}」のように記載することで「サンプルロール」と出力したり、 定義された画像として「{icon-sample-user}」のように記載することで「 

」と出力することができます。

```
define-literal.adocに定義された情報は「\{name-role-sample}」のように記載することで「{name-role-sample}」と出力したり、
定義された画像として「\{icon-sample-user}」のように記載することで「{icon-sample-user}」と出力することができます。
```

3.2. セクションタイトルのレベルについて

asciidocはヘッダー(見出し)の扱いが厳格な文書形式です。
また多くの人が同時に修正しやすくGitでのマージ等が低コストで行えるように、 レベル2のヘッダーごとにインクルードによるファイルの分割してください。
また、分割したファイルでは原則レベル3以降のヘッダーを利用し文書の構造化に努めてください。

```
[[include]]
= 参照先の章の名称
include::include.adoc[]
```

インクルードするファイルのレベルは「leveloffset」を利用して以下の記載をすると揃えることができます。

```
include::include.adoc[leveloffset=+2]
```

3.3. 出力設定について

AsciiDocではifdef(ifndef)ディレクティブを利用した簡易的な出力制御が可能です。
本ドキュメントでは主に以下の定義を利用して状況や対象者によって以下のように出力を切り替えています。

```
ifdef::backend-html5[]
HTMLのみの出力内容
endif::[]

```

```
ifndef::backend-pdf[]
PDFでは出力されない内容
endif::[]

ifndef::convert-for-readers[]
自動化した出力では利用しない内容
(「convert-for-writers」は本サンプルで定義している項目)
endif::[]

ifdef::convert-for-writers[]
明示的にendifを記載する場合。
(「convert-index」は本サンプルで定義している項目)
endif::convert-for-writers[]
```

3.4. 画像について

ここでは利用方法を記載します。

3.4.1. ブロックでの記述

ブロック画像

```
.ブロック画像
image::contents/images/Logo/analog-logo.svg[pageTop,{full-page}]
```

画像の指定には定義を利用したサイズルールがあります。
定義を利用することで、全てのサイズを統一することが可能です。

サイズの記載

対象	定義	備考
外部ドキュメントの貼り付け	full-width	センタリングも行います。
サムネイル	thumbnail	
フルサイズの画面幅	full-page	

対象	定義	備考
ダイアログのみの画像	dialog-page	

3.4.2. インラインでの記述

インライン {icon-h-write} で表示 


```
インライン {icon-h-write} で表示
image:contents/images/SvgBadges/badges/badge-user-analog-check.svg[]
```

3.5. 表の書き方

普段の利用でMarkdownより大きな利便性を感じる場面は表の扱いです。
ここでは本ドキュメントで利用している記載法を紹介します。

複雑な表の扱い

TH サイズ1	TH サイズ2	TH サイズ3
<で左寄せ R1C1	^で中央寄せ R1C2	>で右寄せ R1C3
hで個別のヘッダ	AsciiDoc記法で改行→	セル内で 改行
TD	AsciiDoc記法のリスト→	• <で左寄せを明示 • aでAsciiDoc記法を許容 • [.cellsfont]#内容#にてフォントをテーブルと合わせる
.n+縦に結合	TD 色の指定	TD 色の指定(継承)
	n+横に結合	
	TD 色の指定	↑ 結合されたカラムの定義はしません 色の指定は連続する

結合時など色の指定の影響(継承の扱いは微妙)

aで内部にテーブル

no.	内容
1	ただしセパレータを!にかえる

```
.複雑な表の扱い
[%header,cols="1,2,3"]
|===
|TH サイズ1 |TH サイズ2 |TH サイズ3

<|<で左寄せ R1C1
^|^で中央寄せ R1C2
>|>で右寄せ R1C3

h|hで個別のヘッダ
|AsciiDoc記法で改行→
|セル内で +
改行

```

```
|TD
|AsciiDoc記法のリスト→
<a|
* [.cellsfont]#<で左寄せを明示#
* [.cellsfont]#aでAsciiDoc記法を許容#
* [.cellsfont]#[.cellsfont]#内容#にてフォントをテーブルと合わせる#

.3+|.n+縦に結合 +
| {set:cellbgcolor:#00ccff} TD +
色の指定
| {set:cellbgcolor:#transparent} TD +
色の指定(継承)

2+|n+横に結合

| {set:cellbgcolor:#00ccff} TD +
色の指定
| ↑ 結合されたカラムの定義はしません +
色の指定は連続する

3+a| {set:cellbgcolor:#transparent}

[.cellsfont]#結合時など色の指定の影響(継承の扱いは微妙)#

[.cellsfont]#aで内部にテーブル#
[%header,cols="1,3"]
!===
![.cellsfont]#no.# ![.cellsfont]#内容#
^![.cellsfont]#1#
![.cellsfont]#ただしセパレータを\!にかえる#
!===
// 隙間が足りない事があるので隠し文字をいれる
{zwsp}
|===
//最終セルのカラーをリセットして他のテーブルに影響しない様にする
{set:cellbgcolor!}
```

3.6. ロール指定による文字のスタイル指定

以下のようにロールを利用してスタイルを適用できます。

- 指定なし
- **強調によるボールド**
- 大きめ文字
- 赤
- オレンジ大きめ文字
- **青の大きめ文字ボールド**

- テーブル内と同じ文字
- テーブル内の赤字

```
* 指定なし
* *強調によるボールド*
* [.big]#大きめ文字#
* [.red]#赤#
* [.darkorange.big]#オレンジ大きめ文字#
* [.navy.big]#*青の大きめ文字ボールド*#
* [.cellsfont]#テーブル内と同じ文字#
* [.cellsfont.red]#テーブル内の赤字#
```

3.6.1. 本ドキュメントでロール指定可能な色と色名について

本ドキュメントでは以下のブロックに示す色をHTMLおよびPDF共通で利用できるよう
に「css/docinfo.css」および「themes/default-role-color-theme.yml」に定義しています。
利用の際はSVGなども細かく色を合わせるとともに、ドキュメント全体での統一感を失わないように注意してください。
なお、色名のリストはPDFでは大きくなりすぎるので出力しません。

3.7. 特殊なAttributeの利用

3.7.1. リンクの定義

利用しているActionのリポジトリ

本ドキュメントは{link-site-preview}で閲覧できます。
{zwsp}は日本語との結合時に必要です。

```
link:https://github.com/Analog-inc/asciidoctor-
action[利用しているActionのリポジトリ]

本ドキュメントは{zwsp}{link-site-preview}で閲覧できます。 +
\{zwsp}は日本語との結合時に必要です。
```

3.7.2. 定義の途中で色指定

通常の記載は『【テスト】ボタン』で、一部のみ赤字は『【テスト】ボタン』です。 /がpassでそのまま出力されるために2行に分けています。

```
:name-button-test: 【テスト】ボタン
:name-button-test-red-name: pass:quotes[[.red]#【テスト】#]
:name-button-test-red: {name-button-test-red-name}{zwsp}ボタン

```

```
通常の記載は『{name-button-test}』で、一部のみ赤字は『{name-button-test-red}』です。
/{zwsp}がpassでそのまま出力されるために2行に分けています。
```

3.7.3. 定義内で改行

定義内で改行はしても 表示では改行しません。
定義内で改行して
表示でも改行します。

```
:sentence-no-newline: 定義内で改行はしても \
表示では改行しません。

:sentence-newline: 定義内で改行して + \
表示でも改行します。

{sentence-no-newline} +
{sentence-newline}
```

3.8. 脚注

脚注はアイコンが左側に表示されるスタイルです。 変更容易性や表現の幅を考慮しブロック形式で記載することをお勧めします。

NOTE

インフォメーションマークが出力されます。
補足となる内容や追加の情報が必要な場合に記載してください。

```
[NOTE]
.[.navy.big]#*NOTE*#
====
インフォメーションマークが出力されます。 +
補足となる内容を記載してください。
====
```

TIP

電球マークが出力されます。
少し高度な利用方法などを記載してください。
ユーザーが行う解決策などを提示する場合に利用してください。

```
[TIP]
.[.big]#*TIP*#
====
電球マークが出力されます。 +
```

```
少し高度な利用方法などを記載してください。
====
```

WARNING

三角オレンジの注意マークが出力されます
システムを利用する上で不用意にデータの閲覧ができなくなるなどの
問題を回避する方法を記載してください。

```
[WARNING]
.[.orange.big]#*WARNING*#
====
三角オレンジの注意マークが出力されます +
システムを利用する上で不用意にデータの閲覧ができなくなるなどの +
問題を回避する方法を記載してください。
====
```

IMPORTANT

丸い赤の警告マークが出力されます
システムを利用する上でセキュリティリスクなどを記載してください。

```
[IMPORTANT]
.[.red.big]#*IMPORTANT*#
====
丸い赤の警告マークが出力されます +
システムを利用する上でセキュリティリスクなどを記載してください。
====
```

CAUTION

現在ある不具合やバグ、明らかな矛盾点など記載してください。

```
[CAUTION]
.[.darkgreen.big]#*CAUTION*#
====
現在ある不具合やバグ、明らかな矛盾点など記載してください。
====
```

3.9. ブロック

3.9.1. Example block (例示)

〇〇について

例示的な内容を指定するか、
または勧告ブロックを定義します。

1. リスト
2. リスト

```
.〇〇について
[example]
====
例示的な内容を指定するか、 +
または勧告ブロックを定義します。

. リスト
. リスト
====
```

3.9.2. Listing block

1. 〇〇について

```
// 入力されたソースコードやキーボード入力が表示されます

ただし +
吹き出し機能が使えます。①

document.write("Hello world!!");
```

① 吹き出しの内容をここで定義します。

```
.〇〇について
[listing, javascript]
\----
// 入力されたソースコードやキーボード入力が表示されます

ただし +
吹き出し機能が使えます。①

document.write("Hello world!!");

\----
<1> 吹き出しの内容をここで定義します。
```

3.9.3. Fenced

2. ○○について

```
// 入力されたソースコードまたはキーボード入力が表示されます。
ドキュメントで有効になっていて、言語が設定されている場合は、 +
ソース・ハイライターで色付けされます。

document.write("Hello world!!");
```

```
.○○について
```
// 入力されたソースコードまたはキーボード入力が表示されます。
ドキュメントで有効になっていて、言語が設定されている場合は、 +
ソース・ハイライターで色付けされます。

document.write("Hello world!!");
```
```

3.9.4. Literal block

○○について

```
入力された通りに +
出力テキストが表示されます
吹き出し機能が使えます。①
```

① 吹き出しの内容をここで定義します。

```
 スペースを開けた一行が同じ表示になるので注意
```

```
.○○について
[literal]
....
入力された通りに +
出力テキストが表示されます
吹き出し機能が使えます。①
....
<1> 吹き出しの内容をここで定義します。

 スペースを開けた一行が同じ表示になるので注意
```

3.9.5. Passthrough block

出力に直接送られる +

未処理のコンテンツ

```
.〇〇について
[pass]
++++
出力に直接送られる +
未処理のコンテンツ
++++
```

3.9.6. Blockquotes(引用)

3. 〇〇について

ミスをしたことのない人は、 +
新しいことに挑戦したことがありません。
吹き出し機能が使えます。①

① 吹き出しの内容をここで定義します。

```
.〇〇について
[quote, Albert Einstein,proverb,]
\----
ミスをしたことのない人は、 +
新しいことに挑戦したことがありません。
吹き出し機能が使えます。①
\----
<1> 吹き出しの内容をここで定義します。
```

3.9.7. Sidebar

〇〇について

文書の流れの外に表示される
テキストやコンテンツを 非表示にする

1. リスト
2. リスト

```
.〇〇について
[sidebar]
****
文書の流れの外に表示される +
テキストやコンテンツを
非表示にする
```

```
7
8 . リスト
9 . リスト
10 ****
```

3.9.8. Source

4. 〇〇について

```
1 // 入力されたとおりに表示されるソースコードまたはキーボード入力。
2 ドキュメントで有効になっていて、言語が設定されている場合は、 +
3 ソース・ハイライターで色付けされます。
4 吹き出し機能が使えます。①
5
6 document.write("Hello world!!");
```

① 吹き出しの内容をここで定義します。

```
1 .〇〇について
2 [source,Java]
3 /----
4 // 入力されたとおりに表示されるソースコードまたはキーボード入力。
5 ドキュメントで有効になっていて、言語が設定されている場合は、 +
6 ソース・ハイライターで色付けされます。
7 吹き出し機能が使えます。①
8
9 document.write("Hello world!!");
10
11 /----
12 <1> 吹き出しの内容をここで定義します。
```

3.9.9. Stem

〇〇について

```
インタプリタに直接送られる未処理のコンテンツ（AsciiMath や LaTeX math など）
```

```
1 .〇〇について
2 [stem]
3 ++++
4 インタプリタに直接送られる未処理のコンテンツ（AsciiMath や LaTeX math など）
5 ++++
```

3.10. リスト

3.10.1. 順序無しリストのネスト

- レベル１
  - レベル２
    - レベル３
  - レベル２

```
* レベル１
** レベル２
*** レベル３
** レベル２
```

3.10.2. 順序ありリストのネスト

1. 手順１
2. 手順２
   a. 手順２a
   b. 手順２b
3. 手順３

```
. 手順１
. 手順２
.. 手順２a
.. 手順２b
. 手順３
```

3.10.3. チェックリスト

リーフレットなどを作成する場合に利用してください。

☑ チェック済みの要素

☑ これもチェック済みの要素

☐ 未チェックの要素

```
- [*] チェック済みの要素
- [x] これもチェック済みの要素
- [ ] 未チェックの要素
```

3.11. カウンター

カウンターは型を定義することができる。
1. カウンターの利用方法
2. 自動的にインクリメント
見せずにインクリメント
4. 飛んでるはず

1. リセットは普通のAttributeと同じ

a. カウンターは型を定義することができる。
b. インクリメントができればいい

```
{counter2:sequence-count-sample-no:0}カウンターは型を定義することができる。 +
{counter:sequence-count-sample-no}.{nbsp}カウンターの利用方法 +
{counter:sequence-count-sample-no}.{nbsp}自動的にインクリメント +
{counter2:sequence-count-sample-no}見せずにインクリメント +
{counter:sequence-count-sample-no}.{nbsp}飛んでるはず

:sequence-count-sample-no: 0

{counter:sequence-count-sample-no}.{nbsp}リセットは普通Attributeと同じ

{counter:sequence-count-sample-
alphabet:a}.{nbsp}カウンターは型を定義することができる。 +
{counter:sequence-count-sample-alphabet}.{nbsp}インクリメントができればいい
```

3.12. 改ページ

```
<<<
```

3.13. コメントアウト

3.13.1. インライン

```
//はコメントアウトになり出力されません。
```

3.13.2. ブロック

```
////
ここにコメント
```

```
////
```

3.14. ラベル

辞書形式に利用します。

第一項
: 第一項の定義

  二条
  : 第一項二条の定義

第二項
: 第二項の定義

```
1 第一項:: 第一項の定義
2 第一項二条::: 第一項二条の定義
3 第二項:: 第二項の定義
```

3.15. 水平罫線

水平罫線は以下のように記載します。

```
1 '''
```

3.16. インラインマクロ

3.16.1. Inline image macro

「image:」で始まるインライン画像はインラインマクロの一つです。
詳しくは「3.4.2, “インラインでの記述”」を参照してください。

3.16.2. Inline pass macro

「pass:」を利用するとAsciiDocの文法などが無視され未処理でそのまま出力されます。

3.16.3. Inline Macro Processor Extension

Inline Macro Processor Extensionは自分で処理を記載して利用するAsciidoctor Extensionsの1種類です。

3.16.3.1. datetimeformat

現在はAsciiDocで定義されたAttributesのみ対応しています。

本ドキュメントは2021/01/01版です。
2024年08月09日
2024-08-09
本ドキュメントは 2024年08月09日 13時02分47秒 に更新されました。
2024年08月09日 13時02分
13時02分
2024/08/09 13:02:47
2024/08/09 13:02
13:02
2024-08-09T13:03:03+09:00
13時02分47秒
13:03:03
Fri, 09 Aug 2024 04:02:47 GMT
2024-08-09T13:02:47+09:00
Fri, 09 Aug 2024 13:02:47 +0900
Fri, 09 Aug 2024 13:02:47 +0900
2024-08-09T13:02:47+09:00
08/09/24 13:02:47 JST

```
本ドキュメントは{zwsp}datetimeformat:revdate[/]{zwsp}版です。 +
datetimeformat:docdate[年月日] +
datetimeformat:localdate[-] +
本ドキュメントは
datetimeformat:docdatetime[年月日時分秒]
に更新されました。 +
datetimeformat:docdatetime[年月日時分] +
datetimeformat:docdatetime[時分] +
datetimeformat:docdatetime[/:ss] +
datetimeformat:docdatetime[/:] +
datetimeformat:docdatetime[:] +
datetimeformat:localdatetime[T] +
datetimeformat:doctime[時分秒] +
datetimeformat:localtime[:ss] +
datetimeformat:docdatetime[httpdate] +
datetimeformat:docdatetime[iso8601] +
datetimeformat:docdatetime[rfc2822] +
datetimeformat:docdatetime[rfc822] +
datetimeformat:docdatetime[xmlschema] +
datetimeformat:docdatetime[%D %T %Z]
```

3.17. 記号の記載

一部の記号は記載時にリプレースやエスケープを行う必要があります。

例えば「|」はリプレースとして「{vbar}」と記載を行います。 また、テーブル内で別のセパレーターとして「!」と利用する場合などはエスケープとして「/!」とすることで出力できます。

```
1 例えば「{vbar}」はリプレースとして「\{vbar}」と記載を行います。
2 また、テーブル内で別のセパレーターとして「!」と利用する場合などはエスケープとして「/
  !」とすることで出力できます。
```

以下はリプレースを行うべき記号や文字のリストとなります。

リプレース対象の定義済み文字 [1][2][3]

Attribute name	Replacement text	Appearance
blank	nothing	
empty	nothing	
sp	single space	
nbsp		
zwsp[4]	&#8203;	
wj[5]	&#8288;	
apos	&#39;	'
quot	&#34;	"
lsquo	&#8216;	‘
rsquo	&#8217;	’
ldquo	&#8220;	“
rdquo	&#8221;	”
deg	&#176;	°
plus	&#43;	+
brvbar	&#166;	¦
vbar	|	|
amp	&	&
lt	<	<
gt	>	>
startsb	[	[
endsb	]	]
caret	^	^
asterisk	*	*
tilde	~	~
backslash	\	\
backtick	`	`
two-colons	::	::
two-semicolons	;;	;;
cpp	C++	C++

[1] いくつかの置換はUnicode文字ですが、他の置換は数値の文字参照です(例えば、")。 これらの文字参照は、Unicode文字の使用がAsciiDocの構文に干渉したり、レンダラー(すなわちブラウザ)を混乱させる可能性がある場合に使用されます。 参照をレンダラーが理解できるものに変換するのはコンバータ次第です(マニュアルページとPDFコンバータの両方が処理するもの)。

[2] Asciidoctor は、定義済みの属性を再割り当てすることを妨げるものではありません。 しかし、出力形式が異なるエンコーディング方式の使用を必要としない限り、それらを読み取り専用として扱うのが最善です。 これらの属性は、コンテンツとプレゼンテーションを切り離すための効果的なツー

ルです。

[3] Asciidoctorでは、HTMLで定義されている名前付き文字参照(別名：名前付き実体)のいずれかを使用できます。 しかし、名前付き文字参照を使用すると、PDFなどの非HTML出力を生成する際に問題が発生することがあります。 これらの名前を解決するために必要な参照テーブルが定義されていない可能性があるからです。私たちの推奨事項は、XMLで定義されているもの(すなわち lt ・ gt ・ amp ・ quot ・ apos)を除いて、名前付き文字参照の使用を避けることです。 その代わりに、数値の文字参照を使用してください。

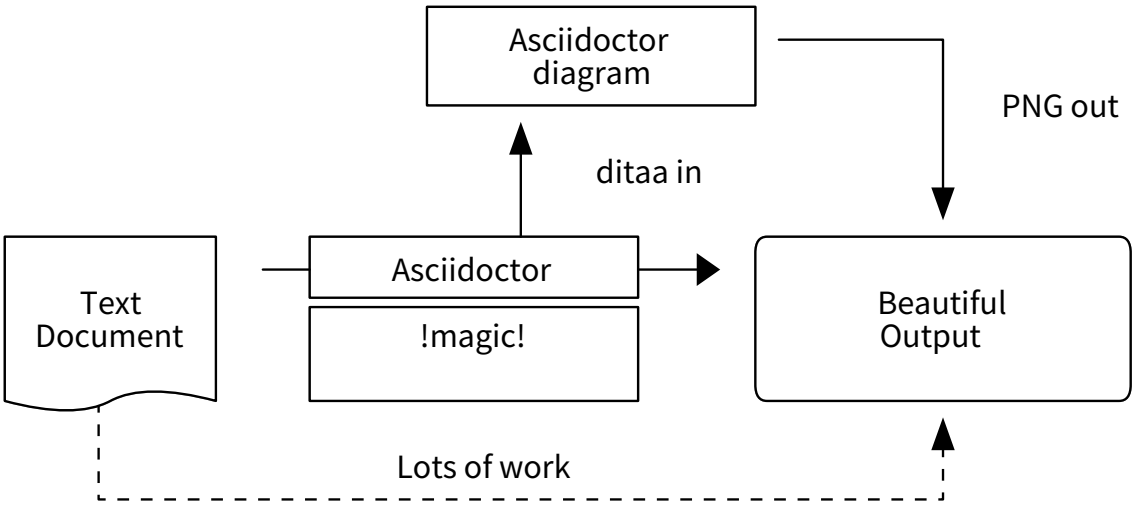
[4] ゼロ幅空間(ZWSP)は、必要に応じて長い単語を分割できる場所を示すUnicodeのコードポイントです。

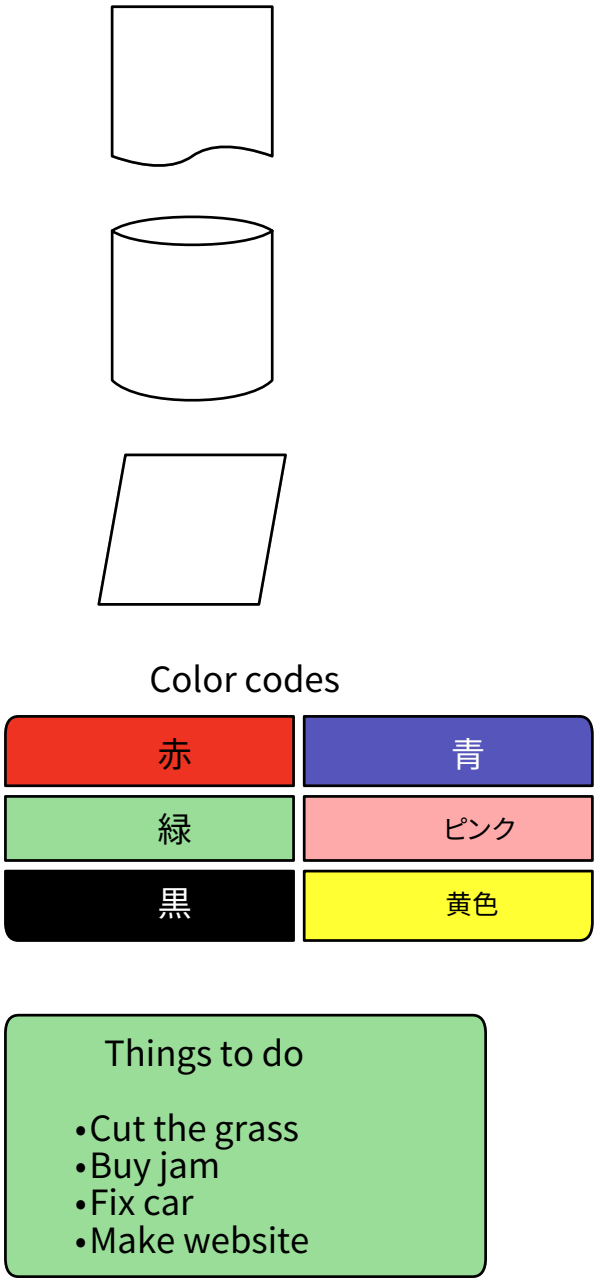
[5] ワードジョイナー(WJ)は、その位置での改行を防ぐUnicodeのコードポイントです。

3.18. ダイアグラム

Asciidoctorではダイアグラムを記載できますが、生成画像はプレビューできません。
ダイアグラムの名称は必ず一意のものを設定してください。

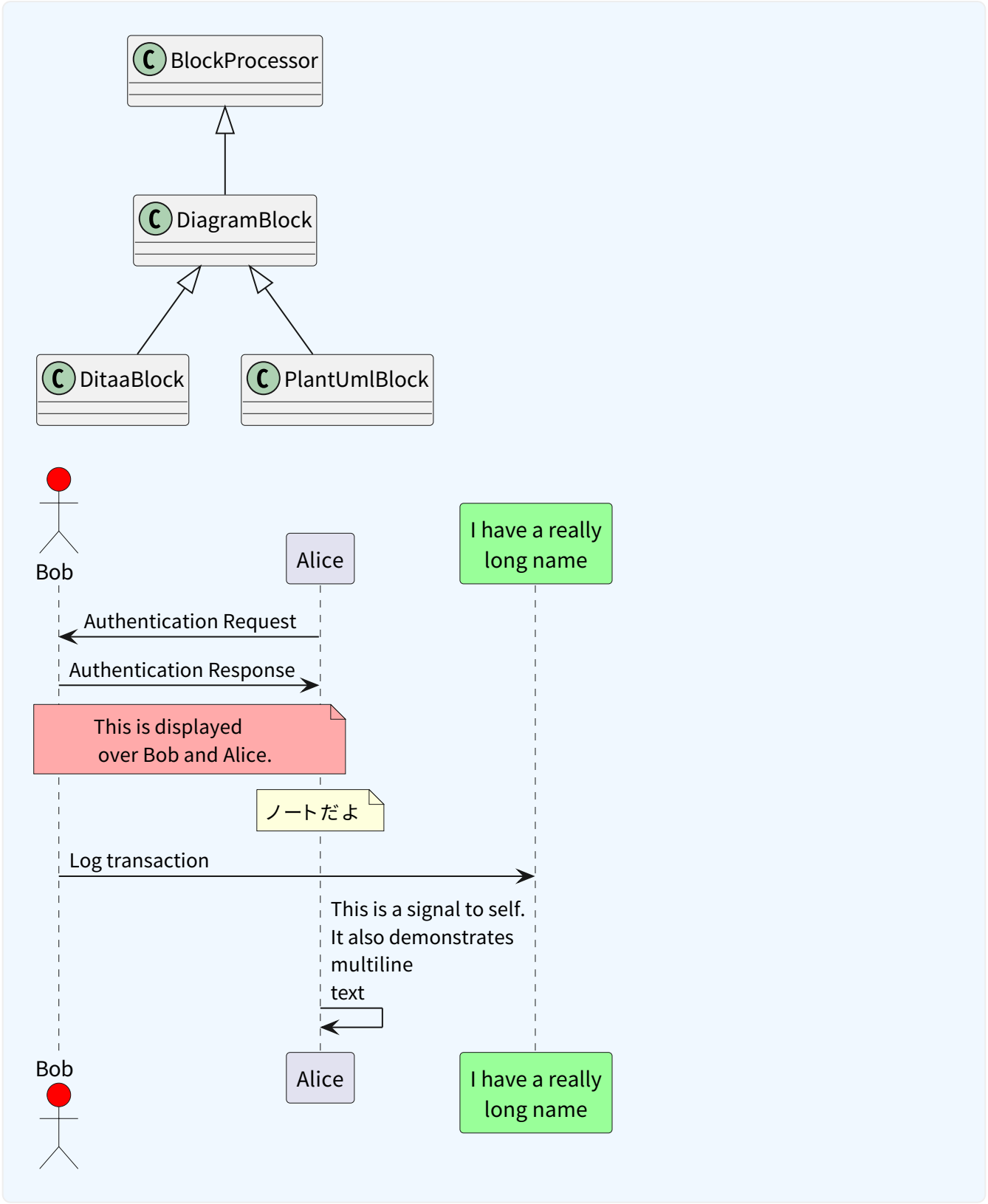
3.18.1. ditaaで記述するダイアグラム





3.18.2. PlantUMLで記述するダイアグラム

UMLを記述する場合はPlantUMLがおすすめ。
多彩なUMLを気軽に記述できる。


BlockProcessor
DiagramBlock
DitaaBlock
PlantUmlBlock
Bob
Alice
I have a really long name
Authentication Request
Authentication Response
This is displayed over Bob and Alice.
ノートだよ
Log transaction
This is a signal to self.
It also demonstrates multiline
text
Bob
Alice
I have a really long name

4. 本ドキュメントの設定について

4.1. 参照ディレクトリの扱い

本ドキュメントはできる限り VS Code 上でファイルを編集している時のリアルタイムプレビューと、Github Actions を利用したコンバートの両立を行えるようにしています。 その中でいくつかの対応が必要になったので記載します。

4.1.1. Asciidoc ファイルのファイル名の扱い

基本的に .adoc 拡張子のみを対応しています。 またrootディレクトリ直下の index- プリフィックスのみコンバート処理を実施します。 さらに README.adoc はコンバート処理を考慮しない様にしています。 そして、define- プリフィックスのものは特に注意が必要です。 この要件は以下の記載を上記以外のファイルに行うことで、記載しやすくしています。

```
1 == タイトル
2 :includedir: ../../
3 include::{includedir}contents/defines/define-all-include.adoc[]
4 ifeval::[{env} == vscode]
5 ifndef::imagesdir[]
6 :imagesdir: ../../
7 endif::imagesdir[]
8 endif::[]
9
10 本文
```

① ここでのパスはプロジェクト(ドキュメント)ルートへの相対パスを設定

これにより画像は全てプロジェクト(ドキュメント)ルートからのフルパス(VS codeでファイル右クリックで相対パスってとったパス)を指定することでPDFもHTMLも問題なく出力できます。 一方で、includeは {homedir} てつけるとフルパス、自分同じ場所か配下なら {homedir} をつけずに相対パスにして扱うことができます。

4.1.2. ディレクトリ名を指定する属性の話

includedir

この指定は本ドキュメント固有のCustom Document Attributeです。この指定を利用することでincludeを指定しやすくしています。
本指定をindexは空、個別のファイルを相対パスで行うことで、複雑なディレクトリ構成でのincludeの指定を容易にしています。
この定義の名称は Asciidocterの公式ドキュメントのInclude file resolutionを参考にしています。

basedir

Asciidocに定義されたBuilt-In Document Attributeです。この定義はCLIで動作する際にフルパスで値が設定されるため利用していません。

imagesdir

asciidocに定義された default asset directory Attributeです。この定義はHTML(:data-uri:のしては除く) の場合と画像を埋め込みするPDFで扱いが変わるため、CLIで指定します。画像へのリンクは(ドキュメント)ルートからのフルパスを利用することになります。

iconsdir

asciidocに定義された default asset directory Attributeです。現在は利用していませんし、動作の解析もしていません。

stylesdir

asciidocに定義された default asset directory Attributeです。この定義はおそらくHTMLの場合のみ利用されますが、現在は利用していませんし、動作の解析もしていません。

scriptsdir

asciidocに定義された default asset directory Attributeです。この定義はおそらくHTMLの場合のみ利用されますが、現在は利用していませんし、動作の解析もしていません。