ADOBE® INDESIGN® CS6

ADOBE INDESIGN CS6
スクリプティングガイド：
APPLESCRIPT

Adobe

Adobe® InDesign® CS6 スクリプティングガイド：AppleScript

ドキュメントの更新ステータス （ドキュメント全体。各章の更新ステータスについては、それぞれの章を参照してください）		
CS6	更新	ドキュメント全体で CS5 から CS6 への変更およびバージョン 7.0 から 8.0 への変更。

目次

1 はじめに

章の更新ステータス	
CS6	変更なし

このマニュアルでは、次の作業を行う方法について説明します。

- Adobe® InDesign® のスクリプティング環境の操作
- 高度なスクリプティング機能の使用
- ドキュメントに関連する基本的な作業（マススプレッドの設定、プリント、書き出しなど）
- ページアイテム（長方形、楕円形、グラフィックの線、多角形、テキストフレーム、グループ）の操作
- InDesign ドキュメントのテキストや書式の操作（テキストの検索と変更など）
- ダイアログボックスなどのユーザーインターフェイスアイテムの作成
- メニューのカスタマイズや追加と、メニューアクションの作成
- ユーザーインターフェイスのイベントに対する応答
- XML の操作（XML 要素の作成、XML の読み込み、レイアウトへの XML 要素の追加など）
- XML ルール（InDesign で XML を高速かつ簡単に操作できる新しいスクリプティング機能）の適用

ここでは、読者が『Adobe InDesign スクリプティングチュートリアル』を既に読んでおり、スクリプトを作成、インストール、実行する方法を理解しているものとして説明を行います。スクリプト環境に接続したり、スクリプトエディターから InDesign スクリプトオブジェクトモデルを表示したりする方法については、『Adobe InDesign スクリプティングチュートリアル』を参照してください。

このマニュアルに掲載されているスクリプトの使用方法

このマニュアルに掲載されているスクリプトのほとんどは、完全なスクリプトではありません。このマニュアルの説明内容に関連するスクリプトから特定部分のみを抜粋した、断片的なスクリプトです。このマニュアルに掲載されているスクリプト行をコピーして、スクリプトエディタに貼り付けることができますが、スクリプトを実行するには、さらに編集が必要になります。また、このマニュアルからコピーしたスクリプトには、（マニュアルのレイアウトにより）改行などの文字が含まれている可能性があることに注意してください。このような文字が含まれていると、スクリプトを正常に実行できません。

このマニュアルに掲載されているすべてのスクリプトの Zip アーカイブは、InDesign のスクリプティングホームページ（http://www.adobe.com/jp/products/indesign/scripting/index.html）で入手できます。アーカイブをダウンロードして展開したら、該当するスクリプト言語に対応するフォルダーを、InDesign フォルダー内の Scripts フォルダーの下にある Scripts Panel フォルダーに移動します。これで、InDesign のスクリプトパネルからスクリプトを実行できます。

スクリプトの構造について

サンプルスクリプトはすべて、main、mySetup、mySnippet、myTeardown（ハンドラー）などの共通のテンプレートを使用して記述されています。この記述方式は、自動化テストとパブリケーションを簡略化するためのものです。通常はスクリプトをこのように構築する必要はありません。スクリプト作成者の関心を引くと思われるスクリプトの断片は、mySnippet ハンドラー内に記述されています。

詳細情報

InDesign スクリプティングに関するユーザーフォーラム（http://www.adobeforums.com）でも、InDesign スクリプティングに関する情報が入手できます。このフォーラムでは、質問をしたり、質問に答えたり、新たに作成したスクリプトを別のユーザーと共有したりできます。このフォーラムでは、多数のサンプルスクリプトが提供されています。

2 スクリプティング機能

章の更新ステータス		
CS6	更新	12 ページの「スクリプトのバージョン対応」および 3 つの副節を、更新、修正および明確化しました。

この章では、InDesign のスクリプティング環境に関連するスクリプティングテクニックを紹介します。InDesign のスクリプトモデルを構成するオブジェクトのほとんどは、アプリケーションのデフォルトやドキュメントを操作するためのオブジェクトですが、この章ではまず、スクリプトそのものを制御する方法について説明します。

この章の内容は、次のとおりです。

- `script preferences` オブジェクトとそのプロパティ
- 実行中のスクリプトへの参照の取得
- 以前のバージョンのスクリプトオブジェクトモデルに基づいたスクリプトの実行
- `do script` メソッドを使用したスクリプトの実行
- スクリプトラベルの操作
- InDesign の起動時におけるスクリプトの自動実行

ここでは、読者が『Adobe InDesign スクリプティングチュートリアル』を既に読んでおり、該当するスクリプト言語で InDesign のスクリプトを作成、インストール、実行する方法を理解しているものとして説明を行います。

スクリプトの環境設定

`script preferences` オブジェクトには、InDesign でのスクリプトの実行方法に関連するオブジェクトやプロパティが用意されています。次の表に、script preferences オブジェクトの各プロパティの詳細を示します。

プロパティ	説明
`enable redraw`	スクリプトパネルからスクリプトを実行している最中の画面の再描画をオンまたはオフにします。
`scripts folder`	Scripts フォルダーへのパス。
`scripts list`	使用可能なスクリプトのリスト。このプロパティは、次に示すような配列の配列を返します。 `[[fileName, filePath], ...]` *`fileName`* はスクリプトファイルの名前を表し、*`filePath`* はスクリプトへのフルパスを表します。この機能を使用すれば、目的のスクリプトがインストールされているどうかを確認できます。

プロパティ	説明
user interaction level	このプロパティは、ユーザーに表示する警告やダイアログボックスを制御します。このプロパティを never interact に設定すると、警告やダイアログボックスは一切表示されなくなります。interact with alerts に設定すると、警告は有効になりますが、ダイアログボックスは無効になります。interact with all に設定すると、警告とダイアログボックスは通常どおりに表示されます。警告の表示をオフにする機能は、スクリプトでドキュメントを開く場合に便利です。InDesign では、フォントやリンク先のグラフィックファイルが見つからないと警告が表示されることがあります。このプロパティを never interact に設定してからドキュメントを開き、スクリプトを終了する直前にプロパティを元に戻せば（interact with all に設定すれば）、警告が表示されるのを避けられます。
version	使用中のスクリプティング環境のバージョン。詳しくは、12 ページの「スクリプトのバージョン対応」を参照してください。このプロパティは、アプリケーションのバージョンと同じではないことに注意してください。

現在のスクリプトの取得

現在のスクリプトへの参照を取得するには、アプリケーションオブジェクトの active script プロパティを使用します。このプロパティは、現在のスクリプトが保存されている場所を基準にしてファイルやフォルダーを検索する場合に役立ちます。次のスクリプトにその例を示します（チュートリアルスクリプトの ActiveScript より）。

```
tell application "Adobe InDesign CS6"
    set myScript to active script
    display dialog ("The current script is: " & myScript)
    tell application "Finder"
        set myFile to file myScript
        set myParentFolder to container of myFile
    end tell
    display dialog ("The folder containing the active script is: " & myParentFolder)
end tell
```

スクリプトエディタでスクリプトをデバッグしている場合は、active script プロパティを取得しようとするとエラーが発生します。active script プロパティで返されるのは、スクリプトパレットから実行したスクリプトのみです。

スクリプトのバージョン対応

InDesign では、以前のバージョンの InDesign スクリプトオブジェクトモデルに基づいて作成されたスクリプトを実行できます。新しいバージョンの InDesign で古いスクリプトを実行する場合は、次の点に注意する必要があります。

- ターゲットの設定 - スクリプトを実行する InDesign のバージョン（現在のバージョン）をターゲットに設定する必要があります。13 ページの「ターゲットの設定」に説明するとおり、ターゲットの設定方法は言語によって異なります。
- コンパイル - この段階では、スクリプトで使用されている名前が InDesign で認識できるスクリプト ID にマッピングされます。13 ページの「コンパイル」に説明するとおり、コンパイルの方法は言語によって異なります。
- インタープリターによる実行 - この段階では、スクリプト ID が InDesign の適切なハンドラーに関連付けられます。これにより InDesign は、以前のバージョンのスクリプトオブジェクトモデルに基づいて作成されたスクリプトも正しく解釈できるようになります。そのためには、スクリプトでアプリケーションのスクリプト環境設定に古いオブジェクトモデルを明示的に設定するか（14 ページの「インタープリ

ターによる実行」に示すとおり）、次のようにスクリプトパネル用のフォルダーからスクリプトを実行します。

フォルダー	スクリプトの InDesign バージョン
`Version 8.0 Scripts`	CS6
`Version 7.0 Scripts`	CS5 および CS5.5
`Version 6.0 Scripts`	CS4
`Version 5.0 Scripts`	CS3
`Version 2.0 Scripts`	CS2

ターゲットの設定

スクリプトを実行する InDesign のバージョン（現在のバージョン）が、スクリプトでターゲットとして明示的または暗黙的に設定されている必要があります。スクリプトパネルからスクリプトを実行すると暗黙的にターゲットが設定されます。

AppleScript で明示的にターゲットを設定するには、`tell` 文を使用します。

```
--Target InDesign CS6
tell application "Adobe InDesign CS6"
```

コンパイル

通常は、ターゲットとして設定したアプリケーションの用語説明を使用してコンパイルが実行されますが、`using terms from` 文を使用すれば、別のバージョンの InDesign の用語説明を使用してコンパイルを実行できます。例えば、CS6（バージョン 8.0）内で CS5 スクリプト（バージョン 7.0）を使用するには、次のようにします。

```
tell application "Adobe InDesign CS6" --target CS6
    using terms from application "InDesign CS5" --compile using CS5
        --InDesign CS5 version script goes here.
    end using terms from
end tell
```

これを動作させるには、application オブジェクトの `publish terminology` コマンドを使用して、CS5 バージョンの AppleScript 用語説明を生成する必要があります。これによって、アプリケーションの環境設定フォルダー内の `Scripting Support` フォルダーにある、DOM のバージョンと同じ名前のフォルダーに用語説明が生成されます。例えば、CS5 用語説明を `/ ユーザ /< ユーザー名 >/ ライブラリ /Preferences/Adobe InDesign/Version 8.0-J/ja_JP/Scripting Support/8.0` フォルダーに生成するには、次のようにします。

```
tell application "Adobe InDesign CS6"
    --publish the InDesign CS5 dictionary (version 7.0 DOM)
    publish terminology version 7.0
end tell
```

インタープリターによる実行

インタプリタで使用するスクリプトオブジェクトモデルのバージョンを取得したり設定したりするには、InDesign アプリケーションオブジェクトの `script preferences` オブジェクトを使用します。デフォルトでは、アプリケーションの現在のバージョンに設定されており、変更されるまでそのバージョンが使用されます。

例えば、スクリプトオブジェクトモデルのバージョンを CS5（7.0）に変更するには、次のようにします。

```
--Set to 7.0 scripting object model
set version of script preferences to 7.0
```

do script メソッドの使用

`do script` メソッドを使用すると、スクリプトの中で別のスクリプトを実行することができます。このメソッドには、有効なスクリプトコードを表す文字列か、ディスク上のスクリプトファイルを指定できます。指定するスクリプトは、現在のスクリプトと同じ言語であっても、異なる言語であってもかまいません。使用できる言語はプラットフォームによって異なります。Mac OS® の場合は AppleScript または JavaScript を実行できます。Windows® の場合は VBScript または JavaScript を実行できます。

`do script` メソッドは、次の様々な場合に利用できます。

- 使用したい機能が、メインのスクリプト言語ではサポートされておらず、別の言語でサポートされている場合。例えば、VBScript にはファイルブラウザーやフォルダーブラウザーを表示する機能はありませんが、JavaScript にはこの機能が用意されています。AppleScript では三角関数（サインやコサイン）の計算に時間がかかる場合がありますが、JavaScript では高速に処理できます。JavaScript では、Microsoft® Excel スプレッドシートのセルの内容を照会することはできませんが、AppleScript や VBScript を使用すれば行うことができます。どの場合も、別の言語で断片的なスクリプトコードを作成して `do script` メソッドで実行すれば、メインのスクリプト言語の制約を回避することができます。
- スクリプトを動的に生成して実行する場合。実行時にスクリプトの文字列を動的に生成して、`do script` メソッドで実行することができます。この方法は、選択されている内容や作成するオブジェクトの属性に基づいてダイアログボックスやパネルをカスタマイズしたい場合などに適しています。
- オブジェクトにスクリプトを埋め込む場合。`do script` メソッドを使用すれば、オブジェクトの `label` プロパティに文字列として保存されているスクリプトを実行することができます。このテクニックを利用すると、特定のパラメーターに基づいてレイアウトプロパティを制御したり内容を更新したりするスクリプトを、オブジェクト自体に埋め込むことができます。また、XML 要素にも、属性またはコンテンツとしてスクリプトを埋め込むことができます。18 ページの「起動時におけるスクリプトの自動実行」を参照してください。

do script へのパラメーターの渡し方

`do script` で実行するスクリプトにパラメーターを渡すには、次のようにします（チュートリアルスクリプトの DoScriptParameters より）。

```
tell application "Adobe InDesign CS6"
    --Create a list of parameters.
    set myParameters to {"Hello from do script", "Your message here."}
    --Create a JavaScript as a string.
    set myJavaScript to "alert(¥"First argument: ¥" + arguments[0] +
    ¥"¥¥rSecond argument: ¥" + arguments[1])"
    --Run the JavaScript using the do script command.
    do script myJavaScript language javascript with arguments myParameters
    --Create an AppleScript as a string.
    set myAppleScript to "tell application ¥"Adobe InDesign CS6¥"" & return
    set myAppleScript to myAppleScript & "display dialog (¥"First argument: ¥" &
    item 1 of arguments & return & ¥"Second argument: ¥" & item 2 of arguments)" & return
    set myAppleScript to myAppleScript & "end tell"
    --Run the AppleScript using the do script command.
    do script myAppleScript language applescript language with arguments myParameters
end tell
```

do script からの戻り値

次のスクリプトは、do script で実行されるスクリプトから値を返す方法を示しています。この例では、文字列として実行される JavaScript を使用していますが、スクリプトファイルも同様に使用できます。また、この例で返す値は 1 つですが、配列を使用すると複数の値を返すことができます（完全なスクリプトについては、DoScriptReturnValues スクリプトを参照してください）。

```
set myDocument to document 1
set myPage to page 1 of myDocument
set myTextFrame to text frame 1 of myPage
tell myDocument
    set myDestinationPage to make page
end tell
set myPageIndex to name of myDestinationPage
set myID to id of myTextFrame
set myJavaScript to "var myDestinationPage = arguments[1];" & return
set myJavaScript to myJavaScript & "myID = arguments[0];" & return
set myJavaScript to myJavaScript & "var myX = arguments[2];" & return
set myJavaScript to myJavaScript & "var myY = arguments[3];" & return
set myJavaScript to myJavaScript & "var myPageItem =
app.documents.item(0).pages.item(0).pageItems.itemByID(myID);" & return
set myJavaScript to myJavaScript &
"myPageItem.duplicate(app.documents.item(0).pages.item(myDestinationPage));" & return
--Create an array for the parameters we want to pass to the JavaScript.
set myArguments to {myID, myPageIndex, 0, 0}
set myDuplicate to do script myJavaScript language javascript with arguments myArguments
--myDuplicate now contains a reference to the duplicated text frame.
--Change the text in the duplicated text frame.
set contents of myDuplicate to "Duplicated text frame."
```

別の方法として、アプリケーションの script args オブジェクトを使用して別のスクリプトから値を取得する方法があります（この名前は、「script arguments」を縮めたものです）。次のスクリプトにその例を示します（完全なスクリプトについては、DoScriptScriptArgs を参照してください）。

```
tell application "Adobe InDesign CS6"
    --Create a string to be run as an AppleScript.
    set myAppleScript to "tell application ¥"Adobe InDesign CS6¥"" & return
    set myAppleScript to myAppleScript & "tell script args" & return
    set myAppleScript to myAppleScript & "set value name ¥"ScriptArgumentA¥"
    value ¥"This is the first AppleScript script argument value.¥"" & return
    set myAppleScript to myAppleScript & "set value name ¥"ScriptArgumentB¥"
    value ¥"This is the second AppleScript script argument value.¥"" & return
    set myAppleScript to myAppleScript & "end tell" & return
    set myAppleScript to myAppleScript & "end tell"
    --Run the AppleScript string.
    do script myAppleScript language applescript language
    --Retrieve the script argument values set by the script.
    tell script args
        set myScriptArgumentA to get value name "ScriptArgumentA"
        set myScriptArgumentB to get value name "ScriptArgumentB"
    end tell
    --Display the script argument values in a dialog box.
    display dialog "ScriptArgumentA: " & myScriptArgumentA & return & "ScriptArgumentB: " &
myScriptArgumentB
    --Create a string to be run as a JavaScript.
    set myJavaScript to "app.scriptArgs.setValue(¥"ScriptArgumentA¥", ¥"This is the first
JavaScript script argument value.¥");" & return
    set myJavaScript to myJavaScript & "app.scriptArgs.setValue(¥"ScriptArgumentB¥", ¥"This is
the second JavaScript script argument value.¥");" & return
    --Run the JavaScript string.
    do script myJavaScript language javascript
    --Retrieve the script argument values set by the script.
    tell script args
        set myScriptArgumentA to get value name "ScriptArgumentA"
        set myScriptArgumentB to get value name "ScriptArgumentB"
    end tell
    --Display the script argument values in a dialog box.
    display dialog "ScriptArgumentA: " & myScriptArgumentA & return & "ScriptArgumentB: " &
myScriptArgumentB
end tell
```

do script を使用した取り消しの制御

InDesign では、ほとんどすべてのアクションを取り消すことができます。ただし、行ったアクションのほとんどすべてがディスクに書き出されるので、犠牲も伴います。ユーザーインターフェイスのツールを使用した通常の作業では、取り消しによって問題が発生することはありません。しかし、短時間に何千ものアクションの実行が可能であるスクリプトの場合、ディスクへのアクセスが絶えず繰り返されているので、パフォーマンスに深刻な影響を与える場合があります。

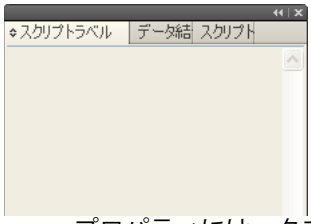
`do script` コマンドには、このパフォーマンスの問題を解決するために 2 つのパラメーターが用意されており、InDesign の取り消し動作に関してスクリプトの実行方法を制御することができます。次の例で、これらのパラメーターを示します。

```
--Given a script "myAppleScript" and an array of parameters "myParameters"...
tell application "Adobe InDesign CS6"
do script myJavaScript language javascript with arguments myArguments undo mode fast entire
script undo name "Script Action"
end tell
--undo modes can be:
--auto unto: Add no events to the Undo queue.
--entire script: Put a single event in the Undo queue.
--fast entire script: Put a single event in the Undo queue.
--script request: Undo each script action as a separate event.
--The last parameter is the text that appears in the Undo menu item.
```

スクリプトラベルの操作

InDesign の多くのスクリプトオブジェクトには、label プロパティが用意されています。例えば、ページアイテム（長方形、楕円形、グループ、多角形、テキストフレーム、グラフィックの線）、テーブルのセル、ドキュメント、ストーリー、ページなどのオブジェクトでは、label プロパティが使用できます。このプロパティには、大量のテキストを保存することができます。

ページアイテムのラベルは、スクリプトラベルパネルを使用して表示、入力、編集できます（次の図を参照。このパネルを表示するには、ウィンドウ／ユーティリティ／スクリプトラベルを選択します）。また、スクリプトを使用して、オブジェクトにラベルを追加したり、スクリプトのラベルを読み取ったりすることもできます。ストーリー、ページ、段落スタイルなどの多くのオブジェクトは、スクリプトラベルパネルを使用してラベルを設定したり表示したりすることはできません。



label プロパティには、タブ区切りやカンマ区切りのテキスト、HTML、XML など、あらゆる形式のテキストデータを含めることができます。スクリプトもテキストであるため、label プロパティに保存することができます。

名前を持つアイテム（段落スタイル、カラー、レイヤーなど）がその name を使用して参照できるように、ページアイテムはその label を使用して参照することができます。label プロパティを使用してアイテムを参照する例を、次のスクリプトに示します（完全なスクリプトについては、ScriptLabel を参照してください）。

```
set myDocument to make document
set myPage to page 1 of myDocument
set myPageWidth to page width of document preferences of myDocument
set myPageHeight to page height of document preferences of myDocument
--Create 10 random page items.
repeat with i from 1 to 10
    set myX1 to my myGetRandom(0, myPageWidth, false)
    set myY1 to my myGetRandom(0, myPageHeight, false)
    set myX2 to my myGetRandom(0, myPageWidth, false)
    set myY2 to my myGetRandom(0, myPageHeight, false)
    tell myPage
        set myRectangle to make rectangle with properties {geometric bounds:{myY1, myX1, myY2,
myX2}}
    end tell
    if my myGetRandom(0, 1, true) = 1 then
        set label of myRectangle to "myScriptLabel"
    end if
end repeat
set myCount to 0
repeat with i from 1 to count of page items of myPage
```

```
    if label of page item i of myPage is "myScriptLabel" then
        set myCount to myCount + 1
    end if
end repeat
display dialog ("Found " & myCount & " page items with the label.")
--This function gets a random number in the range myStart to myEnd.
on myGetRandom(myStart, myEnd, myInteger)
    set myRange to myEnd - myStart
    if myInteger = true then
        set myRandom to myStart + (random number from myStart to myEnd)
    else
        set myRandom to myStart + (random number from myStart to myEnd) as integer
    end if
    return myRandom
end myGetRandom
```

また、`label` プロパティがサポートされているオブジェクトでは、カスタムラベルもサポートされています。スクリプトでカスタムラベルを設定するには、`insert label` メソッドを使用します。カスタムラベルを抽出するには、`extract label` メソッドを使用します。次のスクリプトにその例を示します（チュートリアルスクリプトの CustomLabel より）。

```
tell application "Adobe InDesign CS6"
    set myDocument to make document
    tell view preferences of myDocument
        set horizontal measurement units to points
        set vertical measurement units to points
    end tell
    set myPage to page 1 of myDocument
    tell myPage
        set myRectangle to make rectangle with properties
        {geometric bounds:{72, 72, 144, 144}}
        --Insert a custom label using insert label. The first parameter is the
        --name of the label, the second is the text to add to the label.
        tell myRectangle
            insert label key "CustomLabel" value "This is some text stored
            in a custom label."
            --Extract the text from the label and display it in an alert.
            set myString to extract label key "CustomLabel"
        end tell
        display dialog ("Custom label contained: " & myString)
    end tell
end tell
```

起動時におけるスクリプトの自動実行

InDesign の起動時にスクリプトを自動実行させるには、Scripts フォルダーの下にあるスタートアップ用のフォルダーである Startup Scripts フォルダーにスクリプトファイルを保存します（詳しくは、『Adobe InDesign スクリプティングチュートリアル』の「スクリプトのインストール」を参照してください）。

3 ドキュメント

章の更新ステータス		
CS6	変更	この章で説明する作業実行方法のリストを更新しました。 41 ページの「調整型レイアウトの作成」という節の追加。 44 ページの「コンテンツの収集とドロップ」という節の追加。 50 ページの「グレースケール PDF の書き出し」という副節の追加。 52 ページの「PDF フォームとしての書き出し」という副節の追加。 古い情報となった CS5 の新機能に関する言及を、39 ページの「ページサイズとレイアウトの調整」、39 ページの「ページのサイズ変更とフレーム調整」および 40 ページの「ページの変形」から削除。

InDesign で行う作業の多くは、ドキュメントに関連する作業です。例えば、ドキュメントの作成、保存、プリント、書き出しを行ったり、ドキュメントにページアイテム、カラー、スタイル、テキストなどを配置したりします。ドキュメントに関連する作業のほとんどは、InDesign スクリプティングを使用して自動化できます。

この章では、次の作業を行う方法について説明します。

- 基本的なドキュメント管理作業
  - 新規ドキュメントの作成
  - ドキュメントのオープン
  - ドキュメントの保存
  - ドキュメントのクローズ
- 基本的なページレイアウト操作
  - ページサイズやドキュメントの長さの設定
  - 裁ち落としと印刷可能領域の定義
  - ページの段組やマージンの指定
- ペーストボードの外観の変更
- ガイドやグリッドの使用
- 測定単位や定規の原点の変更
- ドキュメントプリセットの定義と適用
- マスターページ（マススプレッド）の設定
- テキストフォーマットのデフォルトの設定
- XMP メタデータ（ファイルに関する情報）の追加
- ドキュメントテンプレートの作成
- 透かしの作成
- ページ別に異なるサイズの適用（複数のページサイズ）

- 様々なデバイスまたはサイズや方向の異なるドキュメントに対応するための、ページに対する柔軟なレイアウトフォーマットの適用
- ドキュメントのプリント
- Adobe PDF としてのドキュメントの書き出し
- EPS としてのドキュメントのページごとの書き出し
- ePub ファイルとしてのドキュメントの書き出し

ここでは、読者が『Adobe InDesign スクリプティングチュートリアル』を既に読んでおり、スクリプトを作成、インストール、実行する方法を理解しているものとして説明を行います。

ドキュメントの基本的な操作

ドキュメントのオープン、クローズ、保存といった操作は、ドキュメントに対する最も基本的な操作です。ここでは、スクリプティングでこれらの操作を実行する方法について説明します。

新規ドキュメントの作成

新規ドキュメントを作成する方法を、次のスクリプトに示します （完全なスクリプトについては、MakeDocument を参照してください）。

```
tell application "Adobe InDesign CS6"
    set myDocument to make document
end tell
```

ドキュメントプリセットを使用してドキュメントを作成するには、`make` コマンドのオプションのパラメーターを使用して、ドキュメントプリセットを指定します。次のスクリプトにその例を示します （完全なスクリプトについては、MakeDocumentWithPreset を参照してください）。

```
tell application "Adobe InDesign CS6"
    --Replace "myDocumentPreset" in the following line with the name
    --of the document preset you want to use.
    set myDocument to make document with properties
    {document preset:"myDocumentPreset"}
end tell
```

ドキュメントウィンドウを表示せずにドキュメントを作成することもできます。次のスクリプトにその例を示します（チュートリアルスクリプトの MakeDocumentWithParameters より）。

```
--Creates a new document without showing the document window.
--The "showing window" parameter controls the visibility of the document.
--Hidden documents are not minimized, and will not appear until
--you tell the document to create a new window.
tell application "Adobe InDesign CS6"
    set myDocument to make document with properties {showing window:false}
    --To show the window:
    --tell myDocument
    --set myWindow to make window
    --end tell
end tell
```

スクリプトの処理によっては、ドキュメントウィンドウを非表示にすることで実行速度がより速くなることがあります。

ドキュメントを開く

既存のドキュメントを開く方法を、次のスクリプトに示します（完全なスクリプトについては、OpenDocument を参照してください）。

```
tell application "Adobe InDesign CS6"
    --You'll have to fill in your own file path.
    set myDocument to open "yukino:myTestDocument.indd"
end tell
```

open コマンドの showing window パラメーターを false に設定すると（デフォルトでは true になっています）、ドキュメントが表示されないように（非表示に）なります。スクリプトのパフォーマンスを向上させたい場合は、この方法を利用したいと思うでしょう。非表示のドキュメントを表示するには、新規ウィンドウを作成します。次のスクリプトにその例を示します（チュートリアルスクリプトの OpenDocumentInBackground より）。

```
tell application "Adobe InDesign CS6"
    --You can use the "showing window" parameter to open files
    --without displaying them. This can speed up many scripting
    --operations, and makes it possible for a script to operate
    --on a file in the background. To display a document you've
    --opened this way, tell the document to create a new window.
    --You'll have to fill in your own file path.
    set myDocument to open "yukino:myTestDocument.indd" <lb>
    without showing window
    --At this point, your script could change or get information
    --from the hidden document. Once you've done that, you can show
    --the document window:
    tell myDocument to make window
end tell
```

ドキュメントの保存

InDesign のユーザーインターフェイスでは、ファイルを保存するにはファイル／保存を選択し、別の名前でファイルを保存するにはファイル／別名で保存を選択します。InDesign スクリプティングで同じ処理を行うには、save コマンドを使用します。次のスクリプトにその例を示します（チュートリアルスクリプトの SaveDocument より）。

```
--Saves the active document.
--If the active document has been changed since it was last saved, save it.
tell application "Adobe InDesign CS6"
    if modified of active document is true then
        tell active document to save
    end if
end tell
```

save コマンドには、オプションのパラメーターが 2 つあります。最初のパラメーター（to）には、保存先のファイルを指定します。2 つ目のパラメーター（stationery）は、true に設定するとドキュメントがテンプレートとして保存されます。次のスクリプトにその例を示します（チュートリアルスクリプトの SaveDocumentAs より）。

```
--If the active document has not been saved (ever), save it.
tell application "Adobe InDesign CS6"
    if saved of active document is false then
        --If you do not provide a file name, InDesign displays the Save dialog box.
        tell active document to save saving in "yukino:myTestDocument.indd"
    end if
end tell
```

ドキュメントをテンプレートとして保存することもできます。次のスクリプトにその例を示します（チュートリアルスクリプトの SaveAsTemplate より）。

```
--Save the active document as a template.
tell application "Adobe InDesign CS6"
    set myDocument to active document
    tell myDocument
        if saved is true then
            --Convert the file name to a string.
            set myFileName to full name
            set myFileName to my myReplace(myFileName, ".indd", ".indt")
        else
            --If the document has not been saved, then give it a default file
            --name/file path. You'll have to fill in the file path.
            set myFileName to "yukino:myTestDocument.indt"
        end if
        save to myFileName with stationery
    end tell
end tell
on myReplace(myString, myFindString, myChangeString)
    set AppleScript's text item delimiters to myFindString
    set myTextList to every text item of (myString as text)
    set AppleScript's text item delimiters to myChangeString
    set myString to myTextList as string
    set AppleScript's text item delimiters to ""
    return myString
end myReplace
```

ドキュメントを閉じる

`close` コマンドはドキュメントを閉じます。次のスクリプトにその例を示します（チュートリアルスクリプトの CloseDocument より）。

```
tell application "Adobe InDesign CS6"
    close document 1
    --document 1 always refers to the front-most document.
    --Note that you could also use:
    --close active document
end tell
```

`close` コマンドには、最大で 2 つのオプションパラメーターを指定できます。次のスクリプトにその例を示します（チュートリアルスクリプトの CloseWithParameters より）。

```
tell application "Adobe InDesign CS6"
--Use "saving yes" to save the document,
--or "saving no" to close the document without saving,
--or "saving ask" to display a prompt. If you use
--"saving yes",you'll need to provide a reference
--to a file to save to in the second parameter (saving in).
    --If the file has never been saved (it's an untitled file),
    --display a prompt.
    if saved of active document is not equal to true then
        close active document saving ask
        --Or, to save to a specific file name
        --(you'll have to fill in the file path):
        --set myFile to "yukino:myTestDocument.indd"
        --close active document saving yes saving in myFile
    else
        --If the file has already been saved to a file, save it.
        close active document saving yes
    end if
end tell
```

開いているすべてのドキュメントを保存せずに閉じることもできます。次のスクリプトにその例を示します（チュートリアルスクリプトの CloseAll より）。

```
tell application "Adobe InDesign CS6"
    tell documents to close without saving
end tell
```

基本的なページレイアウト

それぞれのドキュメントでは、デフォルトのページサイズ、割り当てられたページ数、裁ち落としと印刷可能領域、段組とマージンによって、コンテンツを配置するための領域が定義されます。これらのパラメーターはすべて、スクリプティングで操作することができます。次の各節でその例を示します。

ページサイズとドキュメントの長さの定義

InDesign のユーザーインターフェイスで新規ドキュメントを作成するときには、デフォルトのページサイズ、ページ数、ページの方向、見開きページにするかどうかなどを指定できます。InDesign スクリプティングでドキュメントを作成するには `make document` コマンドを使用しますが、上記のような設定は指定できません。ドキュメントを作成した後で、`document preferences` オブジェクトを使用して設定を行います。次のスクリプトにその例を示します（チュートリアルスクリプトの DocumentPreferences より）。

```
tell application "Adobe InDesign CS6"
    set myDocument to make document
    tell document preferences of myDocument
        set page height to "800pt"
        set page width to "600pt"
        set page orientation to landscape
        set pages per document to 16
    end tell
end tell
```

注意 : `application` オブジェクトにも、`document preferences` というオブジェクトがあります。このオブジェクトのプロパティを設定することで、ページの高さや幅などのアプリケーションデフォルトを設定できます。また、個別のページサイズを設定することもできます。詳しくは、「ページサイズとレイアウトの調整」を参照してください。

裁ち落としと印刷可能領域の定義

InDesign で使用される裁ち落としまたは印刷可能領域という用語は、プリントしたり PDF に書き出したりすることが可能な、ページマージンの外部の領域のことを表します。一般に、こうした領域は、ページ枠からはみ出して配置されるオブジェクトがある場合や（裁ち落とし）、ジョブ情報やドキュメント情報を含めておきたい場合に（印刷可能領域）使用されます。この 2 つの領域は、プリントしたり書き出したりするかどうかを、ページ枠とは独立に設定することができます。例えば、ドキュメントの最終プリントでは、印刷可能領域に配置した情報を除外してプリントすることができます。新規ドキュメントの裁ち落としと印刷可能領域を設定する方法を、次のスクリプトに示します （完全なスクリプトについては、BleedAndSlug を参照してください）。

```
tell application "Adobe InDesign CS6"
    --Create a new document.
    set myDocument to make document
    --The bleed and slug properties belong to the document preferences object.
    tell document preferences of myDocument
        --Bleed
        set document bleed bottom offset to "3p"
        set document bleed top offset to "3p"
        set document bleed inside or left offset to "3p"
        set document bleed outside or right offset to "3p"
        --Slug
        set slug bottom offset to "18p"
        set slug top offset to "3p"
        set slug inside or left offset to "3p"
        set slug right or outside offset to "3p"
    end tell
end tell
```

この例のように、裁ち落としの距離がすべて等しい場合は、代わりに `document bleed uniform size` プロパティを使用することもできます。次のスクリプトにその例を示します（チュートリアルスクリプトの UniformBleed より）。

```
tell application "Adobe InDesign CS6"
    --Create a new document.
    set myDocument to make document
    --The bleed properties belong to the document preferences object.
    tell document preferences of myDocument
        --Bleed
        set document bleed top offset to "3p"
        set document bleed uniform size to true
    end tell
end tell
```

印刷可能領域の距離がすべて等しい場合は、`document slug uniform size` プロパティが使用できます。次のスクリプトにその例を示します（チュートリアルスクリプトの UniformSlug より）。

```
tell application "Adobe InDesign CS6"
    --Create a new document.
    set myDocument to make document
    --The bleed properties belong to the document preferences object.
    tell document preferences of myDocument
        --Slug
        set document slug uniform size to true
        set slug top offset to "3p"
    end tell
end tell
```

裁ち落としや印刷可能領域は、幅や高さを設定できるだけでなく、その領域を表すガイドのカラーも設定できます。このプロパティは、`document preferences` オブジェクトではなく、`pasteboard preferences` オブジェクトに用意されています。次のスクリプトにその例を示します（チュートリアルスクリプトの BleedSlugGuideColors より）。

```
tell application "Adobe InDesign CS6"
    --Assumes you have a document open.
    tell pasteboard preferences of active document
        --Any of InDesign's guides can use the UIColors constants...
        set bleed guide color to cute teal
        set slug guide color to charcoal
        --...or you can specify a list of RGB values
        --(with values from 0 to 255)
        set bleed guide color to {0, 198, 192}
        set slug guide color to {192, 192, 192}
    end tell
end tell
```

ページのマージンと段組の設定

ドキュメントの各ページでは、それぞれ独自のマージンや段組を設定することができます。InDesign スクリプティングでは、各ページの margin preferences オブジェクトのプロパティを使用して、それらを設定できます。次のサンプルスクリプトでは、新規ドキュメントを作成してから、マスタースプレッドのすべてのページに対してマージンと段組を設定しています （完全なスクリプトについては、PageMargins を参照してください）。

```
tell application "Adobe InDesign CS6"
    set myDocument to make document
    tell view preferences of myDocument
        set horizontal measurement units to points
        set vertical measurement units to points
    end tell
    tell master spread 1 of myDocument
        tell margin preferences of pages
            set top to 36
            set left to 36
            set bottom to 48
            set right to 36
        end tell
    end tell
end tell
```

特定のページのマージンを設定するには、そのページの margin preferences を使用します。次のスクリプトにその例を示します（チュートリアルスクリプトの PageMarginsForOnePage より）。

```
tell application "Adobe InDesign CS6"
    set myDocument to make document
    tell view preferences of myDocument
        set horizontal measurement units to points
        set vertical measurement units to points
    end tell
    tell margin preferences of page 1 of myDocument
        set top to 36
        set left to 36
        set bottom to 48
        set right to 36
    end tell
end tell
```

InDesign では、対応するマージンの合計よりも小さなページは作成できません。つまり、ページの幅は、左右のページマージンの合計よりも大きくする必要があり、ページの高さは、上下のマージンの合計よりも大きくする必要があります。InDesign のユーザーインターフェイス（新規ドキュメントダイアログボックス）を使用して非常に小さなページ（新聞に掲載する広告など）を作成する場合は、ドキュメントのデフォルトのページマージンフィールドに新しい値を入力することで、正しいマージンサイズが簡単に設定できます。

しかし、スクリプティングで小さなページを作成する場合は、このように簡単にはいきません。なぜなら、ドキュメントの作成時には、アプリケーションのデフォルトの margin preferences が適用されるからです。このマージンは、マスターページを含む、ドキュメント内のすべてのページに適用されます。ドキュメントの margin preferences を変更した場合、影響を受けるのは新規ページのみで、既存のページには影響はありません。設定しようとしたページの高さや幅が、既存のどのページの対応するマージンの合計よりも小さい場合、ページサイズは変更されません。

これを解決するには、2 つの方法があります。1 つは、既存のページのマージンを設定してからページサイズを変更する方法です。次のスクリプトにその例を示します（チュートリアルスクリプトの PageMarginsForSmallPages より）。

```
tell application "Adobe InDesign CS6"
    set myDocument to make document
    tell margin preferences of page 1 of myDocument
        set top to 0
        set left to 0
        set bottom to 0
        set right to 0
    end tell
    tell master spread 1 of myDocument
        tell margin preferences of pages
            set top to 0
            set left to 0
            set bottom to 0
            set right to 0
        end tell
    end tell
    --At this point, you can set your page size to a small width
    --and height (1x1 picas minimum).
    set page height of document preferences of myDocument to "1p"
    set page width of document preferences of myDocument to "6p"
end tell
```

もう 1 つは、アプリケーションのデフォルトの margin preferences を変更してからドキュメントを作成する方法です。次のスクリプトにその例を示します（チュートリアルスクリプトの ApplicationPageMargins より）。

```
tell application "Adobe InDesign CS6"
    tell margin preferences
        --Save the current application default margin preferences.
        set myY1 to top
        set myX1 to left
        set myY2 to bottom
        set myX2 to right
        --Set the application default margin preferences.
        set top to 0
        set left to 0
        set bottom to 0
        set right to 0
    end tell
    --At this point, you can create a new document.
    set myDocument to make document
    --At this point, you can set your page size to a small width and height
    --(1x1 picas minimum).
    set page height of document preferences of myDocument to "1p"
    set page width of document preferences of myDocument to "1p"
    --Reset the application default margin preferences to their former state.
    tell margin preferences
        set top to myY1
        set left to myX1
        set bottom to myY2
        set right to myX2
    end tell
end tell
```

ペーストボードの外観の変更

ペーストボードとは、InDesign のページやスプレッドの周囲にある領域のことです。この領域は、ページアイテムや作業メモを一時的に置いておく場所として利用できます。ペーストボードのサイズやカラーは、スクリプティングを使用して変更することができます。`preview background color` プロパティを使用することで、プレビューモードでのペーストボードのカラーを設定できます。次のスクリプトにその例を示します（チュートリアルスクリプトの PasteboardPreferences より）。

```
tell application "Adobe InDesign CS6"
    set myDocument to make document
    tell pasteboard preferences of myDocument
        --You can use either a number or a measurement string to set the
        --space above/below.
        set minimum space above and below to "12p"
        --You can set the pasteboard background color to any
        --of the predefined UIColor constants...
        set preview background color to gray
        --...or you can specify an array of RGB values
        --(with values from 0 to 255)
        --set preview Background Color to {192, 192, 192}
    end tell
end tell
```

ガイドとグリッド

ガイドやグリッドを使用すると、ドキュメントページにオブジェクトを簡単に配置できるようになります。他のユーザーに提供するテンプレートを作成する場合は、これらのアイテムを適切に配置することで、ユーザーの作業効率を高めることができます。

ガイドの定義

InDesign のガイドを使用すれば、ドキュメントページでオブジェクトを簡単に配置できるようになります。ガイドを使用する方法を、次のスクリプトに示します（完全なスクリプトについては、Guides を参照してください）。

```
tell application "Adobe InDesign CS6"
    set myDocument to make document
    set myPageWidth to page width of document preferences of myDocument
    set myPageHeight to page height of document preferences of myDocument
    tell page 1 of myDocument
        set myMarginPreferences to margin preferences
        --Place guides at the margins of the page.
        make guide with properties {orientation:vertical,
        location:left of myMarginPreferences}
        make guide with properties {orientation:vertical,
        location:(myPageWidth - (right of myMarginPreferences))}
        make guide with properties {orientation:horizontal,
        location:top of myMarginPreferences}
        make guide with properties {orientation:horizontal,
        location:(myPageHeight - (bottom of myMarginPreferences))}
        --Place a guide at the vertical center of the page.
        make guide with properties {orientation:vertical,
        location:(myPageWidth / 2)}
        --Place a guide at the horizontal center of the page.
        make guide with properties {orientation:horizontal,
        location:(myPageHeight / 2)}
    end tell
end tell
```

水平方向のガイドは、指定のページにのみ表示するか、スプレッド内のすべてのページにまたがって表示するかを設定できます。InDesign スクリプティングでは、`fit to page` プロパティを使用してこの設定を制御できます。このプロパティは、垂直方向のガイドには適用されません。

スクリプティングでも、ユーザーインターフェイスを使用した場合と同様に、ガイドのレイヤー、カラー、表示／非表示を変更できます。次のスクリプトにその例を示します（チュートリアルスクリプトの GuideOptions より）。

```
tell application "Adobe InDesign CS6"
    set myDocument to make document
    tell myDocument
        --Create a layer named "guide layer".
        set myLayer to make layer with properties {name:"guide layer"}
        --Add a series of guides to page 1.
        tell page 1
            --Create a guide on the layer we created above.
            make guide with properties {orientation:horizontal, <lb>
            location:"12p", item layer:myLayer}
            make guide with properties {item layer:myLayer, <lb>
            orientation:horizontal, location:"14p"}
            --Make a locked guide.
            make guide with properties {locked:true, <lb>
            orientation:horizontal, location:"16p"}
            --Set the view threshold of a guide.
            make guide with properties {view threshold:100, <lb>
            orientation:horizontal, location:"18p"}
            --Set the guide color of a guide using a UIColors constant.
            make guide with properties {guide color:gray, <lb>
            orientation:horizontal, location:"20p"}
            --Set the guide color of a guide using an RGB array.
            make guide with properties {guide color:{192, 192, 192}, <lb>
            orientation:horizontal, location:"22p"}
        end tell
    end tell
end tell
```

ガイドの作成には、スプレッドやマスタースプレッドの `create guides` コマンドを使用することもできます。次のスクリプトにその例を示します（チュートリアルスクリプトの CreateGuides より）。

```
tell application "Adobe InDesign CS6"
    set myDocument to make document
    tell spread 1 of myDocument
        --Parameters (all optional): row count, column count, row gutter,
        --column gutter, guide color, fit margins, remove existing, layer.
        --Note that the create guides command does not take an RGB
        --array for the guide color parameter.
        create guides number of rows 4 number of columns 4 row gutter "1p"   <lb>
        column gutter "1p" guide color gray with fit margins and remove existing
    end tell
end tell
```

grid preferences の設定

ドキュメントグリッドやベースライングリッドのプロパティを制御するには、`grid preferences` オブジェクトのプロパティを設定します。次のスクリプトにその例を示します（チュートリアルスクリプトの DocumentAndBaselineGrid より）。

```
tell application "Adobe InDesign CS6"
    set myDocument to make document
    set horizontal measurement units of view preferences of myDocument to points
    set vertical measurement units of view preferences of myDocument to points
    tell grid preferences of myDocument
        set baseline start to 56
        set baseline division to 14
        set baseline grid shown to true
        set horizontal gridline division to 14
        set horizontal grid subdivision to 5
        set vertical gridline division to 14
        set vertical grid subdivision to 5
        set document grid shown to true
    end tell
end tell
```

ガイドやグリッドへのスナップ

ドキュメントのグリッドやガイドのスナップ設定は、すべて guide preferences オブジェクトや grid preferences オブジェクトのプロパティとして用意されています。ガイドやグリッドのスナッププロパティを設定する方法を、次のスクリプトに示します （完全なスクリプトについては、GuideGridPreferences を参照してください）。

```
tell application "Adobe InDesign CS6"
    set myDocument to active document
    tell guide preferences of myDocument
        set guides in back to true
        set guides locked to false
        set guides shown to true
        set guides snapto to true
    end tell
    tell grid preferences of myDocument
        set document grid shown to false
        set document grid snapto to true
        --Objects "snap" to the baseline grid when guidePreferences.guideSnapTo
        --is set to true.
        set baseline grid shown to true
    end tell
end tell
```

測定単位と定規の変更

これまでのサンプルスクリプトでは、測定文字列（特定の測定単位を InDesign に使用させるための文字列。例えば、8.5 インチの場合は「8.5i」）を使用してきました。これは、スクリプトの実行時に別の測定単位系が使用されている可能性を考慮しているためです。

スクリプトで使用する測定単位系を指定するには、ドキュメントの view preferences オブジェクトを使用します。次のスクリプトにその例を示します（チュートリアルスクリプトの ViewPreferences より）。

```
tell application "Adobe InDesign CS6"
    set myDocument to active document
    tell view preferences of myDocument
        --Measurement unit choices are:
        --picas, points, inches, inches decimal, millimeters, centimeters, or ciceros
        --Set horizontal and vertical measurement units to points.
        set horizontal measurement units to points
        set vertical measurement units to points
    end tell
end tell
```

スクリプト全体で特定の測定単位系を使用する場合は、スクリプトの先頭で測定単位を変更し、スクリプトの最後で元の測定単位に戻します。次のスクリプトにその例を示します（チュートリアルスクリプトの ResetMeasurementUnits より）。

```
tell application "Adobe InDesign CS6"
    set myDocument to active document
    tell view preferences of myDocument
        set myOldXUnits to horizontal measurement units
        set myOldYUnits to vertical measurement units
        set horizontal measurement units to points
        set vertical measurement units to points
    end tell
    --At this point, you can perform any series of script actions that depend on
    --the measurement units you've set. At the end of the script, reset
    --the measurement units to their original state.
    tell view preferences of myDocument
        set horizontal measurement units to myOldXUnits
        set vertical measurement units to myOldYUnits
    end tell
end tell
```

ドキュメントプリセットの定義と適用

InDesign のドキュメントプリセットを使用すれば、よく使用するドキュメント設定情報（ページサイズ、ページマージン、段組、裁ち落としと印刷可能領域など）を保存して、適用することができます。新規ドキュメントを作成する際には、ドキュメントプリセットに基づいてドキュメントを作成することができます。

既存の値をコピーしてプリセットを作成する

既存のドキュメントの設定をサンプルとして使用してドキュメントプリセットを作成するには、ドキュメントプリセットで使用したいドキュメント設定プロパティが含まれているドキュメントを開き、その後で次のスクリプトを実行します（チュートリアルスクリプトの DocumentPresetByExample より）。

```
tell application "Adobe InDesign CS6"
    if (count documents) > 0 then
        set myDocument to active document
        --If the document preset "myDocumentPreset"
        --does not already exist, create it.
        try
            set myDocumentPreset to document preset "myDocumentPreset"
        on error
            set myDocumentPreset to make document preset with properties
            {name:"myDocumentPreset"}
        end try
        --Fill in the properties of the document preset
        --with the corresponding
        --properties of the active document.
        tell myDocumentPreset
            --Note that the following gets the page margins from the margin preferences
            --of the document; to get the margin preferences from the active page,
            --replace "myDocument" with "active page of active window" in the
            --following line (assuming the active window is a layout window).
            set myMarginPreferences to margin preferences of myDocument
            set left to left of myMarginPreferences
            set right to right of myMarginPreferences
            set top to top of myMarginPreferences
            set bottom to bottom of myMarginPreferences
            set column count to column count of myMarginPreferences
            set column gutter to column gutter of myMarginPreferences
            set document bleed bottom offset to document bleed bottom offset
            of document preferences of myDocument
            set document bleed top offset to document bleed top offset
            of document preferences of myDocument
            set document bleed inside or left offset to document bleed inside
            or left offset of document preferences of myDocument
            set document bleed outside or right offset to document bleed outside
```

```
            or right offset of document preferences of myDocument
            set facing pages to facing pages of document preferences of myDocument
            set page height to page height of document preferences of myDocument
            set page width to page width of document preferences of myDocument
            set page orientation to page orientation of document preferences
            of myDocument
            set pages per document to pages per document of document preferences
            of myDocument
            set slug bottom offset to slug bottom offset of document preferences
            of myDocument
            set slug top offset to slug top offset of document preferences
            of myDocument
            set slug inside or left offset to slug inside or left offset
            of document preferences of myDocument
            set slug right or outside offset to slug right or outside offset
            of document preferences of myDocument
        end tell
    end if
end tell
```

ドキュメントプリセットの作成

明示的に値を指定してドキュメントプリセットを作成するには、次のスクリプトを実行します（チュートリアルスクリプトの DocumentPreset より）。

```
tell application "Adobe InDesign CS6"
    --If the document preset "myDocumentPreset" does not already exist, create it.
    try
        set myDocumentPreset to document preset "myDocumentPreset"
    on error
        set myDocumentPreset to make document preset
        with properties {name:"myDocumentPreset"}
    end try
    --Fill in the properties of the document preset.
    tell myDocumentPreset
        set page height to "9i"
        set page width to "7i"
        set left to "4p"
        set right to "6p"
        set top to "4p"
        set bottom to "9p"
        set column count to 1
        set document bleed bottom offset to "3p"
        set document bleed top offset to "3p"
        set document bleed inside or left offset to "3p"
        set document bleed outside or right offset to "3p"
        set facing pages to true
        set page orientation to portrait
        set pages per document to 1
        set slug bottom offset to "18p"
        set slug top offset to "3p"
        set slug inside or left offset to "3p"
        set slug right or outside offset to "3p"
    end tell
end tell
```

マススプレッドの設定

基本的なドキュメントページのサイズ、印刷可能領域、裁ち落としを設定したら、ドキュメントのマススプレッドを定義するのが一般的です。次のスクリプトにその例を示します（完全なスクリプトについては、MasterSpread を参照してください）。

```
tell application "Adobe InDesign CS6"
    set myDocument to make document
    --Set up the document.
    tell document preferences of myDocument
        set page height to "11i"
        set page width to "8.5i"
        set facing pages to true
        set page orientation to portrait
    end tell
    --Set the document's ruler origin to page origin. This is very important--
    --if you don't do this, getting objects to the correct position on the
    --page is much more difficult.
    set ruler origin of view preferences of myDocument to page origin
    tell master spread 1 of myDocument
        --Set up the left page (verso).
        tell margin preferences of page 1
            set column count to 3
            set column gutter to "1p"
            set bottom to "6p"
            --"left" means inside, "right" means outside.
            set left to "6p"
            set right to "4p"
            set top to "4p"
        end tell
        --Add a simple footer with a section number and page number.
        tell page 1
            set myTextFrame to make text frame ¬
                    with properties {geometric bounds:{"61p", "4p", "62p", "45p"}}
            tell myTextFrame
                set contents of insertion point 1 to section marker
                set contents of insertion point 1 to Em space
                set contents of insertion point 1 to auto page number
                set justification of paragraph 1 to left align
            end tell
        end tell
        --Set up the right page (recto).
        tell margin preferences of page 2
            set column count to 3
            set column gutter to "1p"
            set bottom to "6p"
            --"left" means inside, "right" means outside.
            set left to "6p"
            set right to "4p"
            set top to "4p"
        end tell
        --Add a simple footer with a section number and page number.
        tell page 2
            set myTextFrame to make text frame ¬
                    with properties {geometric bounds:{"61p", "6p", "62p", "47p"}}
            tell myTextFrame
                set contents of insertion point 1 to auto page number
                set contents of insertion point 1 to Em space
                set contents of insertion point 1 to section marker
                set justification of paragraph 1 to right align
            end tell
        end tell
    end tell
end tell
```

ドキュメントページにマススプレッドを適用するには、ドキュメントページの `applied master` プロパティを使用します。次のスクリプトにその例を示します（チュートリアルスクリプトの ApplyMaster より）。

```
--Assumes that the active document has a master page named "B-Master"
--and at least three document pages.
tell application "Adobe InDesign CS6"
    tell active document
        set applied master of page 2 to master spread "B-Master"
    end tell
end tell
```

同じプロパティを使用して、マススプレッドページにマススプレッドを適用します。次のスクリプトにその例を示します（チュートリアルスクリプトの ApplyMasterToMaster より）。

```
--Assumes that the active document has master spread named "B-Master"
--that is not the same as the first master spread in the document.
tell application "Adobe InDesign CS6"
    tell active document
        set applied master of page 2 of master spread 1 to
        master spread "B-Master"
    end tell
end tell
```

XMP メタデータの追加

メタデータとは、ファイルの内容や作成元などの属性を示す情報のことです。InDesign のユーザーインターフェイスでは、ファイル情報ダイアログボックス（ファイル／ファイル情報を選択）を使用して、メタデータの入力、編集、表示を行います。このメタデータには、ドキュメントの作成日、修正日、作成者、著作権のステータスなどがあります。こうした情報はすべて、XMP（Adobe Extensible Metadata Platform）を使用して保存されます。XMP は、ドキュメントにメタデータを埋め込むためのオープンスタンダードです。

XMP について詳しくは、http://partners.adobe.com/public/developer/xmp/sdk に掲載されている XMP の仕様を参照してください。

InDesign スクリプティングでも、ドキュメントに XMP 情報を追加することができます。ドキュメントのすべての XMP プロパティは、ドキュメントの `metadataPreferences` オブジェクトに含まれています。ドキュメントの標準 XMP データを入力する例を次に示します。

この例では、XMP 情報を拡張する方法も示されています。ドキュメントにメタデータを添付する必要があり、そのデータが metadata preferences オブジェクトで定義されているカテゴリーに該当しない場合は、独自のメタデータコンテナを作成できます（この例では `email`）。（完全なスクリプトについては、MetadataExample を参照してください）

```
tell application "Adobe InDesign CS6"
    set myDocument to make document
    tell metadata preferences of myDocument
        set author to "Adobe"
        set copyright info URL to "http://www.adobe.com"
        set copyright notice to "This document is copyrighted."
        set copyright status to yes
        set description to "Example of xmp metadata scripting in InDesign CS"
        set document title to "XMP Example"
        set job name to "XMP_Example_2004"
        set keywords to {"animal", "mineral", "vegetable"}
        --The metadata preferences object also includes the read-only
        --creator, format, creationDate, modificationDate,
        --and serverURL properties that are automatically entered
        --and maintained by InDesign.
        --Create a custom XMP container, "email"
        set myNewContainer to create container item ¬
                    namespace "http://ns.adobe.com/xap/1.0/" path "email"
        set property namespace "http://ns.adobe.com/xap/1.0/" ¬
                    path "email/*[1]" value "someone@adobe.com"
    end tell
end tell
```

ドキュメントテンプレートの作成

次の例では、新規ドキュメントを作成し、裁ち落としと印刷可能領域を定義し、ドキュメントの XMP メタデータに情報を追加し、マスターページを設定し、ページフッターを追加し、印刷可能領域の表にジョブ情報を追加しています（完全なスクリプトについては、DocumentTemplate を参照してください）。

```
tell application "Adobe InDesign CS6"
    --Make a new document.
    set myDocument to make document
    tell myDocument
        tell document preferences
            set page width to "7i"
            set page height to "9i"
            set page orientation to portrait
        end tell
        tell margin preferences
            set top to ((14 * 4) & "pt") as string
            set left to ((14 * 4) & "pt") as string
            set bottom to "74pt"
            set right to ((14 * 5) & "pt") as string
        end tell
        --Set up the bleed and slug areas.
        tell document preferences
            --Bleed
            set document bleed bottom offset to "3p"
            set document bleed top offset to "3p"
            set document bleed inside or left offset to "3p"
            set document bleed outside or right offset to "3p"
            --Slug
            set slug bottom offset to "18p"
            set slug top offset to "3p"
            set slug inside or left offset to "3p"
            set slug right or outside offset to "3p"
        end tell
        --Create a color.
        try
            set myColor to color "PageNumberRed"
        on error
            set myColor to make color with properties {name:"PageNumberRed",
            model:process, color value:{20, 100, 80, 10}}
        end try
        --Next, set up some default styles.
        --Create up a character style for the page numbers.
        try
            set myCharacterStyle to character style "page_number"
        on error
            set myCharacterStyle to make character style
            with properties {name:"page_number"}
        end try
        set fill color of myCharacterStyle to color "PageNumberRed"
        --Create up a pair of paragraph styles for the page footer text.
        --These styles have only basic formatting.
        try
            set myParagraphStyle to paragraph style "footer_left"
        on error
            set myParagraphStyle to make paragraph style
            with properties {name:"footer_left"}
        end try
        --Create up a pair of paragraph styles for the page footer text.
        try
            set myParagraphStyle to paragraph style "footer_right"
        on error
            set myParagraphStyle to make paragraph style  ¬
                    with properties {name:"footer_right",  ¬
                    based on:paragraph style "footer_left", justification:right align}
```

```
end try
--Create a layer for guides.
try
    set myLayer to layer "GuideLayer"
on error
    set myLayer to make layer with properties {name:"GuideLayer"}
end try
--Create a layer for the footer items.
try
    set myLayer to layer "Footer"
on error
    set myLayer to make layer with properties {name:"Footer"}
end try
--Create a layer for the slug items.
try
    set myLayer to layer "Slug"
on error
    set myLayer to make layer with properties {name:"Slug"}
end try
--Create a layer for the body text.
try
    set myLayer to layer "BodyText"
on error
    set myLayer to make layer with properties {name:"BodyText"}
end try
tell view preferences
    set ruler origin to page origin
    set horizontal measurement units to points
    set vertical measurement units to points
end tell
--Document baseline grid and document grid
tell grid preferences
    set baseline start to 56
    set baseline division to 14


    set baseline grid shown to false
    set horizontal gridline division to 14
    set horizontal grid subdivision to 5
    set vertical gridline division to 14
    set vertical grid subdivision to 5
    set document grid shown to false
end tell
--Document XMP information.
tell metadata preferences
    set author to "Adobe"
    set copyright info URL to "http:--www.adobe.com"
    set copyright notice to "This document is not copyrighted."
    set copyright status to no
    set description to "Example 7 x 9 book layout"
    set document title to "Example"
    set job name to "7 x 9 book layout template"
    set keywords to {"7 x 9", "book", "template"}
    set myNewContainer to create container item
    namespace "http://ns.adobe.com/xap/1.0/" path "email"
    set property namespace "http://ns.adobe.com/xap/1.0/"
    path "email/*[1]" value "someone@adobe.com"
end tell
--Set up the master spread.
tell master spread 1
    tell page 1
        set myMarginPreferences to margin preferences
        set myBottomMargin to (page height of document preferences of
        myDocument - (bottom of myMarginPreferences)
        set myLeftMargin to right of myMarginPreferences
        set myRightMargin to (page width of document preferences of myDocument)
```

```
    - (left of myMarginPreferences)
    make guide with properties {orientation:vertical,
    location:myRightMargin,item layer:layer "GuideLayer" of myDocument}
    make guide with properties {orientation:vertical,
    location:myLeftMargin,item layer:layer "GuideLayer" of myDocument}
    make guide with properties {orientation:horizontal,
    location:top of myMarginPreferences,
    item layer:layer "GuideLayer" of myDocument, fit to page:false}
    make guide with properties {orientation:horizontal,
    location:myBottomMargin,
    item layer:layer "GuideLayer" of myDocument, fit to page:false}
    make guide with properties {orientation:horizontal,
    location:myBottomMargin + 14,
    item layer:layer "GuideLayer" of myDocument, fit to page:false}
    make guide with properties {orientation:horizontal,
    location:myBottomMargin + 28,
    item layer:layer "GuideLayer" of myDocument, fit to page:false}
    set myLeftFooter to make text frame with properties
    {item layer:layer "Footer" of myDocument,
    geometric bounds:{myBottomMargin + 14, right of myMarginPreferences,
    myBottomMargin + 28, myRightMargin}}
    set contents of insertion point 1 of parent story of myLeftFooter
    to section marker
    set contents of insertion point 1 of parent story of myLeftFooter
    to Em space
    set contents of insertion point 1 of parent story of myLeftFooter
    to auto page number
    set applied character style of character 1 of parent story of
    myLeftFooter to character style "page_number" of myDocument
    set applied paragraph style of paragraph 1 of parent story of


    myLeftFooter to paragraph style "footer_left" of myDocument
    --Slug information.
    tell metadata preferences of myDocument
        set myEmail to get property
        namespace "http://ns.adobe.com/xap/1.0/" path "email/*[1]"
        set myDate to current date
        set myString to "Author:" & tab & author & tab & "Description:"
        & tab & description & return & "Creation Date:" & tab & myDate
        & tab & "Email Contact" & tab & myEmail
    end tell
    set myLeftSlug to make text frame with properties
    {item layer:layer "Slug" of myDocument,
    geometric bounds:{(page height of document preferences of myDocument)
     + 36, right of myMarginPreferences,
    (page height of document preferences of myDocument) + 144,
    myRightMargin}, contents:myString}
    tell parent story of myLeftSlug
        convert to table text 1
    end tell
    --Body text master text frame.
    set myLeftFrame to make text frame with properties   ¬
    {item layer:layer "BodyText" of myDocument,
    geometric bounds:{top of myMarginPreferences,
    right of myMarginPreferences, myBottomMargin, myRightMargin}}
end tell
tell page 2
    set myMarginPreferences to margin preferences
    set myLeftMargin to left of myMarginPreferences
    set myRightMargin to (page width of document preferences of myDocument)
    - (right of myMarginPreferences)
    make guide with properties {orientation:vertical,
    location:myLeftMargin,item layer:layer "GuideLayer" of myDocument}
    make guide with properties {orientation:vertical,
    location:myRightMargin, item layer:layer "GuideLayer" of myDocument}
```

```
                set myRightFooter to make text frame with properties
                {item layer:layer "Footer" of myDocument,
                geometric bounds:{myBottomMargin + 14, left of myMarginPreferences,
                myBottomMargin + 28, myRightMargin}}
                set contents of insertion point 1 of parent story of myRightFooter
                to auto page number
                set contents of insertion point 1 of parent story of myRightFooter
                to Em space
                set contents of insertion point 1 of parent story of myRightFooter
                to section marker
                set applied character style of character -1 of parent story of
                myRightFooter to character style "page_number" of myDocument
                set applied paragraph style of paragraph 1 of parent story of
                myRightFooter to paragraph style "footer_right" of myDocument
                --Slug information.
                set myRightSlug to make text frame with properties
                {item layer:layer "Slug" of myDocument,
                geometric bounds:{(page height of document preferences of myDocument)
                + 36, left of myMarginPreferences,(page height of document preferences
                of myDocument) + 144,myRightMargin}, contents:myString}
                tell parent story of myRightSlug
                    convert to table text 1


                end tell
                --Body text master text frame.
                set myRightFrame to make text frame with properties
                {item layer:layer "BodyText" of myDocument,
                geometric bounds:{top of myMarginPreferences,
                    left of myMarginPreferences, myBottomMargin, myRightMargin}}
            end tell
        end tell
        --Add section marker text--this text will appear in the footer.
        set marker of section 1 of myDocument to "Section 1"
        --When you link the master page text frames, one of the frames sometimes
        --becomes selected. Deselect it.
        select nothing
    end tell
end tell
```

透かしの作成

InDesign や InDesign Server では、スクリプトを使用してドキュメントに透かしを適用することができます。現時点では、InDesign に透かしを操作するユーザーインターフェイスは存在しません。

ドキュメントの透かし環境設定は、スクリプトを使用した次の 2 種類の方法で設定できます。

- アプリケーションレベルの watermark preferences が設定されている場合、InDesign で作成するすべての新規ドキュメントに、この透かし環境設定が適用されます。この設定は、既存のドキュメントには影響を与えません。
- ドキュメントレベルの watermark preferences は、該当のドキュメントにのみ適用されます。ドキュメントの watermark preferences を設定したり変更したりすると、ドキュメントの以前の透かし環境設定が上書きされます。

ドキュメントレベルおよびアプリケーションレベルの透かし環境設定は、いずれもドキュメントを閉じるかアプリケーションを終了してから、スクリプトで変更されるまで保持されます。

ドキュメントオブジェクトとアプリケーションオブジェクトの両方に、同じ透かし環境設定のグループがあります。

watermark preferences の設定

アプリケーションレベルで透かしを設定する方法を、次のスクリプトに示します。このコードの実行後は、作成するドキュメントのすべてに透かしが適用されます （アプリケーションの環境設定を指定する完全なスクリプトについては、ApplicationWatermark を参照してください）。

```
tell watermark preferences
    set watermark visibility to true
    set watermark do print to true
    set watermark draw in back to true
    set watermark text to "Confidential"
    set watermark font family to "Arial"
    set watermark font style to "Bold"
    set watermark font point size to 72
    set watermark font color to red
    set watermark opacity to 60
    set watermark rotation to -45
    set watermark horizontal position to watermark h center
    set watermark horizontal offset to 0
    set watermark vertical position to watermark v center
    set watermark vertical offset to 0
end tell
```

アプリケーションではなくドキュメントを参照するようにすると、同じ環境設定をドキュメントオブジェクトに適用できます （ドキュメントの環境設定を指定する完全なスクリプトについては、DocumentWatermark を参照してください）。

```
tell watermark preferences of document 1
    set watermark visibility to true
    set watermark do print to true
    set watermark draw in back to true
    set watermark text to "Confidential"
    set watermark font family to "Arial"
    set watermark font style to "Bold"
    set watermark font point size to 72
    set watermark font color to blue
    set watermark opacity to 60
    set watermark rotation to -45
    set watermark horizontal position to watermark h center
    set watermark horizontal offset to 0
    set watermark vertical position to watermark v center
    set watermark vertical offset to 0
end tell
```

透かしの無効化

透かしのアプリケーション設定をオフにすると、新規ドキュメントの透かし設定はデフォルトではオンにならなくなります。ただし、個々のドキュメントに透かしを設定することは可能です。アプリケーションレベルの透かしをオフにする方法を、次のスクリプトに示します。

```
tell application "Adobe InDesign CS6"
    set watermark visibility of watermark preferences to false
end tell
```

個々のドキュメントの透かしは、いつでもオフにすることができます。次のスクリプトにその例を示します。

```
tell application "Adobe InDesign CS6"
    tell document 1
        set watermark visibility of watermark preferences to false
    end tell
end tell
```

ページサイズとレイアウトの調整

InDesign では、1 つの InDesign ドキュメント内で別々のページサイズを設定できます。デフォルトのページサイズの設定について詳しくは、「ページサイズとドキュメントの長さの定義」を参照してください。

また、個々のページに変形を適用することもできます。

ページの選択

ページをサイズ変更したりページに変形を適用するには、まずページを選択する必要があります。InDesign のユーザーインターフェイスでページを選択するには、ツールパネルのページツールを使用します。また、スクリプトを使用してページを選択することもできます。次のスクリプトにその例を示します （完全なスクリプトについては、PageSelect を参照してください）。

```
--Given a document with four pages (1, 2, 3, 4)...
set myPages to pages of active document
--Select page 2 and 3.
select item 2 of myPages
select item 3 of myPages existing selection add to
--Select last page.
select item -1 of myPages existing selection add to
```

ページのサイズ変更とフレーム調整

スクリプトを使用して、ページ上のページアイテムをサイズ変更したり、フレームを調整したりできます。また、サイズ変更やフレーム調整の操作を複数のページに適用して各ページをサイズ変更することもできます。

注意：最小のページサイズは、ページのマージンによって決まります。詳しくは、「ページのマージンと段組の設定」を参照してください。

resize メソッドを使用してページをサイズ変更する方法を、次のスクリプトに示します （完全なスクリプトについては、PageResize を参照してください）。

```
--Given a document with four pages (1, 2, 3, 4)...
tell pages of active document
    --Resize page to two times bigger
    resize item 2 in inner coordinates from center anchor by multiplying current dimensions by
values {2, 2}
    --Resize page to 400 points width and 600 points height.
    resize item 3 in inner coordinates from center anchor by replacing current dimensions with
values {400, 600}
end tell
```

フレーム調整を行うと、ページのバウンディングボックスが変更されます。そのため、フレーム調整を使用して、バウンディングボックスを大きくしたり小さくしたりすることで、ページをサイズ変更できます。reframe メソッドを使用してページをサイズ変更する方法を、次のスクリプトに示します （完全なスクリプトについては、PageReframe を参照してください）。

```
--Given a document with four pages (1, 2, 3, 4)...
tell item 2 of pages of active document
    --Make the page one inch wider and one inch higher.
    set myBounds to bounds
    set myY1 to item 1 of myBounds
    set myX1 to item 2 of myBounds
    set myY2 to (item 3 of myBounds) + 72
    set myX2 to (item 4 of myBounds) + 72
    reframe in inner coordinates opposing corners {{myX1, myY1}, {myX2, myY2}}
end tell
```

ページの変形

オブジェクトの形状を変更する操作は、変形と呼ばれます。transform メソッドで、ページ上のページアイテムを回転、拡大・縮小、シアーおよび移動（平行移動）して、ページ上で使用できます。変形アーキテクチャの技術的な詳細については、「ページアイテムの変形」を参照してください。

ページを変形するには：

1. 変形マトリックスを作成します。
2. この変形マトリックスを transform メソッドを使用してページに適用します。

スクリプトを使用してページを変形する方法を、次のスクリプトに示します （完全なスクリプトについては、PageTransform を参照してください）。

```
--Given a document with four pages (1, 2, 3, 4)...
set myDocument to active document
set myPages to pages of myDocument

--Rotate a page around its center point.
set myRotateMatrix to make transformation matrix with properties {counterclockwise rotation
angle:27}
my myTransform(item 1 of myPages, myRotateMatrix)

--Scale a page around its center point.
set myScaleMatrix to make transformation matrix with properties {horizontal scale factor:0.8,
vertical scale factor:0.8}
my myTransform(item 2 of myPages, myScaleMatrix)

--Shear a page around its center point.
set myShearMatrix to make transformation matrix with properties {clockwise shear angle:30}
my myTransform(item 3 of myPages, myShearMatrix)

on myTransform(myPage, myTransformationMatrix)
    tell application "Adobe InDesign CS6"
        transform myPage in pasteboard coordinates from center anchor with matrix
myTransformationMatrix
    end tell
end myTransform
```

マスターページオーバーレイ

ページには複数のサイズが設定できるため、ページとそのマスターページで別々のサイズを設定することができます。適用されているマスターを検出するだけでなく、ページでは、マスターページがどのようにページ上に描画されるかを定めるマトリックスも維持されます。これをマスターページオーバーレイと呼びます。ツールパネルのページツールを使用してページを選択するときに、コントロールパネルの「マスターページオーバーレイを表示」チェックボックスをオンにすると、マスターページの配置仕様を確認できます。オーバーレイは、マウスで移動することができます。これは、マスターオーバーレイのマトリックスに変形を適用することで実現しています。ユーザーインターフェイスからは平行移動（*x* および *y* の移動）のみ可能ですが、スクリプトではその他の操作も実行できます。マスターページオーバーレイを変形する方法を、次のスクリプトに示します （完全なスクリプトについては、MasterPageTransform を参照してください）。

```
--Given a document with four pages (1, 2, 3, 4)...
set myDocument to active document
set myPages to pages of myDocument

--Rotate master page overlay around its top-left corner.
set myRotateMatrix to make transformation matrix with properties {counterclockwise rotation
angle:27}
set master page transform of item 1 of myPages to myRotateMatrix

--Scale master page overlay around its top-left corner.
set myScaleMatrix to make transformation matrix with properties {horizontal scale factor:0.5,
vertical scale factor:0.5}
set master page transform of item 2 of myPages to myScaleMatrix

--Shear master page overlay around its top-left corner.
set myShearMatrix to make transformation matrix with properties {clockwise shear angle:30}
set master page transform of item 3 of myPages to myShearMatrix

--Translate master page overlay 1 inch right and 2 inches down.
set myTranslateMatrix to make transformation matrix with properties {horizontal translation:72,
vertical translation:144}
set master page transform of item 4 of myPages to myTranslateMatrix
```

調整型レイアウトの作成

InDesign のレイアウト調整ワークフローを使用して作成した基本レイアウトは、異なる方向、サイズ、縦横比、またはまったく異なるデバイスクラスに簡単に適応させることができます。基本レイアウトの作成方法は、プリント用ドキュメントのレイアウトを作成する場合と同じです。このワークフローには主に 2 つの機能が関連しています。レイアウトに柔軟性を与えるリキッドレイアウトと、様々な方向およびデバイスクラス用の代替レイアウトを作成する代替レイアウト作成という機能です。

柔軟性のあるレイアウトの作成

レイアウトルールを使用して作成した 1 セットの物理的なページで構成される InDesign ファイルは、サイズや方向が異なるデバイス上でも、そのコンテンツがそれぞれのデバイスに適した状態で表示されます。必要であれば、サイズおよび方向ごとにページの外観を指定できます。次のスクリプト例に、その方法を示します （完全なスクリプトについては、LiquidLayout を参照してください）。

```
tell application "Adobe InDesign CS6"
set myDocument to make document
    set myPage to page 1 of myDocument
    -- Set layout rule to object rule
    tell myPage
        --Reposition and resize objects on the page as it resizes.
        set layout rule to object based
        --Create a text frame on the first page.
        set myTextFrame to make text frame
        set geometric bounds of myTextFrame to my myGetBounds(myDocument, myPage)
        set contents of myTextFrame to "This is object based layoutrule sample doc."
        --Create a rectangle
        set myItem to make rectangle
        set geometric bounds of myItem to {20, 20, 70, 70}
        set vertical layout constraints of myItem to {flexible dimension,
            flexible dimension, flexible dimension}
        set horizontal layout constraints of myItem to {fixed dimension,
            flexible dimension, flexible dimension}
    end tell
        end tell
tell application "Adobe InDesign CS6"
set myDocument2 to make document
    set myPage2 to page 1 of myDocument2
    -- Set layout rule to scale
```

```
    tell myPage2
        --Scale objects on the page as it resizes.
        set layout rule to scale
        --Create a text frame on the first page.
        set myTextFrame2 to make text frame
        set geometric bounds of myTextFrame2 to my myGetBounds(myDocument2, myPage2)
        set contents of myTextFrame2 to "This is scale layoutrule sample doc."
        set myItem2 to make rectangle
        set geometric bounds of myItem2 to {20, 20, 70, 70}
    end tell
end tell
tell application "Adobe InDesign CS6"
set myDocument3 to make document
    set myPage3 to page 1 of myDocument3
    -- Set layout rule to recenter
    tell myPage3
        --Recenter objects on the page as it resizes.
        set layout policy to recenter
        --Create a text frame on the first page.
        set myTextFrame3 to make text frame
        set geometric bounds of myTextFrame3 to my myGetBounds(myDocument3, myPage3)
        set contents of myTextFrame3 to "This is recenter layoutrule sample doc."
        set myItem3 to make rectangle
        set geometric bounds of myItem3 to {20, 20, 70, 70}
    end tell
end tell
tell application "Adobe InDesign CS6"
    set myDocument4 to make document
    set myPage4 to page 1 of myDocument4
    -- Set layout rule to guide based
    tell myPage4
        --objects based on the page as it resizes.
        set layout policy to guide based
        --Create a text frame on the first page.
        set myTextFrame4 to make text frame
        set geometric bounds of myTextFrame4 to my myGetBounds(myDocument4, myPage4)
        set contents of myTextFrame4 to "This is guide based layoutrule sample doc."
        set myItem4 to make rectangle
        set geometric bounds of myItem4 to {20, 20, 70, 70}
        make guide with properties {orientation:horizontal, location:"20p", guide type:liquid}
    end tell
end tell
tell application "Adobe InDesign CS6"
    set myDocument5 to make document
    set myPage5 to page 1 of myDocument5
    -- Set layout rule to use master
    tell myPage5
        --Use master on the page as it resizes.
        set layout rule to use master
        --Create a text frame on the first page.
        set myTextFrame5 to make text frame
        set geometric bounds of myTextFrame5 to my myGetBounds(myDocument5, myPage5)
        set contents of myTextFrame5 to "This is master page layoutrule sample doc."
        set myItem5 to make rectangle
        set geometric bounds of myItem5 to {20, 20, 70, 70}
    end tell
end tell
```

ガイドベースのレイアウトへのガイドの追加

ガイドベースのレイアウトルールにガイドスライスを追加する方法を、次のスクリプト例に示します （完全なスクリプトについては、AddGuides を参照してください）。

```
tell application "Adobe InDesign CS6"
    set myDocument to make document
    set myPage to page 1 of myDocument
    tell myPage
            set layout rule to guide based
        set myTextFrame to make text frame
        --Set the bounds of the text frame.
        set geometric bounds of myTextFrame to my myGetBounds(myDocument, myPage)
        --Enter text in the text frame.
        set contents of myTextFrame to "This is guide-based layoutrule sample doc."
        set myItem to make rectangle
        set geometric bounds of myItem to {20, 20, 70, 70}
    end tell
    tell myPage
        make guide with properties {orientation:horizontal, location:"20p", guide type:liquid}
    end tell
end tell
```

オブジェクトベースのレイアウトへの制約の設定

オブジェクトベースのレイアウトルールに制約を設定する方法を、次のスクリプト例に示します （完全なスクリプトについては、SetConstraints を参照してください）。

```
tell application "Adobe InDesign CS6"
    set myDocument to make document
    set myPage to page 1 of myDocument
    tell myPage
        set layout rule to object based
        set myTextFrame to make text frame
        --Set the bounds of the text frame.
        set geometric bounds of myTextFrame to my myGetBounds(myDocument, myPage)
        --Enter text in the text frame.
        set contents of myTextFrame to "This is object-based layoutrule sample doc."
        set myItem to make rectangle
        set geometric bounds of myItem to {20, 20, 70, 70}
        set vertical layout constraints of myItem to
            {flexible dimension, flexible dimension, flexible dimension}
        set horizontal layout constraints of myItem to
            {fixed dimension, flexible dimension, flexible dimension}
    end tell
end tell
```

代替レイアウトの作成

代替レイアウト作成は、コンテンツが含まれている既存ページの範囲を取得し、そのコンテンツのリンク付きコピーを、同じドキュメント内の新しいページセット上に自動的に作成します。これは、方向およびデバイスクラスごとに異なるレイアウトを作成する必要がある場合に使用します （完全なスクリプトについては、CreateAlternateLayout を参照してください）。

```
tell application "Adobe InDesign CS6"
    set myDocument to make document
    tell document preferences of myDocument
        set facing pages to false
        set pages per document to 1
        set page width to 1024
        set page height to 768
    end tell
    set myPage to page 1 of myDocument
    tell myPage
        set layout rule to object based
        set myItem to make rectangle
        set geometric bounds of myItem to {50, 50, 100, 100}
    end tell
    tell myDocument

            create alternate layout spread items spreads
                name "new layout"
                width page height of document preferences
                height page width of document preferences
                ppi 1
                layout rule preserve existing
                with create text styles and link text stories
    end tell
end tell
```

コンテンツの収集とドロップ

コンテンツドロッパーを使用すると、プリント用にデザインしたファイルからコンテンツをコピーし、別のドキュメントに貼り付けることができます。

スクリプトを使用してドキュメント内のコンテンツを収集する方法を、次のスクリプトに示します （完全なスクリプトについては、ContentCollectorAndPlacer を参照してください）。

```
--Invoke load with full parameters
tell content placer object 1
    load page items myRectangle
    with link page items, link stories, map styles and showing options
end tell
```

収集したコンテンツを別のドキュメントに配置する方法を、次のスクリプトに示します （完全なスクリプトについては、ContentCollectorAndPlacer を参照してください）。

```
--Invoke Page.contentPlace with full parameters
tell myPage1
    content place page items rectangles of myPage
    with link page items, link stories, map styles and showing options
end tell
```

ドキュメントのプリント

次のスクリプトでは、現在の print preferences を使用して、アクティブなドキュメントをプリントします（完全なスクリプトについては、PrintDocument を参照してください）。

```
tell application "Adobe InDesign CS6"
    print active document
end tell
```

ページ範囲を使用したプリント

プリントするページの範囲を指定するには、プリントを行う前に、ドキュメントの print preferences オブジェクトの page range プロパティを設定します。次のスクリプトにその例を示します（チュートリアルスクリプトの PrintPageRange より）。

```
tell application "Adobe InDesign CS6"
    set page range of print preferences to "1-3, 10"
    print
end tell
```

print preferences の設定

print preferences オブジェクトに保持されているプロパティは、プリントダイアログボックスの各パネルに含まれているオプションに対応しています。スクリプティングで print preferences を設定する方法を、次のスクリプトに示します（完全なスクリプトについては、PrintPreferences を参照してください）。

```
--PrintPreferences.as
--An InDesign CS6 AppleScript
--Sets the print preferences of the active document.
tell application "Adobe InDesign CS6"
    --Get the bleed amounts from the document's bleed and add a bit.
    tell document preferences of active document
        set myX1Offset to document bleed inside or left offset + 3
        set myY1Offset to document bleed top offset + 3
        set myX2Offset to document bleed outside or right offset + 3
        set myY2Offset to document bleed bottom offset + 3
    end tell
    tell print preferences of active document
        --Properties corresponding to the controls in the General panel
        --of the Print dialog box.
        --activePrinterPreset is ignored in this example--we'll set our
        --own print preferences.
        --printer can be either a string (the name of the printer) or
        --postscript file.
        set printer to postscript file
        --Here's an example of setting the printer to a specific printer.
        --set printer to "AGFA-SelectSet 5000SF v2013.108"
        --If the printer property is the name of a printer, then the ppd property
        --is locked (and will return an error if you try to set it).
        try
            --set PPD to "Device Independent"
        end try
        --If the printer property is set to postscript file, the copies
        --property is unavailable. Attempting to set it will generate an error.
        --set copies to 1
        --If the printer property is set to Printer.postscript file, or if the
        --selected printer does not support collation, then the collating
        --property is unavailable. Attempting to set it will generate an error.
        --set collating to false
        set reverse order to false
        --The setting of color output determines the settings available
        --to almost all other properties in the print preferences.
        try
            set color output to separations
        end try
        --pageRange can be either PageRange.allPages or a page range string.
        set page range to all pages
        set print spreads to false
        set print master pages to false
        --If the printer property is set to postScript file, then
        --the print file property contains the file path to the output file.
        --set printFile to "yukino:test.ps"
```

```
set sequence to all
--If trapping is on, setting the following properties
--will produce an error.
try
    if trapping is off then
        set print blank pages to false
        set print guides grids to false
        set print nonprinting to false
    end if
end try
------------------------------------------------------------------------
--Properties corresponding to the controls in the Setup panel of
--the Print dialog.
------------------------------------------------------------------------
set paper size to custom
--Page width and height are ignored if paper Size is not custom.
--set paper height to 1200
--set paper width to 1200
set print page orientation to portrait
set page position to centered
set paper gap to 0
set paper offset to 0
set paper transverse to false
set scale height to 100
set scale width to 100
set scale mode to scale width height
set scale proportional to true
--If trapping is on (application built in or Adobe inRip),
--attempting to set the following properties will produce an error.
if trapping is off then
    set thumbnails to false
    --The following properties are not needed because thumbnails is set to false.
    --set thumbnails per page to 4
    set tile to false
    --The following properties are not needed because tile is set to false.
    --set tiling overlap to 12
    --set tiling type to auto
end if


------------------------------------------------------------------------
--Properties corresponding to the controls in the Marks and Bleed
--panel of the Print dialog.
------------------------------------------------------------------------
--Set the following property to true to print all printer's marks.
--set all Printer Marks to true;
set use document bleed to print to false
--If use document bleed to print is true then setting any of the
-- bleed properties
--will result in an error.
set bleed bottom to myY2Offset
set bleed top to myY1Offset
set bleed inside to myX1Offset
set bleed outside to myX2Offset
--If any bleed area is greater than zero, then export the bleed marks.
if bleed bottom is equal to 0 and bleed top is equal to 0 and ¬
            bleed inside is equal to 0 and bleed outside is equal to 0 then
    set bleed marks to true
else
    set bleed marks to false
end if
set color bars to true
set crop marks to true
set include slug to print to false
set mark line weight to p125pt
set mark offset to 6
```

```
--set mark Type to default
set page information marks to true
set registration marks to true
-------------------------------------------------------------------------
--Properties corresponding to the controls in the Output panel
--of the Print dialog.
-------------------------------------------------------------------------
set negative to true
set color output to separations
--Note the lowercase "i" in "Builtin"
set trapping to application builtin
set screening to "175 lpi/2400 dpi"
set flip to none
--If trapping is on, attempting to set the following properties
--will generate an error.
if trapping is off then
    set print black to true
    set print cyan to true
    set print magenta to true
    set print yellow to true
end if
--Only change the ink angle and frequency when you want to override the
--screening set by the screening specified by the screening property.
--set black angle to 45
--set black frequency to 175
--set cyan angle to 15
--set cyan frequency to 175
--set magenta angle to 75
--set magenta frequency to 175
--set yellow angle to 0
--set yellow frequency to 175
--The following properties are not needed (because colorOutput
--is set to separations).
--set composite angle to 45


--set composite frequency to 175
--set simulate overprint to false
-------------------------------------------------------------------------
--Properties corresponding to the controls in the Graphics panel
--of the Print dialog.
-------------------------------------------------------------------------
set send image data to all image data
set font downloading to complete
try
    set download PPD fonts to true
end try
try
    set data format to binary
end try
try
    set PostScript level to level 3
end try
-------------------------------------------------------------------------
--Properties corresponding to the controls in the Color Management panel
--of the Print dialog.
-------------------------------------------------------------------------
--If the use color management property of color settings is false,
--attempting to set the following properties will return an error.
try
    set source space to use document
    set intent to use color settings
    set CRD to use document
    set profile to PostScript CMS
end try
-------------------------------------------------------------------------
```

```
        --Properties corresponding to the controls in the Advanced panel
        --of the Print dialog.
        ------------------------------------------------------------------------
        set OPI image replacement to false
        set omit bitmaps to false
        set omit EPS to false
        set omit PDF to false
        --The following line assumes that you have a flattener preset
        --named "high quality flattener".
        try
            set flattener preset name to "high quality flattener"
        end try
        set ignore spread overrides to false
    end tell
end tell
```

プリントプリセットを使用したプリント

プリントプリセットを使用してドキュメントをプリントするには、`print` コマンドでプリントプリセットを指定します。

```
tell application "Adobe InDesign CS6"
    --The following line assumes that you have defined a print preset
    --named "myPrintPreset".
    print active document using "myPrintPreset"
end tell
```

PDF としてのドキュメントの書き出し

InDesign スクリプティングでは、ページレイアウトドキュメントから PDF ファイルを作成する処理を完全に制御できます。

PDF の書き出し

次のスクリプトでは、現在の PDF 書き出しオプションを使用して、現在のドキュメントを PDF として書き出しています （完全なスクリプトについては、ExportPDF を参照してください）。

```
tell application "Adobe InDesign CS6"
    tell active document
        --You'll have to fill in a valid file path for your system.
        export format PDF type to "yukino:test.pdf" without showing options
    end tell
end tell
```

次のスクリプトでは、PDF 書き出しプリセットを使用して PDF を書き出しています （完全なスクリプトについては、ExportPDFWithPreset を参照してください）。

```
tell application "Adobe InDesign CS6"
    tell active document
        --You'll have to fill in a valid file path for your system and
        --a valid PDF export preset name.
        export format PDF type to "yukino:test.pdf" using PDF export preset
        "myTestPreset" without showing options
    end tell
end tell
```

PDF 書き出しオプションの設定

次のスクリプトでは、PDF 書き出しオプションを設定してから、書き出しを行っています（完全なスクリプトについては、ExportPDFWithOptions を参照してください）。

```
tell application "Adobe InDesign CS6"
    --Get the bleed amounts from the document's bleed.
    tell document preferences of active document
        set myX1Offset to document bleed inside or left offset
        set myY1Offset to document bleed top offset
        set myX2Offset to document bleed outside or right offset
        set myY2Offset to document bleed bottom offset
    end tell
    tell PDF export preferences
        --Basic PDF output options.
        set page range to all pages
        set acrobat compatibility to acrobat 6
        set export guides and grids to false
        set export layers to false
        set export nonprinting objects to false
        set export reader spreads to false
        set generate thumbnails to false
        try
            set ignore spread overrides to false
        end try
        set include bookmarks to true
        set include hyperlinks to true
        try
            set include ICC profiles to true
        end try
        set include slug with PDF to false
        set include structure to false
        set interactive elements option to do not include
        --Setting subset fonts below to zero disallows font subsetting
        --set subset fonts below to some other value to use font subsetting.
        set subset fonts below to 0
        --Bitmap compression/sampling/quality options.
        set color bitmap compression to zip
        set color bitmap quality to eight bit
        set color bitmap sampling to none
        --threshold to compress color is not needed in this example.
        --color bitmap sampling dpi is not needed when color bitmap sampling
        --is set to none.
        set grayscale bitmap compression to zip
        set grayscale bitmap quality to eight bit
        set grayscale bitmap sampling to none
        --threshold to compress gray is not needed in this example.
        --grayscale bitmap sampling dpi is not needed when grayscale bitmap
        --sampling is set to none.
        set monochrome bitmap compression to zip
        set monochrome bitmap sampling to none
        --threshold to compress monochrome is not needed in this example.
        --monochrome bitmap sampling dpi is not needed when monochrome bitmap
        --sampling is set to none.
        --Other compression options.
        set compression type to Compress None
        set compress text and line art to true
        set crop images to frames to true
        set optimize PDF to true
        --Printers marks and prepress options.
        set bleed bottom to myY2Offset
        set bleed top to myY1Offset
        set bleed inside to myX1Offset
        set bleed outside to myX2Offset
        --If any bleed area is greater than zero, then export the bleed marks.
```

```
        if bleed bottom is 0 and bleed top is 0 and bleed inside is 0 and
        bleed outside is 0 then
            set bleed marks to true
        else
            set bleed marks to false
        end if
        set color bars to true
        --Color tile size and gray tile size are not used
        --unless the compression method chosen is JPEG 2000.
        --set color tile size to 256
        --set Gray tile size to 256
        set crop marks to true
        set omit bitmaps to false
        set omit EPS to false
        set omit PDF to false
        set page information marks to true
        set page marks offset to "12 pt"
        set PDF color space to unchanged color space
        set PDF mark type to default


        set printer mark weight to p125pt
        set registration marks to true
        --simulate overprint is only available when the export standard
        --is PDF/X-1a or PDF/X-3
        --set simulate overprint to false
        set use document bleed with PDF to true
        --Set viewPDF to true to open the PDF in Acrobat or Adobe Reader.
        set view PDF to false
    end tell
    --Now export the document.
    tell active document
        --You'll have to fill in a valid file path for your system.
        export format PDF type to "yukino:test.pdf" without showing options
    end tell
end tell
```

グレースケール PDF の書き出し

カラーを含む InDesign ドキュメントは、カラー PDF またはグレースケール PDF に書き出すことができます。前述の 49 ページの「PDF 書き出しオプションの設定」節のスクリプトを変更し、グレースケールでの書き出しを行うには、`PDF color space` オプションを、`unchanged color space` から gray に変更します。次に例を示します。

```
tell PDF export preferences
    set PDF color space to gray
end tell
```

（カラーが含まれている小さいドキュメントを作成し、グレースケールで PDF に書き出す他のスクリプトについては、GreyscalePDFforIDS を参照してください）。

ページ範囲の PDF 書き出し

指定したページ範囲を PDF として書き出す方法を、次のスクリプトに示します （完全なスクリプトについては、ExportPageRangeAsPDF を参照してください）。

```
--Assumes you have a document open, and that that document
--contains at least 12 pages.
tell application "Adobe InDesign CS6"
    tell PDF export preferences
        --page range can be either all pages or a page range string
        --(just as you would enter it in the Print or Export PDF dialog box).
        set page range to "1, 3-6, 7, 9-11, 12"
    end tell
    tell active document
        --You'll have to fill in a valid file path for your system and
        --a valid PDF export preset name.
        export format PDF type to "yukino:test.pdf" using
        PDF export preset "myTestPreset" without showing options
    end tell
end tell
```

ページごとの PDF 書き出し

次のスクリプトでは、ドキュメントの各ページを個別の PDF ファイルとして書き出しています（完全なスクリプトについては、ExportEachPageAsPDF を参照してください）。

```
--Display a "choose folder" dialog box.
tell application "Adobe InDesign CS6"
    if (count documents) is not equal to 0 then
        my myChooseFolder()
    else
        display dialog "Please open a document and try again."
    end if
end tell
on myChooseFolder()
    set myFolder to choose folder with prompt "Choose a Folder"
    --Get the folder name (it'll be returned as a Unicode string)
    set myFolder to myFolder as string
    --Unofficial technique for changing Unicode folder name to plain text string.
    set myFolder to «class ktxt» of (myFolder as record)
    if myFolder is not equal to "" then
        my myExportPages(myFolder)
    end if
end myChooseFolder
on myExportPages(myFolder)
    tell application "Adobe InDesign CS6"
        set myDocument to active document
        set myDocumentName to name of myDocument
        set myDialog to make dialog with properties {name:"File Naming Options"}
        tell myDialog
            tell (make dialog column)
                tell (make dialog row)
                    make static text with properties {static label:"Base name:"}
                    set myBaseNameField to make text editbox with properties
                    {edit contents:myDocumentName, min width:160}
                end tell
            end tell
        end tell
        set myResult to show myDialog
        if myResult is true then
            --The name of the exported files will be the base name + the value
            --of the counter + ".pdf".
            set myBaseName to edit contents of myBaseNameField
            --Remove the dialog box from memory.
            destroy myDialog
            repeat with myCounter from 1 to (count pages in myDocument)
                set myPageName to name of page myCounter of myDocument
                set page range of PDF export preferences to name of page
                myCounter of myDocument
```

```
                --Generate a file path from the folder name, the base document
                --name, and the page name.
                --Replace any colons in the page name (e.g., "Sec1:1") so that
                --they don't cause problems with file naming.
                set myPageName to my myReplace(myPageName, ":", "_")
                set myFilePath to myFolder & myBaseName & "_" & myPageName & ".pdf"
                tell myDocument
                    --The export command will fail if you provide the file path
                    --as Unicode text--that's why we had to convert the folder name
                    --to plain text.
                    export format PDF type to myFilePath
                end tell
            end repeat
        else
            destroy myDialog
        end if
    end tell
end myExportPages
on myReplace(myString, myFindString, myChangeString)
    set AppleScript's text item delimiters to myFindString
    set myTextList to every text item of (myString as text)
    set AppleScript's text item delimiters to myChangeString
    set myString to myTextList as string
    set AppleScript's text item delimiters to ""
    return myString
end myReplace
```

インタラクティブ機能を使用した PDF の書き出し

次のスクリプトは、インタラクティブ機能を使用したドキュメントを PDF として書き出します （完全なスクリプトについては、ExportInteractivePDF を参照してください）。

```
--Given a document "myDocument" containing at least two spreads...
repeat with myCounter from 1 to (count spreads of myDocument)
    set page transition type of spread myCounter of myDocument to wipe transition
    set page transition direction of spread myCounter of myDocument to left to right
    set page transition duration of spread myCounter of myDocument to medium
end repeat
set myDesktopFolder to path to desktop as string
set myFile to myDesktopFolder & "InteractivePDF.pdf"
tell myDocument
    export format interactive PDF to myFile without showing options
end tell
```

PDF フォームとしての書き出し

フォームフィールドを含む PDF を作成するには、PDF として保存した InDesign ドキュメントを Acrobat で開いてフォームフィールドを適用するか、テキストフィールド、ラジオボタン、署名フィールドなどを直接 InDesign 上で適用し、Acrobat での編集作業を必要としない状態で PDF にフォームとして書き出します。

次のスクリプトでは、フォーム要素を InDesign ドキュメントに適用し、ドキュメントを PDF フォームとして書き出しています （完全なスクリプトについては、ExportInteractivePDFForm を参照してください）。

```
tell page 1 of myDocument
    --Create a textframe as firstname label
    set myTextFrame to make text frame with properties
        {geometric bounds:{15, 15, 20, 35}, contents:"FirstName: "}
    --Create a textbox as firstname input box
    set myTextBox to make text box with properties {geometric bounds:{15, 40, 20, 75}}
    --Create another textframe as lastname label
    set myTextFrame1 to make text frame with properties
        {geometric bounds:{30, 15, 25, 35}, contents:"LastName: "}
    --Create another textbox as lastname input box
    set myTextBox to make text box with properties {geometric bounds:{30, 40, 25, 75}}
    --Create a TextFrame to introduce the following checkbox
    set myTextFrame2 to make text frame with properties
        {geometric bounds:{15, 80, 20, 95}, contents:"Hobby:"}
    --Create some CheckBoxes
    set myCheckBox to make check box with properties
        {geometric bounds:{15, 100, 20, 105}, name:"Football"}
    set myTextFrame3 to make text frame with properties
        {geometric bounds:{15, 107, 20, 125}, contents:"Football"}
    set myCheckBox1 to make check box with properties
        {geometric bounds:{15, 130, 20, 135}, name:"Basketball"}
    set myTextFrame4 to make text frame with properties
        {geometric bounds:{15, 137, 20, 160}, contents:"Basketball"}
    set myCheckBox2 to make check box with properties
        {geometric bounds:{15, 165, 20, 170}, name:"Pingpong"}
    set myTextFrame5 to make text frame with properties
        {geometric bounds:{15, 172, 20, 193}, contents:"Pingpong"}
    --Create a button for submit
    set submitButton to make button with properties
        {geometric:{45, 15, 35, 35}, name:"Submit"}
end tell
tell state 1 of submitButton
    set myRightArrow1 to make polygon with properties {fill color:color "Green" of myDocument,
stroke color:"None"}
    set entire path of path 1 of myRightArrow1 to {{15, 35}, {35, 40}, {15, 45}}
end tell
--Add the Rollover state.
tell submitButton
    set myRolloverState1 to make state
end tell
tell myRolloverState1
    set myRolloverArrow1 to make polygon with properties {fill color:color "Green" of myDocument,
stroke color:"None"}
    set entire path of path 1 of myRolloverArrow1 to {{15, 35}, {35, 40}, {15, 45}}
    --Add a shadow to the polygon in the Rollover state.
end tell
tell drop shadow settings of fill transparency settings of myRolloverArrow1
    set mode to drop
    set angle to 90
    set x offset to 0
    set y offset to 0
    set size to 6
end tell
tell submitButton
    set myClickState1 to make state
end tell
tell myClickState1
    set myClickArrow1 to make polygon with properties {fill color:color "Blue" of myDocument,
stroke color:"None"}
    set entire path of path 1 of myClickArrow1 to {{15, 35}, {35, 40}, {15, 45}}
```

```
end tell
--Set the behavior for the button.
tell submitButton
    set submitForm to make submit form behaviors with properties {behavior event:mouse up}
end tell

tell page 1 of myDocument
    --Create a button for print
    set printButton to make button with properties {geometric:{45, 15, 35, 35}, name:"Submit"}
end tell
tell state 1 of printButton
   set myRightArrow2 to make polygon with properties {fill color:color "Red" of myDocument, stroke
color:"None"}
    set entire path of path 1 of myRightArrow2 to {{40, 35}, {60, 40}, {40, 45}}
end tell
--Add the Rollover state.
tell printButton
    set myRolloverState2 to make state
end tell
tell myRolloverState2
    set myRolloverArrow2 to make polygon with properties {fill color:color "Red" of myDocument,
stroke color:"None"}
    set entire path of path 1 of myRolloverArrow2 to {{40, 35}, {60, 40}, {40, 45}}
    --Add a shadow to the polygon in the Rollover state.
end tell
tell drop shadow settings of fill transparency settings of myRolloverArrow2
    set mode to drop
    set angle to 90
    set x offset to 0
    set y offset to 0
    set size to 6
end tell
tell printButton
    set myClickState2 to make state
end tell
tell myClickState2
    set myClickArrow2 to make polygon with properties {fill color:color "Blue" of myDocument,
stroke color:"None"}
    set entire path of path 1 of myClickArrow2 to {{40, 35}, {60, 40}, {40, 45}}
end tell
--Set the behavior for the button.
tell printButton
    set PrintForm to make print form behaviors with properties {behavior event:mouse up}
end tell
--Export the document to PDF.
set myDesktopFolder to path to desktop as string
set myFile to myDesktopFolder & "SubmitForm.pdf"
tell myDocument
    export format interactive PDF to myFile without showing options
end tell
```

EPS としてのページの書き出し

ドキュメントを EPS として書き出すと、ファイルの各ページが個別の EPS グラフィックとして保存されます（1 つの EPS には 1 ページしか書き出せないようになっています）。複数のページを書き出した場合、ファイル名にはページのインデックスが付加されます。ドキュメント内のページのインデックスが、ページの名前になるとは限りません（ページが含まれているセクションのセクションオプションによって定義されます）。

すべてのページの EPS 書き出し

次のスクリプトでは、アクティブなドキュメントのページが、1 つまたは複数の EPS ファイルに書き出されます （完全なスクリプトについては、ExportAsEPS を参照してください）。

```
tell application "Adobe InDesign CS6"
    set page range of EPS export preferences to all pages
    tell active document
        --You'll have to fill in a valid file name for your system.
        --Files will be named "myFile_01.eps", "myFile_02.eps", and so on.
        set myFileName to "yukino:myFile.eps"
        export format EPS type to myFileName without showing options
    end tell
end tell
```

ページ範囲の EPS 書き出し

EPS として書き出すページを制御するには、書き出しを行う前に、EPS export preferences の `page range` プロパティに、書き出すページのページ範囲を示す文字列を指定します （完全なスクリプトについては、ExportPageRangeAsEPS を参照してください）。

```
--Assumes you have a document open, and that that document
--contains at least 12 pages.
tell application "Adobe InDesign CS6"
    tell EPS export preferences
        --page range can be either all pages or a page range string
        --(just as you would enter it in the Print or Export EPS dialog box).
        set page range to "1, 3-6, 7, 9-11, 12"
    end tell
    tell active document
        --You'll have to fill in a valid file path for your system.
        export format EPS type to "yukino:test.eps" without showing options
    end tell
end tell
```

ファイル名を指定した EPS 書き出し

次のスクリプトでは、前述の例と比べてファイル名を細かく制御しながら各ページを EPS として書き出しています （完全なスクリプトについては、ExportEachPageAsEPS を参照してください）。

```
--Display a "choose folder" dialog box.
tell application "Adobe InDesign CS6"
    if (count documents) is not equal to 0 then
        my myChooseFolder()
    else
        display dialog "Please open a document and try again."
    end if
end tell
on myChooseFolder()
    set myFolder to choose folder with prompt "Choose a Folder"
    --Get the folder name (it'll be returned as a Unicode string)
    set myFolder to myFolder as string
    --Unofficial technique for changing Unicode folder name to plain text string.
    set myFolder to «class ktxt» of (myFolder as record)
    if myFolder is not equal to "" then
        my myExportPages(myFolder)
    end if
end myChooseFolder
on myExportPages(myFolder)
    tell application "Adobe InDesign CS6"
        set myDocument to active document
        set myDocumentName to name of myDocument
        set myDialog to make dialog with properties {name:"ExportPages"}
        tell myDialog
            tell (make dialog column)
                tell (make dialog row)
                    make static text with properties {static label:"Base Name:"}
                    set myBaseNameField to make text editbox with properties
```

```
                {edit contents:myDocumentName, min width:160}
            end tell
        end tell
    end tell
    set myResult to show myDialog
    if myResult is true then
        --The name of the exported files will be the base name + the
        --value of the counter + ".pdf".
        set myBaseName to edit contents of myBaseNameField
        --Remove the dialog box from memory.
        destroy myDialog
        repeat with myCounter from 1 to (count pages in myDocument)
            --Get the name of the page and assign it to the variable "myPageName"
            set myPageName to name of page myCounter of myDocument
            --Set the page range to the name of the specific page.
            set page range of EPS export preferences to myPageName
            --Generate a file path from the folder name, the base document
            --name, and the page name.
            --Replace any colons in the page name (e.g., "Sec1:1") so that
            --they don't cause problems with file naming.
            set myPageName to my myReplace(myPageName, ":", "_")
            set myFilePath to myFolder & myBaseName & "_" & myPageName & ".eps"
            tell myDocument
                export format EPS type to myFilePath without showing options
            end tell
        end repeat
    else
        destroy myDialog
    end if
end tell
end myExportPages
on myReplace(myString, myFindString, myChangeString)
    set AppleScript's text item delimiters to myFindString
    set myTextList to every text item of (myString as text)
    set AppleScript's text item delimiters to myChangeString
    set myString to myTextList as string
    set AppleScript's text item delimiters to ""
    return myString
end myReplace
```

EPub の書き出し

InDesign スクリプティングでは、ページレイアウトドキュメントから EPub ファイルを作成する処理を完全に制御できます。

現在のドキュメントの書き出し

次のスクリプトでは、デフォルトのオプションを使用して、現在のドキュメントを EPub として書き出しています（完全なスクリプトについては、ExportEPub を参照してください）。

```
set myDocument to document 1
tell myDocument
    export format EPUB to myFile without showing options
end tell
```

EPub ファイルの作成をユーザーがさらに細かく制御できるようにするには、3 番目のパラメーターに true を指定します。これにより、EPub 書き出しオプションダイアログボックスが表示されます。

EPub 書き出しオプションの設定

次のスクリプトでは、EPub 書き出しオプションを設定してから、書き出しを行っています （完全なスクリプトについては、ExportEPubWithOptions を参照してください）。

```
tell myDocument
    tell EPub export preferences
        --Apply image alignment to anchored object settings.
        set apply image alignment to anchored object settings to false

        --The unit of space.
        set space unit to css em
        --The unit of margin.
        set margin unit to css em

        --Bottom margin of the epub.
        set bottom margin to 5
        --Left margin of the epub.
        set left margin to 5
        --Right margin of the epub.
        set right margin to 5
        --Top margin of the epub.
        set top margin to 5

        --If true, break InDesign document into smaller pieces when generating epub.
        set break document to true
        -- The name of paragraph style to break InDesign document.
        set paragraph style name to ""

        --The buttet export option.
        set bullet export option to as text

        set CSS export option to embedded CSS
        set custom image size option to size relative to page width

        set embed font to true

        set epub cover to first page
        --This will take effect only when epub cover is set to external image.
        --set cover image file to "cover.jpg"

        set epub publisher to "Adobe Devtech"

        --The export order.
        set export order to layout order

        --If true, output footnote immediately after its paragraph.
        set footnote follow paragraph to false

        --If true, export epub in XHTML format. Otherwise, in DTBook format.
        set format to true
```

```
        set GIF options interlaced to true
        set GIF options palette to windows palette

        --The epub unique identifier, like ISBN.
        set id to "123"

        --Ignore object level image conversion settings.
        set ignore object conversion settings to true

        --Alignment applied to images.
        set image alignment to align center
        --The file format to use for converted images.
        set image conversion to automatic


        set image export resolution to ppi 150
        --Image page break settings to be used with objects.
        set image page break to page break after
        --Space After applied to images.
        set image space after to 2
        --Space Before applied to images.
        set image space before to 2

        --If true, include CSS definition.
        set include CSS definition to true
        --If true, output document metadata into epub.
        set include document metadata to true

        set JPEG options format to baseline encoding
        set JPEG options quality to high

        --The PNG compression level.
        set level to 5

        set numbered list export option to as text

        --If true, format image based on layout appearence.
        set preserve layout appearence to true
        --If true, output local style override.
        set preserve local override to true

        set strip soft return to true
        --If true, image page break settings will be used in objects.
        set use image page break to true
        --Use InDesign TOC style to generate epub TOC.
        set use toc style to true

        set view document after export to false
    end tell

    export format EPUB to myFile without showing options
end tell
```

4 レイヤーの操作

章の更新ステータス	
CS6	変更なし

InDesign のレイヤーは、レイアウト内のオブジェクトの重なり順を制御する上で重要な役割を果たします。レイヤーは、互いに上へと重ね合わせられている複数枚の透明な水平面と考えることができます。また、同じ種類のコンテンツを特定のレイヤーや一連のレイヤーに配置して、レイヤーを構成用のツールとして使用することもできます。

ドキュメントには 1 つ以上のレイヤーを含めることができますが、各ドキュメントには少なくとも 1 つのレイヤーが含まれます。レイヤーはドキュメントと同じ幅で、特定のページやスプレッドに関連付けられることはありません。

この章では、InDesign レイアウトのレイヤーに関連するスクリプティングテクニックと、レイヤーに関する一般的な操作について説明します。

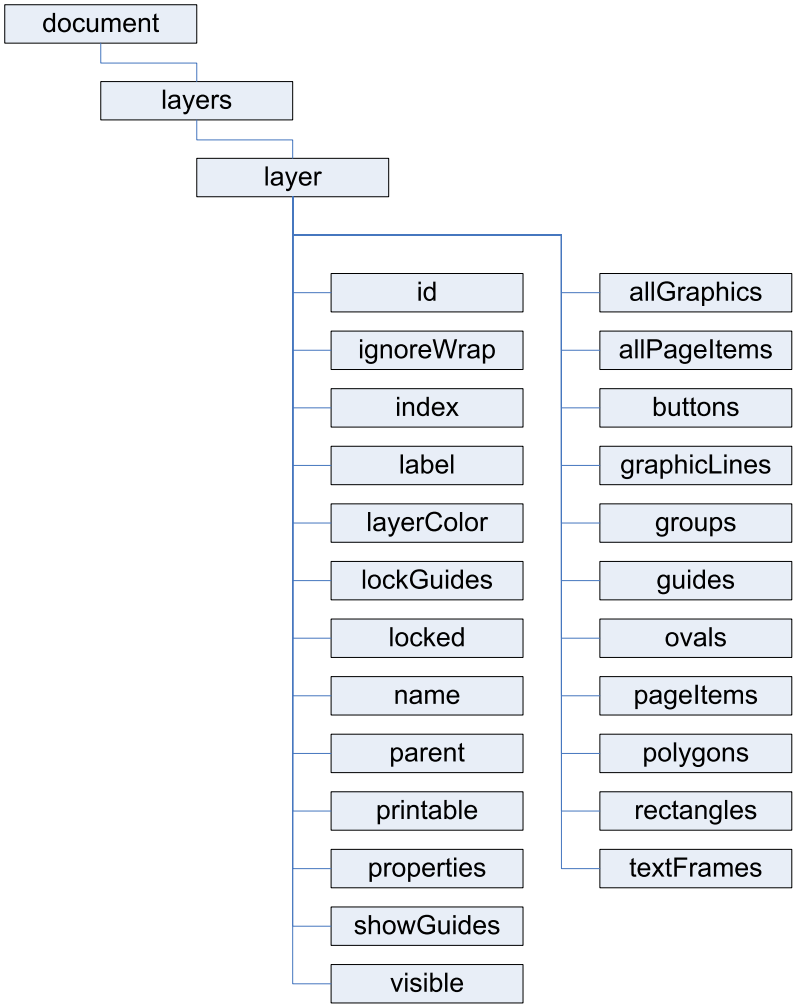
レイヤーオブジェクトモデルについて

次の図に、レイヤーオブジェクトモデルを示します。図を参照するにあたって、次の点に注意してください。

- 図は、ドキュメントのオブジェクト階層におけるレイヤーの位置と内容に焦点を合わせています。スクリプトでレイヤーの中身を操作するその他の方法についてすべてを図式化しようとしていません（例えば、レイヤーオブジェクト内にテキストフレームオブジェクトがあることがわかりますが、ストーリーやテキストオブジェクト、ページ、スプレッドからもテキストフレームオブジェクトへの参照を取得することができます）。
- オブジェクト名は JavaScript 形式のものですが、オブジェクト階層はすべてのスクリプト言語で同じです。
- 左側の列にレイヤーの基本的なプロパティを、右側にレイヤーオブジェクトに含めることができるオブジェクトを示しています。

重要な点として、ページアイテムのコレクションと `all page items` のコレクションの違いに注意してください。ページアイテムのコレクションは、レイヤー内の最上位レベルのページアイテムのみが含まれたコレクションです。例えば、ページアイテムがグループに含まれている場合、`page items` のコレクションにこのページアイテムは現れません。これに対し、`all page items` のコレクションは、レイヤーに割り当てられたすべてのページアイテムが統合されたコレクションです。オブジェクト階層における位置は関係ありません。レイヤー上のグループに含まれるページアイテムは、`all page items` のコレクションには現れます。

同様に、`all graphics` プロパティには、オブジェクト階層における位置にかかわらず、レイヤーに割り当てられたページアイテムに含まれているすべてのグラフィックが含まれます。



レイヤーのスクリプト操作

InDesign のユーザーインターフェイスでは、レイヤーパネルを使用して、レイヤーの追加、削除、並べ替え、複製、結合を行います。また、レイヤーパネルのレイヤープロキシをドラッグ＆ドロップして、選択しているページアイテムを割り当てるレイヤーを変更することもできます （オブジェクトをレイヤーに割り当てる方法について詳しくは、InDesign ヘルプを参照してください）。ここでは、InDesign スクリプティングでこれらの操作を実行する方法について説明します。

レイヤーの作成

新規レイヤーを作成する方法を、次のスクリプトに示します （完全なスクリプトについては、AddLayer を参照してください）。

```
--Given a document "myDocument"...
tell myDocument
    set myLayer to make layer
end tell
```

新規レイヤーを作成した場合、ドキュメント内のすべてのレイヤーの一番上に配置されます。

レイヤーの参照

InDesign スクリプティングでは、いくつかの方法でレイヤーオブジェクトを参照できます。ここでは、レイヤーを参照する一般的な方法について説明します。

アクティブなレイヤーの取得

アクティブなレイヤーとは、新しいオブジェクトが作成されるレイヤーです。スクリプトを使用してアクティブなレイヤーを取得できます。次のスクリプトにその例を示します （完全なスクリプトについては、ActiveLayer を参照してください）。

```
--Given a document "myDocument"...
tell myDocument
    set myLayer to active layer
end tell
```

レイヤーインデックスによるレイヤーの参照

ドキュメントのレイヤーコレクションに含まれるレイヤーのインデックスを使用して、レイヤーへの参照を取得できます。レイヤーのインデックスを使用して、レイヤーに対して繰り返し処理を行う方法を、次のスクリプトに示します （完全なスクリプトについては、HideOtherLayers を参照してください）。

```
--Given a document "myDocument"...
tell myDocument
    set myTargetLayer to active layer
    set myLayerName to name of myTargetLayer
    repeat with myCounter from 1 to (count layers)
    --If the layer is not the target layer, hide it.
    set myName to name of layer myCounter
        if myName is not equal to myLayerName then
            set visible of layer myCounter of myDocument to false
        end if
    end repeat
end tell
```

ドキュメントのレイヤーコレクションに含まれるレイヤーを参照するときには、負の数も使用できます。レイヤー -1 は、コレクション内の最後（一番下）のレイヤーを参照します。

レイヤー名によるレイヤーの参照

レイヤーの名前を使用してレイヤーへの参照を取得することもできます。次のスクリプトにその例を示します （完全なスクリプトについては、LayerName を参照してください）。

```
set myLayer to layer "Text Layer" of document 1
```

レイヤーの削除

特定のドキュメントからレイヤーを削除するには、`delete` メソッドを使用します。次のスクリプトにその例を示します （完全なスクリプトについては、DeleteLayer を参照してください）。ドキュメント内の最後に残ったレイヤーは削除できません。

```
--Given a document "myDocument" containing a layer named "Delete This Layer"...
set myLayer to layer "Delete This Layer" of myDocument
tell myLayer
    delete
end tell
```

レイヤーの移動

ドキュメント内のレイヤーの重なり順を変更するには、move メソッドを使用します。次のスクリプトにその例を示します （完全なスクリプトについては、MoveLayer を参照してください）。

```
--Given a document "myDocument" containing at least two layers...
set myLayerA to layer 1 of myDocument
set myLayerB to layer 2 of myDocument
tell myLayer1
    move to after myLayer2
end tell
```

レイヤーの複製

レイヤーのコピーを作成するには、duplicate メソッドを使用します。次のスクリプトにその例を示します（完全なスクリプトについては、DuplicateLayer を参照してください）。

```
-Given a layer "myLayer"...
tell myLayer
    set myNewLayer to duplicate
end tell
```

レイヤーの結合

2 つ以上のレイヤーを、レイヤーに割り当てられているページアイテムも含めて 1 つのレイヤーに結合する方法を、次のスクリプトに示します （完全なスクリプトについては、MergeLayers を参照してください）。

```
--Given the layers "myLayer1" and "myLayer2"...
tell myLayer1
    merge with myLayer2
end tell
```

レイヤーへのページアイテムの割り当て

レイヤーにページアイテムを割り当てるには、ページアイテムを作成するときにレイヤーを参照するか、既存のページアイテムの item layer プロパティを設定します。この両方の方法で、レイヤーにページアイテムを割り当てる方法を、次のスクリプトに示します （完全なスクリプトについては、AssignPageItemsToLayers を参照してください）。

```
--Given a reference to a page "myPage," and a document "myDocument"...
tell myPage
    --Create a text frame on a layer named "TextFrames"
    set myTextFrame to make text frame with properties
    {item layer:layer "TextFrames", geometric bounds:{72, 72, 144, 144}}
    --Create a rectangle on the current target layer.
    set myRectangle to make rectangle with properties
    {geometric bounds:{72, 144, 144, 216}}
    --Move the rectangle to a specific layer.
    set item layer of myRectangle to layer "Rectangles" of myDocument
    --Create a series of ovals.
    repeat with myCounter from 72 to 172 by 10
        make oval with properties {geometric bounds:{216, myCounter, 226,
        myCounter + 10}}
    end repeat
    --Move all of the ovals on the page to a specific layer.
    repeat with myCounter from 1 to (count ovals)
        set item layer of oval myCounter to layer "Ovals" of myDocument
    end repeat
end tell
```

レイヤーのプロパティの設定

レイヤーのプロパティは、レイヤーのレイヤー名、カラー、表示／非表示などの属性を制御します。ここでは、レイヤーのプロパティを操作する方法について説明します。

レイヤーの基本的なプロパティの設定

レイヤーの基本的なプロパティには、レイヤーの名前、レイヤーのハイライトカラー、レイヤーの表示／非表示、レイヤー上のテキストオブジェクトがテキストの回り込み設定を無視するかどうかの設定などがあります。これらの基本的なレイヤーのプロパティを設定する方法を、次のスクリプトに示します （完全なスクリプトについては、BasicLayerProperties を参照してください）。

```
--Given a document "myDocument"...
tell myDocument
    set myLayer to make layer
    set name of myLayer to "myLayer"
    set layer color of myLayer to charcoal
    set ignore wrap of myLayer to false
    set visible of myLayer to true
end tell
```

レイヤーガイドの操作

ページアイテムと同じように、特定のレイヤーにガイドを割り当てることができます。レイヤーのガイドは、表示／非表示を選択でき、さらにロックをかけたり解除したりすることもできます。レイヤーのガイドを操作する方法を、次のスクリプトに示します （完全なスクリプトについては、LayerGuides を参照してください）。

```
--Given a document "myDocument" and a page "myPage" containing at least one guide...
tell myDocument
    set myLayer to make layer
    --Move all of the guides on the page to the new layer.
    tell myPage
        repeat with myCounter from (count guides) to 1 by -1
            set item layer of guide myCounter to myLayer
        end repeat
    end tell
    set lock guides of myLayer to true
    set show guides of myLayer to true
end tell
```

レイヤーのプリントと表示／非表示の制御

レイヤー上のオブジェクトのプリントと表示／非表示を制御できます。次のスクリプトにその例を示します（完全なスクリプトについては、LayerControl を参照してください）。

```
--Given a document "myDocument" containing layers named "Background,"
--"Language A,", "Language B," and "Language C," export the "Background"
--layer and each "Language" layer to PDF as separate PDF files...
set myList to {"A", "B", "C"}
set myPath to path to desktop as string
tell myDocument
    repeat with myCounter from 1 to 3
        set myVersion to "Language " & item myCounter of myList
        repeat with myLayerCounter from 1 to (count layers)
            if name of layer myLayerCounter is equal to myVersion or
            name of layer myLayerCounter is equal to "Background" then
                set visible of layer myLayerCounter to true
                set printable of layer myLayerCounter to true
            else
                set visible of layer myLayerCounter to false
                set printable of layer myLayerCounter to false
            end if
        end repeat
        set myFilePath to myPath & myVersion & ".pdf"
        export to myFilePath format PDF type
    end repeat
end tell
```

レイヤーのロック

レイヤーにはロックをかけることができます。ロックをかけると、レイヤー上のページアイテムは編集できなくなります。レイヤーのロックをかける方法と解除する方法を、次のスクリプトに示します （完全なスクリプトについては、LockLayersBelow を参照してください）。

```
--Given a document "myDocument"...
tell myDocument
    set myLayer to active layer
    repeat with myCounter from ((index of myLayer) + 1) to (count layers)
        set locked of layer myCounter to true
    end repeat
end tell
```

5 ページアイテムの操作

章の更新ステータス	
CS6	変更なし

この章では、InDesign レイアウトに表示されるページアイテム（長方形、楕円形、グラフィックの線、多角形、テキストフレーム、ボタン、グループなど）に関連するスクリプティングテクニックを紹介します。

この章の内容は、次のとおりです。

- ページアイテムの作成
- ページアイテムの形状
- パスとパスポイントの操作
- グループの作成
- ページアイテムの複製と移動
- ページアイテムの変形
- アーティクルの使用

ページアイテムの作成

InDesign レイアウトのページアイテムは階層的に配置され、なんらかのコンテナオブジェクト内に表示されます。例えば、スプレッド、ページ、その他のページアイテム、グループおよびテキスト文字はすべて、ページアイテムを格納できるコンテナオブジェクトです。InDesign スクリプトオブジェクトモデルにあるコンテナのこの階層は、InDesign のユーザーインターフェイスの階層と同じです。つまり、あるページ上で長方形ツールをドラッグして長方形を作成する操作は、そのページが、作成する長方形のコンテナ（親）になるよう指定していることになります。楕円形を多角形の中に貼り付ける操作は、多角形が楕円形の親になるよう指定していることになります。多角形は親でもありますが、その親であるページにとっては子オブジェクトとなります。

MakeRectangle スクリプトに示すように、一般に、新しいページアイテムの作成は簡単です。ページアイテムを作成するには、そのページアイテムを格納するオブジェクトを指定します。

```
--Given a page "myPage", create a new rectangle at the default size and location...
tell myPage
    set myRectangle to make rectangle
end tell
```

前述のスクリプトでは、新規ドキュメントの最初のページに新しい長方形が作成されます。この長方形は、デフォルトの位置（ページの左上隅付近）に、デフォルトのサイズ（1 辺が約 10 ポイントの正方形）で表示されます。MakeRectangleWithProperties スクリプトに示すように、長方形の移動や大きさの変更は、いずれもその境界線のプロパティに新しい値を指定することで実行されます。

```
--Given a page "myPage", create a new rectangle and specify its size and location...
tell myPage
    set myRectangle to make rectangle with properties
    {geometric bounds:{72, 72, 144, 144}}
end tell
```

ページアイテムの種類

「汎用的な」ページアイテムは作成できない点に注意することが重要です。特定の種類（長方形、楕円形、グラフィックの線、多角形、テキストフレームまたはボタン）のページアイテムを作成する必要があります。ページアイテムの種類は、そのページアイテムの形状を変更したときにも、変化することに注意してください。例えば、長方形は常に、4 つのパスポイントと 90° の内角を備えた閉じた単一のパスで構成されています。しかし、1 つのパスポイントの位置を変更したり、別のパスを追加したりすると、ページアイテムの種類は多角形に変化します。パスを開いて、4 つのパスポイントの 2 つを削除すると、ページアイテムの種類はグラフィックの線に変化します。長方形、楕円形、グラフィックの線、多角形などの種類は、次の要素のみで定義されます。

- オブジェクトにあるパスの数。複数のパスがあるページアイテムは多角形です。
- オブジェクトの最初のパスにあるポイントの数と位置。

ページアイテムの種類を判断するには、次の例を使用します。

```
set myPageItemType to class of myPageItem
```

前述の結果は、ページアイテムの種類を表す文字列になります。

ページアイテムの種類の取得

汎用的なページアイテムへの参照があるときに、そのページアイテムの種類を確認する場合は、class of を使用して特定の種類を取得します。

```
--Given a generic page item "myPageItem"...
set myType to class of myPageItem
display dialog myType
```

ページアイテムの参照

特定のコンテナ（ドキュメント、レイヤー、ページ、スプレッド、グループ、テキストフレーム、ページアイテムなど）の内部にあるページアイテムを参照する場合は、コンテナオブジェクトの `page items` コレクションを使用します。これによって、オブジェクトの内部にある最上レベルのページアイテムのコレクションを参照できます。次に例を示します。

```
set myPageItems to page items of page 1 of document 1
```

結果のコレクション（`myPageItems`）に含まれないものとしては、グループ内部のオブジェクト（コレクションにグループが格納されている場合）、他のページアイテムの内部にあるオブジェクト（コレクションに親ページアイテムが格納されている場合）またはテキストフレーム内のページアイテムがあります。他のページアイテムの内部にネストされているアイテムなど、所定のコンテナにあるすべてのアイテムについて参照を取得するには、`all page items` プロパティを使用します。

```
set myAllPageItems to all page items of page 1 of document 1
```

結果のコレクション（`myAllPageItems`）には、階層内の位置に関係なく、ページ上のすべてのオブジェクトが含まれます。

ページアイテムを参照するもう 1 つの方法は、そのアイテムの label プロパティを使用する方法です。これは、他のオブジェクトの name プロパティ（段落スタイルやレイヤーなど）を使用するのと同じです。次のスクリプトに、ラベルが `myLabel` に設定されているページアイテムの配列を取得する例を示します。

```
set myPageItems to page items whose label is "myLabel" of page 1 of document 1
```

ページ上のページアイテムに指定されたラベルがない場合は、空の配列が返されます。

ページアイテムの形状

ページアイテムに関する作業を行うときは、InDesign のページアイテムの位置とシェイプの指定において、定規と測定値がどのように相互に機能するかを理解している必要があります。InDesign のユーザーインターフェイスのコントロールパネルを使用する場合、InDesign の座標系を既に理解していると想定されますが、要約を簡潔に示しておきます。

- オブジェクトは、定規に表示される座標との相対位置で形成されます。
- 原点をドラッグしたり、定規の基点を変更して原点の位置を変更したりすると、定規の座標は変化します。
- ページアイテムは 1 つ以上のパスで構成され、パスは 2 つ以上のパスポイントで構成されます。パスは、開いた状態または閉じた状態の場合があります。
- パスポイントには、1 つのアンカーポイント（ポイント自体の位置）と、2 つのコントロールハンドル（パスポイントの前にある線区分のカーブを制御する左方向線と、ポイントの後にある線区分のカーブを制御する右方向線）があります。これらの各プロパティには、（x, y）形式の配列があります（x はポイントの水平方向の位置、y は垂直方向の位置）。この配列には、ポイントまたはコントロールハンドルの位置が、現在の定規の座標で格納されます。

前述の内容は、スクリプトでページアイテムを形成する必要がある場合は、原点の位置も制御する必要があり、場合によっては使用時の測定単位を設定する必要があることを意味します。

パスとパスポイントの操作

簡単なページアイテムでは、ほとんどの場合、オブジェクトのシェイプを定義するパスとパスポイントについて考慮する必要はありません。長方形、楕円形、テキストフレームは、この章で前述した例が示すように、それぞれの境界線を指定することで作成できます。

ただし、パスポイントの位置を指定してパスのシェイプを形成または変更することが必要になる場合があります。この場合は、パスの各パスポイントのアンカーポイント、左方向線および右方向線を個別に設定するか（DrawRegularPolygon_Slow スクリプトを参照してください）、パスの entire path プロパティを使用してパスポイントのすべての位置を一度に設定できます（DrawRegularPolygon_Fast スクリプトを参照してください）。後者のアプローチのほうが高速です。

entire path プロパティに使用する配列のアイテムには、アンカーポイントのみを指定するか、アンカーポイントとコントロールハンドルを指定できます。次に、アンカーポイントの位置のみが指定されている配列の例を示します。

```
{{x1, y1}, {x2, y2}, ...}
```

x と y には、アンカーの位置を指定します。

次に、パスポイントがすべて指定されている配列（つまり、左方向線、アンカー、右方向線が、順に指定されている配列）の例を示します。

```
{{{xL1, YL1}, {x1, y1}, {xR1, yR1}}, {{xL2, YL2}, {x2, y2}, {xR2, yR2}}, ...}
```

xL と yL には左方向線を、x と y にはアンカーポイントを、xR と yR には右方向線を指定します。

次の例のように、2 つのアプローチを組み合わせることもできます。

```
{{{xL1, YL1}, {x1, y1}, {xR1, yR1}}, {x2, y2}, ...}
```

対象となるパスのポイントの数と、配列に指定するポイントの数は、同じである必要はありません。entire path プロパティに配列が適用される際は、パスに対してポイントが自動的に追加または削除されます。

次の AddPathPoint スクリプトは、entire path プロパティを使用せずにパスポイントをパスに追加する方法を示しています。

```
--Given a graphic line "myGraphicLine"...
tell path 1 of myGraphicLine
    set myPathPoint to add path point
end tell
--Move the path point to a specific location.
set anchor of myPathPoint to {144, 144}
```

次の DeletePathPoint スクリプトは、パスからパスポイントを削除する方法を示しています。

```
--Given a polygon "myPolygon", remove the
--last path point in the first path.
tell path 1 of myPolgon
    delete path point -1
end tell
```

ページアイテムのグループ化

InDesign のユーザーインターフェイスでページアイテムのグループを作成するには、ページアイテムを選択し、次に、オブジェクトメニューから「グループ」を選択します（または、対応するキーボードショートカットを押します）。InDesign スクリプティングでは、次の Group スクリプトのように、グループ化するページアイテムを格納しているオブジェクト（通常はページやスプレッド）に指示をして、ページアイテムをグループ化します。

```
--Given a page "myPage" containing at least two ovals and two rectangles...
tell myPage
    set myRectangleA to rectangle 1
    set myRectangleB to rectangle 2
    set Oval to oval 1
    set Oval to oval 2
    --Group the items.
    make group with properties{group items:
    {myRectangleA, myRectangleB, myOvalA, myOvalB})
end tell
```

グループを解除するには、次の Ungroup スクリプトのように、グループ解除するグループ自体に指示をします。

```
--Given a group "myGroup"...
tell myGroup
set myPageItems to ungroup
end tell
```

グループ内のページアイテムのシェイプやフォーマット、コンテンツを変更する際、グループの解除は不要です。代わりに、他のページアイテムの場合と同様に、変更するページアイテムへの参照を取得するだけです。

ページアイテムの複製と移動

InDesign のユーザーインターフェイスでは、ページアイテムを選択して新しい位置にドラッグすることで、ページアイテムを移動できます。ページアイテムのコピーは、コピーして貼り付ける方法や、Option または Alt キーを押しながらオブジェクトをドラッグする方法、編集メニューから「複製」、「元の位置にペースト」または「繰り返し複製」を選択する方法で作成することもできます。InDesign スクリプティングでは、ページアイテムの位置を変更する場合は move メソッドを使用し、ページアイテムのコピーを作成する（必要な場合はそのコピーを別の位置に移動する）場合は duplicate メソッドを使用できます。

この move メソッドには、オプションの 2 つのパラメーター move to と move by のいずれかを指定できます。この 2 つのパラメーターは、水平値と垂直値の 2 つの測定単位の配列で構成されています。move to には、原点の現在の位置と相対的で、配列に指定されている位置への絶対的な移動を指定します。move by には、現在のページアイテム自体の位置から相対的に、ページアイテムをどの程度移動するか（距離）を指定します。次の Move スクリプトに、これらの 2 つのアプローチの相違を示します。

```
--Given a reference to a rectangle "myRectangle"...
--Move the rectangle to the location (12, 12).
--Absolute move:
tell myRectangle
    move to {12, 12}
end tell
--Move the rectangle *by* 12 points horizontally, 12 points vertically.
--Relative move:
tell myRectangle
    move by {12, 12}
end tell
--Move the rectangle to another page (rectangle appears at (0,0);
tell document 1
set myPage to make page
end tell
tell myRectangle
    move to myPage
end tell
--To move a page item to another document, use the duplicate method.
```

move メソッドによってオブジェクトが実際に移動することに注意してください。別のドキュメントに移動したページアイテムは、元のドキュメントから削除されます。元のオブジェクトを残したままオブジェクトを別の位置に動かすには、duplicate メソッドを使用します（次の例を参照してください）。

ページアイテムのコピーを作成するには、duplicate メソッドを使用します。デフォルトでは、duplicate メソッドによって、元のオブジェクトと同じ位置にオブジェクトの「クローン」が作成されます。複製したオブジェクトを新しい位置に移動（同じドキュメント内の他のページへの移動や完全に別のドキュメントへの移動など）するには、duplicate メソッドでオプションのパラメーターを使用します。

```
--Given a reference to a rectangle "myRectangle"...
--Duplicate the rectangle and move the
--duplicate to the location (12, 12).
--Absolute move:
tell myRectangle
    set myDuplicate to duplicate to {12, 12}
end tell
--Duplicate the rectangle and move the duplicate *by* 12
--points horizontally, 12 points vertically.
--Relative move:
tell myRectangle
    set myDuplicate to duplicate by {12, 12}
end tell
--Duplicate the rectangle to another page (rectangle appears at (0,0).
tell document 1
    set myPage to make page
end tell
tell myRectangle
     set myDuplicate do duplicate to myPage
end tell
--Duplicate the rectangle to another document.
set myDocument to make document
tell myRectangle
    myDuplicate to page 1 of myDocument
end tell
```

InDesign スクリプティングではコピーして貼り付ける方法も使用できますが、この方法を使用するスクリプトの場合、（コピーする）オブジェクトを選択し、（貼り付ける際は）現在の表示状態に影響される状態で、貼り付け対象要素の位置を設定する必要があります。これは、コピーして貼り付ける方法を使用するスクリプトは、複製して移動する方法を使用するスクリプトよりも確実性が低い（失敗する可能性が高い）ことを意味します。可能な限り、現在の表示や選択状態に依存しないスクリプトを記述してください。

複合パスの作成

InDesign では、オブジェクト／パス／複合パスを作成メニューオプションを使用して、2 つ以上のページアイテムのパスを、複数のパスが含まれる単一のページアイテムに結合することができます。InDesign スクリプティングでこの処理を行うには、ページアイテムの make compound path コマンドを使用します。次のスクリプトにその例を示します（完全なスクリプトについては、MakeCompoundPath スクリプトを参照してください）。

```
--Given a rectangle "myRectangle" and an Oval "myOval"...
tell myRectangle
    make compound path with myOval
end tell
```

複合パスを作成すると、複合パスの作成に使用したオブジェクトの種類に関係なく、結果として形成されるオブジェクトの種類は多角形になります。

複合パスを解除して、複合パスの各パスを個別のページアイテムに変換するには、ページアイテムの release compound path コマンドを使用します。次のスクリプトにその例を示します（完全なスクリプトについては、ReleaseCompoundPath スクリプトを参照してください）。

```
--Given a polygon "myPolygon"...
tell myPolygon
    set myPageItems to release compound path
end tell
```

パスファインダー機能の使用

InDesign のパスファインダー機能には、InDesign ページの複数ページアイテム間の関係を操作する方法が用意されています。ページアイテムのパスを結合したり、あるページアイテムの領域を別のページアイテムから取り除いたり、2 つ以上のページアイテムが交差している領域から新しいページアイテムを作成したりすることができます。すべてのページアイテムでは、パスファインダー機能に関連する AddPath、ExcludeOverlapPath、IntersectPath、MinusBack および SubtractPath の各メソッドがサポートされています。

パスファインダーのすべてのメソッドは同じように機能します。ユーザーは、操作の基本として使用するページアイテムのリストを用意します（ユーザーインターフェイスでパスファインダー操作を選択する前に、一連のページアイテムを選択する場合と同様です）。

いずれかのパスファインダー操作を適用すると、オブジェクトの種類が変更される可能性が非常に高いことに注意してください。操作の結果、1 つまたは複数のパスに存在するポイントの数が幾つになるか、そしてどの位置に来るかによって、オブジェクトの種類が決まります。

2 つのページアイテムを 1 つのページアイテムに結合するには、例えば、次のスクリプトに示すようなアプローチを使用します（完全なスクリプトについては、AddPath を参照してください）。

```
--Given a rectangle "myRectangle" and an Oval "myOval"...
tell myRectangle
    add path with myOval
end tell
```

exclude overlap path コマンドは、重なり合った 2 つ以上のページアイテムの交差していない領域に基づいて新しいパスを作成します。次のスクリプトにその例を示します（完全なスクリプトについては、ExcludeOverlapPath スクリプトを参照してください）。

```
--Given a rectangle "myRectangle" and an Oval "myOval"...
tell myRectangle
    exclude overlap path with myOval
end tell
```

intersect path コマンドは、2 つ以上のページアイテムの交差している領域から新しいページアイテムを作成します。次のスクリプトにその例を示します（完全なスクリプトについては、IntersectPath を参照してください）。

```
--Given a rectangle "myRectangle" and an Oval "myOval"...
tell myRectangle
    intersect path with myOval
end tell
```

minus back コマンドは、最背面のオブジェクトの前面にある 1 つまたは複数のページアイテムから、最背面のオブジェクトと交差している領域を取り除きます。次のスクリプトにその例を示します（完全なスクリプトについては、MinusBack を参照してください）。

```
--Given a rectangle "myRectangle" and an Oval "myOval"...
tell myRectangle
    minus back with myOval
end tell
```

subtract path コマンドは、最前面のオブジェクトの背面にある 1 つまたは複数のページアイテムから、最前面のオブジェクトと交差している領域を取り除きます。次のスクリプトにその例を示します（完全なスクリプトについては、SubtractPath を参照してください）。

```
--Given a rectangle "myRectangle" and an Oval "myOval"...
tell myOval
    subtract with myRectangle
end tell
```

ページアイテムのシェイプの変換

InDesign のページアイテムは、オブジェクト／シェイプを変換メニューで表示されるオプションを使用するか、パスファインダーパネル（ウィンドウ／オブジェクトとレイアウト／パスファインダー）を使用して、他のシェイプに変換できます。InDesign スクリプティングのページアイテムでは、convert shape コマンドがサポートされています。次のスクリプトにその例を示します（完全なスクリプトについては、ConvertShape を参照してください）。

```
--Given a rectangle "myRectangle"...
tell myRectangle
    convert shape given convert to rounded rectangle
end tell
```

convert shape コマンドには、反転パスの開閉方法も用意されています。次のスクリプトにその例を示します（完全なスクリプトについては、OpenPath を参照してください）。

```
--Given a rectangle "myRectangle"...
tell myRectangle
    convert shape given convert to open path
end tell
```

ページアイテムの配置

InDesign レイアウトのページアイテムは、レイヤー内での重なり順を調整することによって、前面または背面に相互に配置したり、別のレイヤーに配置したりできます。次のスクリプトに、レイヤー内でオブジェクトを前面または背面に移動する方法と、オブジェクトの相対的な重なり順を制御する方法を示します（完全なスクリプトについては、StackingOrder を参照してください）。

```
--Given a rectangle "myRectangle" and an oval "myOval",
--where "myOval" is in front of "myRectangle", bring
--the rectangle to the front...
tell myRectangle
bring to front
end tell
```

ページアイテムを作成する場合は、そのページアイテムのレイヤーを指定できますが、レイヤー間でページアイテムを移動することもできます。これを実行する場合は、ページアイテムの item layer プロパティが重要

な役割を果たします。次のスクリプトにその例を示します（完全なスクリプトについては、ItemLayer を参照してください）。

```
--Given a rectangle "myRectangle" and a layer "myLayer",
--send the rectangle to the layer...
tell myRectangle
    set item layer to layer "myLayer" of document 1
end tell
```

ドキュメント内でのレイヤーの重なり順は、レイヤー自体の move コマンドを使用して変更することもできます。次のスクリプトにその例を示します（完全なスクリプトについては、MoveLayer を参照してください）。

```
--Given a layer "myLayer", move the layer behind
--the default layer.
tell myLayer
    move to end of layers of document 1
end tell
```

ページアイテムの変形

変形には、拡大・縮小、回転、シアー（歪み）、移動（または変換）があります。スクリプティングでは、`transform` メソッドを使用して変形を適用します。InDesign CS3（5.0）より前の InDesign バージョンで使用されていた `resize`、`rotate`、`shear` の各メソッドは、この 1 つのメソッドに置き換えられました。

transform メソッドの使用

`transform` メソッドには、オブジェクトに適用する変形または一連の変形を定義する変形マトリックス（`transformation matrix`）オブジェクトが必要です。変形マトリックスには、定規、回転、シアー、変換などの操作の組み合わせを指定できます。

オブジェクトに変形を適用する順序は重要です。異なる順序で変形を適用すると、まったく異なる結果となる可能性があります。

オブジェクトを変形するには、次の 2 つの手順に従います。

1. 変形マトリックスを作成します。
2. この変形マトリックスを `transform` メソッドを使用してオブジェクトに適用します。適用する際は、変形が行われる座標系も指定します。座標系について詳しくは、75 ページの「座標スペース」を参照してください。さらに、変形の中心、つまり変形の基点を指定します。変形の基点について詳しくは、76 ページの「変形の基点」を参照してください。

次のスクリプトに、ページアイテムを変形する基本的なプロセスの例を示します （完全なスクリプトについては、TransformExamples を参照してください）。

```
--Rotate a rectangle "myRectangle" around its center point.
set myRotateMatrix to make transformation matrix with properties {counterclockwise rotation
angle:27}
transform myRectangle in pasteboard coordinates from center anchor with matrix myRotateMatrix
--Scale a rectangle "myRectangle" around its center point.
set myScaleMatrix to make transformation matrix with properties {horizontal scale factor:.5,
vertical scale factor:.5}
transform myRectangle in pasteboard coordinates from center anchor with matrix myScaleMatrix
--Shear a rectangle "myRectangle" around its center point.
set myShearMatrix to make transformation matrix with properties {clockwise shear angle: 30}
transform myRectangle in pasteboard coordinates from center anchor with matrix myShearMatrix
--Rotate a rectangle "myRectangle" around a specified ruler point ({72, 72}).
set myRotateMatrix to make transformation matrix with properties {counterclockwise rotation
angle:27}
transform myRectangle in pasteboard coordinates from {{72, 72}, center anchor} with matrix
myRotateMatrix with considering ruler units
--Scale a rectangle "myRectangle" around a specified ruler point ({72, 72}).
set myScaleMatrix to make transformation matrix with properties {horizontal scale factor:.5,
vertical scale factor:.5}
transform myRectangle in pasteboard coordinates from {{72, 72}, center anchor} with matrix
myScaleMatrix with considering ruler units
```

変形関数を、簡単に使用できる一連のハンドラーに「ラップ」するスクリプトについては、Transform スクリプトを参照してください。

変形マトリックスの操作

一度作成した変形マトリックスは変更できませんが、様々な方法で変形マトリックスを操作して、既存の変形マトリックスに基づいて新規の変換マトリックスを作成できます。次のスクリプトに、変形マトリックスに複数の変形を適用し、元のマトリックスを置き換える方法の例を示します （完全なスクリプトについては、TransformMatrix を参照してください）。

```
--Scale a transformation matrix by 50% in both
--horizontal and vertical dimensions.
set myTransformationMatrix to scale matrix myTransformationMatrix
horizontally by .5 vertically by .5
--Rotate a transformation matrix by 45 degrees.
set myTransformationMatrix to rotate matrix by angle 45
--Shear a transformation matrix by 15 degrees.
set myTransformationMatrix to shear matrix by angle 15
```

`rotate matrix` メソッドを使用する場合は、次の RotateMatrix スクリプトのように、角度ではなくサイン値またはコサイン値を使用してマトリックスを変形できます。

```
set myRectangle to rectangle 1 of page 1 of document 1
set myTransformationMatrix to make transformation matrix
--rotate matrix can take the following parameters: byAngle, byCosine, bySine;
--The following statements are equivalent (0.25881904510252 is the sine of 15 degrees,
0.96592582628907 is the cosine).
set myTransformationMatrix to rotate matrix myTransformationMatrix by angle 15
set myTransformationMatrix to rotate matrix myTransformationMatrix by cosine 0.965925826289
set myTransformationMatrix to rotate matrix myTransformationMatrix by sine 0.258819045103
--Rotate the rectangle by the rotated matrix--45 degrees.
transform myRectangle in pasteboard coordinates from center anchor with matrix
myTransformationMatrix
```

`shear matrix` メソッドを使用する場合は、次の ShearMatrix スクリプトのように、角度ではなく傾斜を使用できます。

```
set myRectangle to rectangle 1 of page 1 of document 1
set myTransformationMatrix to make transformation matrix
--shear matrix can take the following parameters: by angle, by slope
--The following statements are equivalent. slope = rise/run
--so a 45 degree slope is 1.
set myTransformationMatrix to shear matrix myTransformationMatrix by slope 1
--Shear the rectangle.
transform myRectangle in pasteboard coordinates from center anchor with matrix
myTransformationMatrix
```

`invert matrix` メソッドを使用すると、変形マトリックスの逆マトリックスを取得できます。次のスクリプトにその例を示します （完全なスクリプトについては、InvertMatrix を参照してください）。マトリックスの効果は、反転させた変形マトリックスを使用して元に戻すことができます。

```
set myRectangle to rectangle 1 of page 1 of document 1
set myTransformationMatrix to make transformation matrix with properties {counterclockwise
rotation angle:30, horizontal translation:12, vertical translation:12}
transform myRectangle in pasteboard coordinates from center anchor with matrix
myTransformationMatrix
set myNewRectangle to duplicate myRectangle
--Move the duplicated rectangle to the location of the original
--rectangle by inverting, then applying the transformation matrix.
set myTransformationMatrix to invert matrix myTransformationMatrix
transform myNewRectangle in pasteboard coordinates from center anchor with matrix
myTransformationMatrix
```

`catenate matrix` メソッドを使用すると、変形マトリックスを追加できます。次のスクリプトにその例を示します （完全なスクリプトについては、CatenateMatrix を参照してください）。

```
set myTransformationMatrixA to make transformation matrix with properties {counterclockwise
rotation angle:30}
set myTransformationMatrixB to make transformation matrix with properties {horizontal
translation:72, vertical translation:72}
set myRectangle to rectangle -1 of page 1 of document 1
set myNewRectangle to duplicate myRectangle
--Rotate the duplicated rectangle.
transform myNewRectangle in pasteboard coordinates from center anchor with matrix
myTransformationMatrixA
set myNewRectangle to duplicate myRectangle
--Move the duplicate (unrotated) rectangle.
transform myNewRectangle in pasteboard coordinates from center anchor with matrix
myTransformationMatrixB
--Merge the two transformation matrices.
set myTransformationMatrix to catenate matrix myTransformationMatrixA with matrix
myTransformationMatrixB
set myNewRectangle to duplicate myRectangle
--The duplicated rectangle will be both moved and rotated.
transform myNewRectangle in pasteboard coordinates from center anchor with matrix
myTransformationMatrix
```

オブジェクトが変形されている場合は、`transform values of` メソッドを使用して、そのオブジェクトに適用された変形マトリックスを取得できます。次のスクリプトにその例を示します （完全なスクリプトについては、TransformValuesOf を参照してください）。

```
set myRectangle to rectangle -1 of page 1 of document 1
--Note that transform values of always returns a list containing a
--single transformation matrix.
set myTransformArray to transform values of myRectangle in pasteboard coordinates
set myTransformationMatrix to item 1 of myTransformArray
set myRotationAngle to counterclockwise rotation angle of myTransformationMatrix
set myShearAngle to clockwise shear angle of myTransformationMatrix
set myXScale to horizontal scale factor of myTransformationMatrix
set myYScale to vertical scale factor of myTransformationMatrix
set myXTranslate to horizontal translation of myTransformationMatrix
set myYTranslate to vertical translation of myTransformationMatrix
set myString to "Rotation Angle: " & myRotationAngle & return
set myString to myString & "Shear Angle: " & myShearAngle & return
set myString to myString & "Horizontal Scale Factor: " & myXScale & return
set myString to myString & "Vertical Scale Factor: " & myYScale & return
set myString to myString & "Horizontal Translation: " & myXTranslate & return
set myString to myString & "Vertical Translation: " & myYTranslate & return & return
set myString to myString & "Note that the Horizontal Translation and" & return
set myString to myString & "Vertical Translation values are the location" & return
set myString to myString & "of the center anchor in pasteboard coordinates."
display dialog (myString)
```

注意 : このメソッドで返された変形マトリックスの水平変形フィールド値と垂直変形フィールド値は、ペーストボードの座標におけるオブジェクトの左上アンカーの位置です。

座標スペース

前述の変形スクリプトでは、transform メソッドのパラメーターに `pasteboard coordinates` 列挙法が指定されています。このパラメーターによって、変形操作が発生する座標系、つまり座標スペースが決まります。座標スペースの値は、次のいずれかです。

- **`pasteboard coordinates`** は、InDesign ドキュメント全体の座標スペースです。この座標スペースは、ドキュメント内のすべてのスプレッドにわたって適用されます。InDesign の定規または原点には対応しておらず、InDesign のユーザーインターフェイスでページの周囲に表示されるペーストボードの領域とも関係ありません。オブジェクトに適用した変形による、この座標スペースへの影響はありません（例えば、水平軸と垂直軸の角度は変化しません）。

- **`parent coordinates`** は、オブジェクトの親の座標スペースです。親に適用した変形は、親座標に影響を与えます。例えば、親オブジェクトを回転させると、この座標スペースの水平軸と垂直軸の角度が変化します。この場合、親オブジェクトとはオブジェクトを格納しているグループまたはページアイテムを指します。オブジェクトの親がページまたはスプレッドの場合、親座標はスプレッドの座標と同じです。

- **`inner coordinates`** は、オブジェクト自体が作成された座標スペースです。

- **`spread coordinates`** は、スプレッドの座標スペースです。この座標スペースの基点は、スプレッドの中心にあり、ユーザーインターフェイスに表示される定規には対応していません。

次のスクリプトは、座標スペース間の相違を示します （完全なスクリプトについては、CoordinateSpaces を参照してください）。

```
set myRectangle to rectangle 1 of group 1 of page 1 of document 1
set myString to "The page contains a group which has been" & return
set myString to myString & "rotated 45 degrees (counterclockwise)." & return
set myString to myString & "The rectangle inside the group was" & return
set myString to myString & "rotated 45 degrees counterclockwise before" & return
set myString to myString & "it was added to the group." & return & return
set myString to myString & "Watch as we apply a series of scaling" & return
set myString to myString & "operations in different coordinate spaces."
display dialog myString
set myTransformationMatrix to make transformation matrix with properties {horizontal scale
factor:2}
--Transform the rectangle using inner coordinates.
transform myRectangle in inner coordinates from center anchor with matrix myTransformationMatrix
--Select the rectangle and display an alert.
select myRectangle
display dialog "Transformed using inner coordinates."
--Undo the transformation.
tell document 1 to undo
--Transform using parent coordinates.
transform myRectangle in parent coordinates from center anchor with matrix myTransformationMatrix
select myRectangle
display dialog "Transformed using parent coordinates."
tell document 1 to undo
--Transform using pasteboard coordinates.
transform myRectangle in pasteboard coordinates from center anchor with matrix
myTransformationMatrix
select myRectangle
display dialog "Transformed using pasteboard coordinates."
tell document 1 to undo
```

変形の基点

変形の基点は、変形の中心点です。変形の基点は様々な方法で指定できます。

- 境界線スペース：
  - アンカー - オブジェクト自体のアンカーポイント。

    ```
    center anchor
    ```

- 定規スペース：
  - (x, y)、ページインデックス - スプレッドの指定ページ上にある定規基点との相対位置で指定されたポイント。

    ```
    {{72, 144}, 1}
    ```

  - (x, y)、位置 - オブジェクトの指定された位置の親ページとの相対位置で指定されたポイント。位置は、アンカーポイントまたは 1 組の座標として指定できます。位置は、オブジェクトの境界線または表示可能な境界線との相対位置で指定でき、特定の座標スペースに指定できます。

    ```
    {{72, 144}, center anchor}
    ```

- 変形スペース：
  - (x, y) - ペーストボードの座標スペース内のポイント。

    ```
    {72, 72}
    ```

  - (x, y)、座標系 - 指定した座標スペース内のポイント。

    ```
    {{72, 72}, parent coordinates}
    ```

  - ((x, y)) - `transform` メソッドの `in` パラメーターとして指定された座標スペース内のポイント。

```
{{72, 72}}
```

変形の基点オプションを使用する方法について、次のスクリプトにいくつかの例を示します（完全なスクリプトについては、TransformationOrigin を参照してください）。

```
set myRectangle to rectangle 1 of document 1
set myString to "Watch as we rotate the rectangle using different anchor points," & return
set myString to myString & "bounds types, and coordinate spaces." & return & return
set myString to myString & "You might have to drag the alert aside to" & return
set myString to myString & "see the effect of the transformation."
set myNewRectangle to duplicate myRectangle
set myTransformationMatrix to make transformation matrix with properties {counterclockwise
rotation angle:30}
--Rotate around the duplicated rectangle's center point.
transform myNewRectangle in pasteboard coordinates from center anchor with matrix
myTransformationMatrix
--Select the rectangle and display an alert.
select myNewRectangle
display dialog "Rotated around center anchor."
--Undo the transformation.
tell document 1 to undo
--Rotate the rectangle around the ruler location [-100, -100]. Note that the anchor point
specified here specifes the page
--containing the point--*not* that transformation point itself. The transformation gets the ruler
coordinate [-100, -100] based
--on that page. Setting the considerRulerUnits parameter to true makes certain that the
transformation uses the current
--ruler units.
transform myNewRectangle in pasteboard coordinates from {{-100, -100}, top left anchor} with
matrix myTransformationMatrix with considering ruler units
--Move the page guides to reflect the transformation point.
tell guide 1 of page 1 of document 1 to set location to -100
tell guide 2 of page 1 of document 1 to set location to -100
--Select the rectangle and display an alert.
select myNewRectangle
display dialog "Rotated around -100x, -100y."
--Undo the transformation and the guide moves.
tell document 1 to undo
tell document 1 to undo
tell document 1 to undo
```

位置の解決

ある座標スペースに指定したポイントの位置を、別の座標スペースのコンテキストで取得することが必要になる場合があります。取得するには、`resolve` メソッドを使用します。次のスクリプトにその例を示します（完全なスクリプトについては、ResolveLocation を参照してください）。

```
set myRectangle to rectangle 1 of group 1 of page 1 of document 1
--Get ruler coordinate {72, 72} in pasteboard coordinates.
set myPageLocation to resolve myRectangle location {{72, 72}, top right anchor} in pasteboard
coordinates with considering ruler units
--resolve returns a list containing a single item.
set myPageLocation to item 1 of myPageLocation
display dialog "Pasteboard Coordinates:" & return & return & "X: " & item 1 of myPageLocation &
return & "Y: " & item 2 of myPageLocation
--Get ruler coordinate {72, 72} in parent coordinates.
set myPageLocation to resolve myRectangle location {{72, 72}, top right anchor} in parent
coordinates with considering ruler units
--resolve returns a list containing a single item.
set myPageLocation to item 1 of myPageLocation
display dialog "Parent Coordinates:" & return & return & "X: " & item 1 of myPageLocation & return
& "Y: " & item 2 of myPageLocation
```

ポイントの変形

ポイントもオブジェクトと同様に変形することができます。つまり、スクリプト自体に計算を組み込まなくてもスクリプトで様々な算術演算を処理できます。これは AppleScript の場合に特に役立ちます。AppleScript には、ほとんどの変形に必要な基本的な三角関数（サイン、コサイン）が欠如しています。次の ChangeCoordinates サンプルスクリプトは、このアプローチを使用して一連の正多角形を描画する方法を示しています。

```
--General purpose routine for drawing regular polygons from their center point.
on myDrawPolygon(myParent, myCenterPoint, myNumberOfPoints, myRadius, myStarPolygon, myStarInset)
    local myPathPoints, myTransformedPoint
    tell application "Adobe InDesign CS6"
        set myPathPoints to {}
        set myPoint to {0, 0}
        if myStarPolygon is true then
            set myNumberOfPoints to myNumberOfPoints * 2
        end if
        set myInnerRadius to myRadius * myStarInset
        set myAngle to 360 / myNumberOfPoints
        set myRotateMatrix to make transformation matrix with properties
        {counterclockwise rotation angle:myAngle}
        set myOuterTranslateMatrix to make transformation matrix with properties
        {horizontal translation:myRadius}
        set myInnerTranslateMatrix to make transformation matrix with properties
        {horizontal translation:myInnerRadius}
        repeat with myPointCounter from 0 to myNumberOfPoints - 1
            --Translate the point to the inner/outer radius.
            if myStarInset = 1 or my myIsEven(myPointCounter) is true then
                set myTransformedPoint to change coordinates
                myOuterTranslateMatrix point myPoint
            else
                set myTransformedPoint to change coordinates
                myInnerTranslateMatrix point myPoint
            end if
            --Rotate the point.
            set myTransformedPoint to change coordinates
            myRotateMatrix point myTransformedPoint
            copy myTransformedPoint to the end of myPathPoints
            set myRotateMatrix to rotate matrix myRotateMatrix by angle myAngle
        end repeat
        --Create a new polygon.
        tell myParent
            set myPolygon to make polygon
        end tell
        --Set the entire path of the polygon to the array we've created.
        set entire path of path 1 of myPolygon to myPathPoints
        --If the center point is somewhere other than [0,0],
        --translate the polygon to the center point.
        if item 1 of myCenterPoint is not equal to 0 or item 2 of
    myCenterPoint is not equal to 0 then
            set myTransformationMatrix to make transformation matrix with properties
            {horizontal translation:item 1 of myCenterPoint,
            vertical translation:item 2 of myCenterPoint}
            transform myPolygon in pasteboard coordinates from {myCenterPoint,
```

```
            center anchor} with matrix myTransformationMatrix with considering
        ruler units
        end if
    end tell
end myDrawPolygon
--This function returns true if myNumber is even, false if it is not.
on myIsEven(myNumber)
    set myResult to myNumber mod 2
    if myResult = 0 then
        set myResult to true
    else
        set myResult to false
    end if
    return myResult
end myIsEven
```

FunWithTransformations サンプルスクリプトに示すように、`change coordinates` メソッドを使用して、カーブを制御するポイントの位置を変更することもできます。

変形の繰り返し

ユーザーインターフェイスでは変形または一連の変形を繰り返し適用できますが、同じことがスクリプティングでも可能です。変形を繰り返し適用するメソッドは、4 種類あります。

- `transform again`
- `transform again individually`
- `transform sequence again`
- `transform sequence again individually`

`transform again` の使用方法を、次のスクリプトに示します （完全なスクリプトについては、TransformAgain を参照してください）。

```
set myTransformationMatrix to make transformation matrix with properties {counterclockwise
rotation angle:45}
transform myRectangleA in pasteboard coordinates from center anchor with matrix
myTransformationMatrix
set myRectangleB to duplicate myRectangleA
transform myRectangleB in pasteboard coordinates from {{0, 0}, top left anchor} with matrix
myTransformationMatrix with considering ruler units
set myRectangleC to duplicate myRectangleB
set myResult to transform again myRectangleC
set myRectangleD to duplicate myRectangleC
set myResult to transform again myRectangleD
set myRectangleE to duplicate myRectangleD
set myResult to transform again myRectangleE
set myRectangleF to duplicate myRectangleE
set myResult to transform again myRectangleF
set myRectangleG to duplicate myRectangleF
set myResult to transform again myRectangleG
set myRectangleH to duplicate myRectangleG
set myResult to transform again myRectangleH
transform myRectangleB in pasteboard coordinates from center anchor with matrix
myTransformationMatrix
set myResult to transform again myRectangleD
set myResult to transform again myRectangleF
set myResult to transform again myRectangleH
```

サイズ変更とフレーム調整

transform コマンドを使用したページアイテムの拡大・縮小に加え、他の 2 つのコマンド resize と reframe を使用してシェイプのサイズを変更することもできます。これらのコマンドでは、ページアイテムのコンテンツや線幅を拡大・縮小せずに、ページアイテムのパスポイントの位置が変更されます。resize コマンドの使用方法を、次のスクリプトに示します。完全なスクリプトについては、Resize を参照してください。

```
--Given a reference to a rectangle "myRectangle"...
set myDuplicate to duplicate myRectangle
resize myDuplicate in inner coordinates from center anchor by multiplying current dimensions by
values{2, 2}
```

reframe コマンドの使用方法を、次のスクリプトに示します。完全なスクリプトについては、Reframe を参照してください。

```
--Given a reference to a rectangle "myRectangle"...
set myBounds to geometric bounds of myRectangle
set myX1 to item 2 of myBounds - 72
set myY1 to item 1 of myBounds - 72
set myX2 to item 4 of myBounds + 72
set myY2 to item 3 of myBounds + 72
set myDuplicate to duplicate myRectangle
myDuplicate = myRectangle.duplicate();
reframe myDuplicate in inner coordinates opposing corners {{myY1, myX1},{myY2, myX2}}
```

アーティクルの使用

アーティクルを使用することで、デザイナーや制作アーティストはページアイテム間の関係を簡単に作成することができます。関係を使用することで、ePub、HTML またはアクセシブル PDF への書き出しに使用するコンテンツや、コンテンツの順序を定義できます。画像、グラフィック、テキストなど、レイアウトに含まれている既存のページアイテムを組み合わせてアーティクルを作成することができます。アーティクルを作成した後は、ページアイテムを追加または削除したり、ページアイテムの順序を変更したりできます。

アーティクルリストの取得

ドキュメント内のアーティクルを取得するには、次のスクリプトを使用します（完全なスクリプトについては、CreateArticle を参照してください）。

```
set myDocument to document 1
set myArticles to articles of myDocument
```

新しいアーティクルの追加

ドキュメントに新しいアーティクルを追加するには、次のスクリプトを使用します（完全なスクリプトについては、CreateArticle を参照してください）。

```
set newArticle1 to make article with properties {name:"Article1", article export status:true}
```

アーティクルの削除

アーティクルを削除するには、次のスクリプトに示すように、削除対象のアーティクル自体に対して削除を指示します（完全なスクリプトについては、RemoveArticle を参照してください）。

```
tell article 1 of myDocument to delete
```

アーティクルの順序変更

アーティクルの順序を変更するには、次のスクリプトに示すように、移動対象のアーティクルに対して参照アーティクルと参照アーティクルを基準にした相対位置を指示します（完全なスクリプトについては、ReorderArticles を参照してください）。

```
tell myDocument
    set article1 to make article with properties {name:"Article1"}
    set article2 to make article with properties {name:"Article2"}
    set article3 to make article with properties {name:"Article3"}
    set article4 to make article with properties {name:"Article4"}

    move article4 to beginning of article 0
    move article1 to end of article 0
    move article3 to after article4
    move article2 to before article1
end tell
```

アーティクルへの新規メンバーの追加

アーティクルに新規メンバーを追加するには、次のスクリプトを使用します（完全なスクリプトについては、AddRemoveArticleMember を参照してください）。

```
set myDocument to document 1
tell myDocument
    set article1 to make article with properties {name:"Article1"}
    set article2 to make article with properties {name:"Article2"}
    set article3 to make article with properties {name:"Article3"}

    set myPage to page 1 of myDocument

    --add rectangle 1 of myPage to article1
    tell article1
        make article member with properties {item ref:rectangle 1 of myPage}
        make article member with properties {item ref:oval 1 of myPage}
    end tell

    tell article2
        make article member with properties {item ref:rectangle 2 of myPage}
        make article member with properties {item ref:oval 2 of myPage}
    end tell

    tell article3
        make article member with properties {item ref:group 1 of myPage}
    end tell
end tell
```

アーティクルのメンバーの順序変更

アーティクルのメンバーの順序を変更するには、次のスクリプトに示すように、移動対象のメンバーに対して参照オブジェクトと参照オブジェクトを基準にした相対位置を指示します（完全なスクリプトについては、ReorderArticleMembers を参照してください）。

```
--Assume there are four members in the article in the following order:
--rectangle, oval, text frame, group
make article member with properties {item ref:rectangle 1 of myPage}
make article member with properties {item ref:oval 1 of myPage}
make article member with properties {item ref:text frame 1 of myPage}
make article member with properties {item ref:group 1 of myPage}

move article member 1 to end of article member 0
move article member 3 to beginning of article member 0
move article member 2 to after article member 3
```

アーティクルからのメンバーの削除

アーティクルからメンバーを削除するには、次のスクリプトに示すように、削除対象のアーティクルメンバー自体に対して削除を指示します（完全なスクリプトについては、AddRemoveArticleMember を参照してください）。

```
tell article member 1 of article1 to delete
```

6 テキストと組版

章の更新ステータス		
CS6	更新	101 ページの「テキストフレームをコンテンツに合わせる」という節の追加。

テキストの入力、編集、フォーマットといった作業は、InDesign ドキュメントの製作工程の中でも、最も時間のかかる作業の 1 つです。したがって、テキストや書式に関する作業を自動化できれば、生産性の大幅な向上が期待できます。

この章では、テキストや書式に関する一般的な操作をスクリプトで実行する方法について説明します。最初は簡単なサンプルスクリプトから始めて、徐々に複雑なスクリプトへと進んでいきます。

ここでは、読者が『Adobe InDesign スクリプティングチュートリアル』を既に読んでおり、スクリプトを作成、インストール、実行する方法を理解しているものとして説明を行います。また、InDesign でテキストを操作する方法や、組版に関する基本的な用語を理解しているものとしています。

テキストの入力と読み込み

ここでは、InDesign ドキュメントにテキストを配置する方法について説明します。InDesign のユーザーインターフェイスを使用してテキストフレームにテキストを入力したり、テキストファイルを配置したりするのと同じように、スクリプトを使用してテキストフレームを作成し、ストーリーにテキストを挿入し、ページ上にテキストファイルを配置することができます。

テキストフレームの作成

次のスクリプトでは、テキストフレームを作成し、フレームの境界（サイズ）を設定し、フレームにテキストを入力しています（完全なスクリプトについては、MakeTextFrame というチュートリアルスクリプトを参照してください）。

```
--Given a document "myDocument"...
set myDocument to document 1
set myPage to page 1 of myDocument
tell myPage
    set myTextFrame to make text frame
    --Set the bounds of the text frame.
    set geometric bounds of myTextFrame to {72, 72, 288, 288}
    --Enter text in the text frame.
    set contents of myTextFrame to "This is some example text."
    --Note that you could also use a properties record
    --to set the bounds and contents of the text in a
    --single line:
    --set myTextFrame to make text frame with properties{geometric bounds:{72, 72, 288, 288},
contents:"This is some example text."}
end tell
```

次のスクリプトでは、ページマージンによって定義される領域と同じサイズのテキストフレームを作成しています。ここで使用している `myGetBounds` は、様々なスクリプトで活用できる便利な関数であり、この章の多くの例でも使用されています （完全なスクリプトについては、MakeTextFrameWithinMargins を参照してください）。

```
--Given a document "myDocument"
set myPage to page 1 of myDocument
tell myPage
    set myTextFrame to make text frame
    --Set the bounds of the text frame.
    set geometric bounds of myTextFrame to my myGetBounds(myDocument, myPage)
end tell
```

次のスクリプトに myGetBounds ハンドラーを示します。

```
on myGetBounds(myDocument, myPage)
    tell application "Adobe InDesign CS6"
        tell document preferences of myDocument
            set myPageWidth to page width
            set myPageHeight to page height
        end tell
        tell margin preferences of myPage
            if side of myPage is left hand then
                set myX2 to left
                set myX1 to right
            else
                set myX1 to left
                set myX2 to right
            end if
            set myY1 to top
            set myY2 to bottom
        end tell
        set myX2 to myPageWidth - myX2
        set myY2 to myPageHeight - myY2
        return {myY1, myX1, myY2, myX2}
    end tell
end myGetBounds
```

テキストの追加

ストーリーにテキストを追加するには、テキストを挿入する箇所で挿入点の `contents` プロパティを使用します。次のサンプルスクリプトでは、この方法を使用して、ストーリーの末尾にテキストを追加しています（完全なスクリプトについては、AddText を参照してください）。

```
--Given a document "myDocument" with a text frame on page 1
set myTextFrame to text frame 1 of page 1 of myDocument
--Add text to the end of the text frame. To do this,
--We'll use the last insertion point in the story.
set myNewText to "This is a new paragraph of example text."
tell parent story of myTextFrame
    set contents of insertion point -1 to return & myNewText
end tell
```

ストーリーとテキストフレーム

InDesign レイアウトの中にあるテキストは、すべてストーリーの中に含まれています。どのストーリーにも 1 つ以上のテキストフレームを含めることができます。テキストフレームを 1 つ作成するとストーリーが 1 つ作成されます。ストーリーには複数のテキストフレームを含めることができます。

前述のスクリプトでは、テキストフレームの末尾ではなく、親ストーリーの末尾にテキストを追加しました。これは、テキストの長さやフォーマットによっては、テキストフレームの末尾がストーリーの末尾ではないことがあるからです。親ストーリーの末尾にテキストを追加することによって、テキストフレームの状態に関係なく、確実にテキストが追加されるようにしています。

テキストフレームの親ストーリーを取得するには、テキストフレームの `parent text frame` プロパティを使用します。多くの場合、テキストフレームのテキストを操作するよりも、ストーリーのテキストを操作するほうが便利です。この違いを次のスクリプトに示します。表示される警告を見ると、テキストフレームには

オーバーセットテキストが含まれておらず、ストーリーには含まれていることがわかります（完全なスクリプトについては、StoryAndTextFrame を参照してください）。

```
--Given a document "myDocument" with a text frame on page 1...
set myTextFrame to text frame 1 of myDocument
--Now add text beyond the end of the text frame.
set myString to return & "This is some overset text"
tell insertion point -1 of myTextFrame to set contents to myString
set myString to contents of text 1 of myTextFrame
display dialog ("The last paragraph in this dialog should be ¥"This is some overset text¥". Is it?"
& return & myString)
set myString to contents of parent story of myTextFrame
display dialog ("The last paragraph in this alert should be ¥"This is some overset text¥". Is it?"
& return & myString)
```

InDesign ドキュメントの中にあるテキストオブジェクトの関係について詳しくは、93 ページの「テキストオブジェクト」を参照してください。

テキストの置換

次のスクリプトでは、適当なオブジェクトの内容を変更することで、1 つの単語をある語句に置換しています（完全なスクリプトについては、ReplaceWord を参照してください）。

```
--Given a document "myDocument" with a text frame on page 1...
set myTextFrame to text frame 1 of page 1 of myDocument
--Replace the third word with the phrase "a little bit of".
tell word 3 of parent story of myTextFrame
    set contents to "a little bit of"
end tell
```

次のスクリプトでは、段落内のテキストを置換しています（完全なスクリプトについては、ReplaceText を参照してください）。

```
--Given a document "myDocument" with a text frame on page 1...
set myTextFrame to text frame 1 of page 1 of myDocument
--Replace the text in the second paragraph without
--replacing the return character at the end of the paragraph
--(character -2 is the character before the return).
tell parent story of myTextFrame
    set myText to object reference of text from character 1 to character -2 of paragraph 2
    set contents of myText to "This text replaces the text in paragraph 2."
end tell
    end tell
```

このスクリプトでは、改行文字を除外しています。改行を削除してしまうと、段落に適用されている段落スタイルが変更される可能性があるからです。したがって、文字範囲を指定して、段落の先頭の文字から、段落の末尾の文字（改行文字）の直前にある文字までを置換しています。

特殊文字の挿入

スクリプトエディタでは Unicode がサポートされているので、InDesign に渡すテキスト文字列に Unicode 文字を簡単に入力できます。また、Unicode 文字の字形の識別番号を指定する <nnnn> という InDesign ショートカット（*nnnn* は、目的の文字の Unicode コード）を使用することもできます。次のスクリプトでは、特殊文字をいくつかの方法で入力しています（ここでは、`myGetBounds` 関数は省略しています。この関数については、83 ページの「テキストフレームの作成」または SpecialCharacters というチュートリアルスクリプトを参照してください）。

```
--Given a document "myDocument" containing a story...
set myStory to story 1 of myDocument
--Entering special characters directly:
set contents of myStory to "Registered trademark: ®" & return & "Copyright: ©" & return &
"Trademark: ™" & return
--Entering special characters by their Unicode glyph ID value:
set contents of insertion point -1 of myStory to "Not equal to: <2260>" & return
set contents of insertion point -1 of myStory to "Square root: <221A>" & return
set contents of insertion point -1 of myStory to "Square root: <00B6>" & return
--Entering special characters by their enumerations:
set contents of insertion point -1 of myStory to "Automatic page number marker: "
set contents of insertion point -1 of myStory to auto page number
set contents of insertion point -1 of myStory to return & "Section symbol: "
set contents of insertion point -1 of myStory to section symbol
set contents of insertion point -1 of myStory to return & "En dash: "
set contents of insertion point -1 of myStory to En dash
set contents of insertion point -1 of myStory to return
```

特定の文字の Unicode ID を調べたい場合は、InDesign の字形パレットを使用するのが最も簡単です。パレット内で文字の上にカーソルを移動すると、その文字の Unicode 値が表示されます。Unicode について詳しくは、http://www.unicode.org を参照してください。

テキストの配置とテキスト読み込み環境設定の指定

テキスト文字列の入力だけでなく、ワードプロセッサーやテキストエディターで作成したテキストファイルを配置することもできます。ドキュメントページにテキストファイルを配置する方法を、次のスクリプトに示します（完全なスクリプトについては、PlaceTextFile を参照してください）。

```
--Given a document "myDocument"
set myPage to page 1 of myDocument
--Get the top and left margins to use as a place point.
tell margin preferences of myPage
    set myX to left
    set myY to right
end tell
--Autoflow a text file on the current page.
--Parameters for the place command of a page:
--file as file or string
--[place point as list {x, y}
--[destination layer as layer object or string]
--[showing options as Boolean (default is false)]
--[autoflowing as Boolean (default is false)]
--You'll have to fill in a valid  file path on your system.
tell myPage
    --Note that if the PlacePoint parameter is inside a column, only the vertical (y)
    --coordinate will be honored--the text frame will expand horizontally to fit the column.
    set myStory to place alias "Macintosh HD:scripting:test.txt" place point {myX, myY}
autoflowing yes without showing options
end tell
```

既存のテキストフレームにテキストファイルを配置する方法を、次のスクリプトに示します （ここでは、myGetBounds 関数は省略しています。この関数については、83 ページの「テキストフレームの作成」または PlaceTextFileInFrame というチュートリアルスクリプトを参照してください）。

```
--Given a document "myDocument" with a text frame on page 1...
set myTextFrame to text frame 1 of page 1 of myDocument
tell insertion point -1 of myTextFrame
    --You'll need to fill in a valid file path for your system.
    place "yukino:test.txt" without showing options
end tell
```

テキスト内の特定の場所にテキストファイルを挿入する方法を、次のスクリプトに示します（ここでは、myGetBounds 関数は省略しています。この関数については、83 ページの「テキストフレームの作成」または InsertTextFile というチュートリアルスクリプトを参照してください）。

```
--Given a document "myDocument" with a text frame on page 1...
--Place a text file at the end of the text.
set myTextFrame to text frame 1 of page 1 of myDocument
tell insertion point -1 of parent story of myTextFrame
    --You'll need to fill in a valid file path for your system.
    place "yukino:test.txt" without showing options
end tell
```

配置するテキストファイルの種類に応じた読み込みオプションを指定するには、対応する読み込み環境設定オブジェクトを使用します。テキスト読み込み環境設定の指定方法を、次のスクリプトに示します（完全なスクリプトについては、TextImportPreferences を参照してください）。スクリプトのコメントに、各プロパティに指定可能な値を示します。

```
tell text import preferences
    --Options for character set:
    set character set to UTF8
    set convert spaces into tabs to true
    set spaces into tabs count to 3
    --The dictionary property can take many values, such as French, Italian.
    set dictionary to "English: USA"
    --platform options:
    --macintosh
    --pc
    set platform to macintosh
    set strip returns between lines to true
    set strip returns between paragraphs to true
    set use typographers quotes to true
end tell
```

タグ付きテキスト読み込み環境設定の指定方法を、次のスクリプトに示します（完全なスクリプトについては、TaggedTextImportPreferences を参照してください）。

```
tell tagged text import preferences
    set remove text formatting to false
    --Options for style conflict are:
    --publication definition
    --tag file definition
    set style conflict to publication definition
    set use typographers quotes to true
end tell
```

Word および RTF 読み込み環境設定の指定方法を、次のスクリプトに示します（完全なスクリプトについては、WordRTFImportPreferences を参照してください）。

```
tell word RTF import preferences
    --convert page breaks property can be:
    --column break
    --none
    --page break
    set convert page breaks to none
    --convert tables to property can be:
    --unformatted tabbed text
    --unformatted table
    set convert tables to to Unformatted Table
    set import endnotes to true
    set import footnotes to true
    set import index to true
    set import TOC to true
    set import unused styles to false
    set preserve graphics to false
    set preserve local overrides to false
    set preserve track changes to false
    set remove formatting to false
    --resolve character style clash and resolve paragraph style clash properties can be:
    --resolve clash auto rename
    --resolve clash use existing
    --resolve clash use new
    set resolve character style clash to resolve clash use existing
    set resolve paragraph style clash to resolve clash use existing
    set use typographers quotes to true
end tell
```

Excel 読み込み環境設定の指定方法を、次のスクリプトに示します（完全なスクリプトについては、ExcelImportPreferences を参照してください）。

```
tell application "Adobe InDesign CS6"
    tell excel import preferences
        --alignment Style property can be:
        --center align
        --left align
        --right align
        --spreadsheet
        set alignment style to spreadsheet
        set decimal places to 4
        set preserve graphics to false
        --Enter the range you want to import as "start cell:end cell".
        set range name to "A1:B16"
        set sheet index to 1
        set sheet name to "pathpoints"
        set show hidden cells to false
        --table formatting property can be:
        --excel formatted table
        --excel unformatted tabbed text
        --excel unformatted table
        set table formatting to excel formatted table
        set use typographers quotes to true
        set view name to ""
    end tell
end tell
```

テキストの書き出しとテキスト書き出し環境設定の指定

InDesign ドキュメントからテキストを書き出す方法を、次のスクリプトに示します。テキストファイル形式で書き出すには、テキストオブジェクトやストーリーオブジェクトを使用する必要があることに注意してください。1 つの操作でドキュメント内のすべてのテキストを書き出すことはできません（ここでは、myGetBounds 関数は省略しています。この関数については、83 ページの「テキストフレームの作成」または ExportTextFile というチュートリアルスクリプトを参照してください）。

```
tell application "Adobe InDesign CS6"
    set myDocument to active document
    set myPage to page 1 of myDocument
    tell myPage
        set myTextFrame to make text frame with properties {geometric bounds:
        my myGetBounds(myDocument, myPage), contents:placeholder text}
    end tell
    --Text export method parameters:
    --format as enumeration
    --to alias as string
    --showing options boolean
    --version comments string
    --force save boolean
    --Format parameter can be:
    --InCopy CS Document
    --InCopy Document
    --rtf
    --tagged text
    --text type
    --Export the story as text. You'll have to fill in a valid file path on your system.
    tell parent story of myTextFrame
        export to "yukino:test.txt" format text type
    end tell
end tell
```

指定したテキスト範囲を書き出す方法を、次の例に示します（ここでは、myGetBounds 関数は省略しています。この関数については、83 ページの「テキストフレームの作成」または ExportTextRange というチュートリアルスクリプトを参照してください）。

```
--Given a document with a text frame on page 1...
set myTextFrame to text frame 1 of page 1 of active document
set myStory to parent story of myTextFrame
set myStartCharacter to index of character 1 of paragraph 1 of myStory
set myEndCharacter to the index of last character of paragraph 1 of myStory
set myText to object reference of text from character myStartCharacter to character myEndCharacter
of myStory
--Export the text range. You'll have to fill in a valid file path on your system.
tell myText to export to "Macintosh HD:scripting:test.txt" format text type
```

書き出すテキストファイルの種類に応じた書き出しオプションを指定するには、対応する書き出し環境設定オブジェクトを使用します。テキスト書き出し環境設定を指定する方法を、次のスクリプトに示します（完全なスクリプトについては、TextExportPreferences を参照してください）。

```
tell text export preferences
    --Options for character set:
    --UTF8
    --UTF16
    --platform
    set character set to UTF8
    --platform options:
    --macintosh
    --pc
    set platform to macintosh
end tell
```

タグ付きテキスト書き出し環境設定を指定する方法を、次のスクリプトに示します（完全なスクリプトについては、TaggedTextExportPreferences を参照してください）。

```
tell tagged text export preferences
    --Options for character set:
    --ansi
    --ascii
    --gb18030
    --ksc5601
    --shiftJIS
    --unicode
    set character set to unicode
    --tag form options:
    --abbreviated
    --verbose
    set tag form to verbose
end tell
```

ドキュメント内のすべてのテキストを 1 ステップの操作で書き出すことはできません。すべてのテキストをまとめたい場合は、ドキュメント内のテキストを 1 つのストーリーに結合してからそのストーリーを書き出すか、または、生成された複数のファイルをスクリプトで読み込んで 1 つのファイルに書き出す必要があります。次のスクリプトでは、前者の方法を使用しています （ここでは、`myGetBounds` 関数は省略しています。この関数については、83 ページの「テキストフレームの作成」または ExportAllText というチュートリアルスクリプトを参照してください）。書き出す形式が単純なテキスト形式でない場合、後者の方法はかなり複雑になることがあります。

```
tell application "Adobe InDesign CS6"
    if (count documents) > 0 then
        set myDocument to active document
        if (count stories of myDocument) > 0 then
            my myExportAllText(name of myDocument)
        end if
    end if
end tell
```

このスクリプトで参照されている ExportAllText ハンドラーを次に示します。

```
on myExportAllText(myDocumentName)
    tell application "Adobe InDesign CS6"
        --File name for the exported text. Fill in a valid file path on your system.
        set myFileName to "Adobe:test.txt"
        --If you want to add a separator line between stories,
        --set myAddSeparator to true.
        set myAddSeparator to true
        set myNewDocument to make document
        set myDocument to document myDocumentName
        tell page 1 of myNewDocument
            set myTextFrame to make text frame with properties
            {geometric bounds:my myGetBounds(myNewDocument, page 1 of myNewDocument)}
            set myNewStory to parent story of myTextFrame
            repeat with myCounter from 1 to (count stories in myDocument)
                set myStory to story myCounter of myDocument
                --Duplicate the text of the story to the end of the temporary story.
                tell text 1 of myStory
                    duplicate to after insertion point -1 of story 1 of myNewDocument
                end tell
                --If the imported text did not end with a return, enter a return
                --to keep the stories from running together.
                if myCounter is not equal to (count stories of myDocument) then
                    if contents of character -1 of myNewStory is not return then
```

```
                    set contents of insertion point -1 of myNewStory to return
                    if myAddSeparator is true then
                        set contents of insertion point -1 of myNewStory to
                        "----------------------------------------" & return
                    end if
                end if
            end if
        end repeat
        tell myNewStory
            export to myFileName format text type
        end tell
        close myNewDocument saving no
    end tell
end tell
end myExportAllText
```

既存の書き出しフィルターを使用しないでテキストを書き出すこともできます。AppleScript には、テキストファイルをディスクに書き出す機能が用意されています。この機能を利用すれば、スクリプトでドキュメント内のテキストをスキャンして、任意のテキストマークアップスキームを使用して任意の順序でテキストを書き出すことができます。HTML 形式で InDesign テキストを書き出す簡単な例を次に示します（ここでは、`myGetBounds` 関数は省略しています。この関数については、83 ページの「テキストフレームの作成」または ExportHTML というチュートリアルスクリプトを参照してください）。

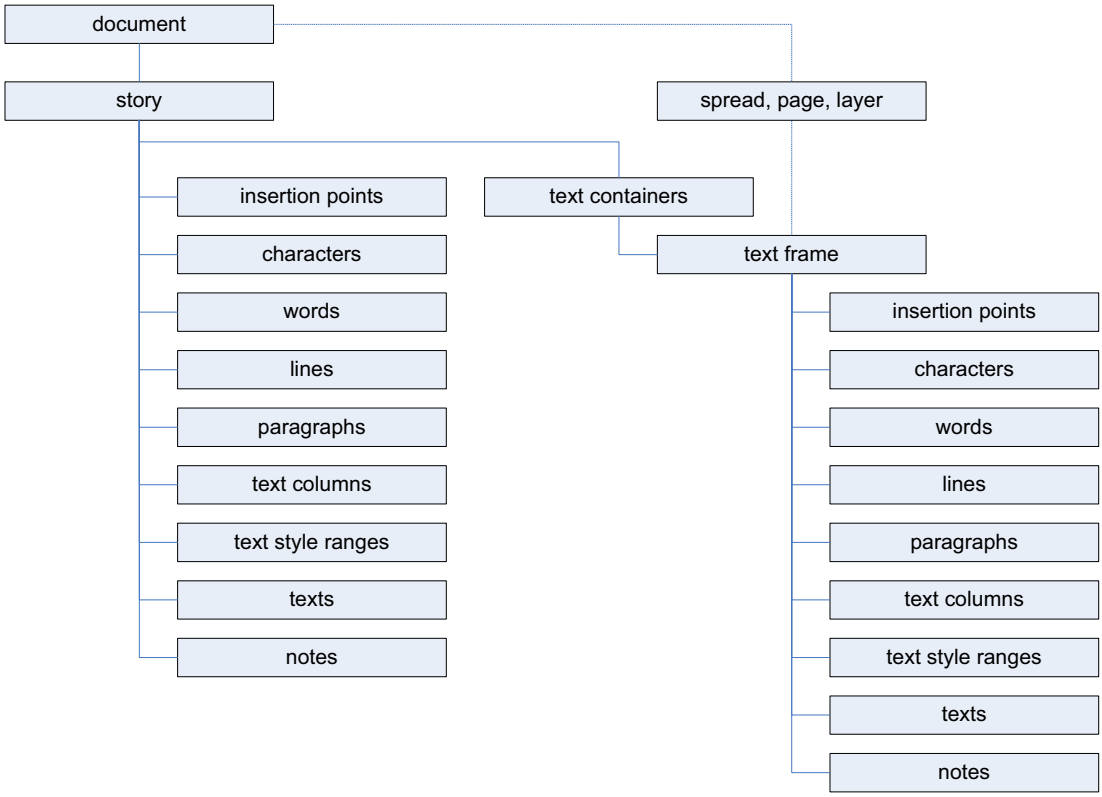
```
tell application "Adobe InDesign CS6"
    --Use the myStyleToTagMapping array to set up your paragraph style to tag mapping.
    set myStyleToTagMapping to {}
    --For each style to tag mapping, add a new item to the array.
    copy {"body_text", "p"} to end of myStyleToTagMapping
    copy {"heading1", "h1"} to end of myStyleToTagMapping
    copy {"heading2", "h3"} to end of myStyleToTagMapping
    copy {"heading3", "h3"} to end of myStyleToTagMapping
    --End of style to tag mapping.
    if (count documents) is not equal to 0 then
        set myDocument to document 1
        if (count stories of myDocument) is not equal to 0 then
            --Open a new text file.
            set myTextFile to choose file name with prompt "Save HTML As"
            --Iterate through the stories.
            repeat with myCounter from 1 to (count stories of myDocument)
                set myStory to story myCounter of myDocument
                repeat with myParagraphCounter from 1 to (count paragraphs of myStory)
                    set myParagraph to object reference of paragraph myParagraphCounter
                    of myStory
                    if (count tables of myParagraph) is 0 then
                        --If the paragraph is a simple paragraph--no tables,
                        --no local formatting--then simply export the text of the
                        --pararaph with the appropriate tag.
                        if (count text style ranges of myParagraph) is 1 then
                            set myTag to my myFindTag(name of applied paragraph style
                            of myParagraph, myStyleToTagMapping)
                            --If the tag comes back empty, map it to the
                            --basic paragraph tag.
                            if myTag = "" then
                                set myTag to "p"
                            end if
                            set myStartTag to "<" & myTag & ">"
                            set myEndTag to "</" & myTag & ">"
                            --If the paragraph is not the last paragraph in the story,
                            --omit the return character.
                            if the contents of character -1 of myParagraph is return then
                                set myText to object reference of text from character 1
                                to character -2 of myParagraph
                                set myString to contents of myText
                            else
                                set myString to contents of myParagraph
```

```
            end if
            --Write the text of the paragraph to the text file.
            if myParagraphCounter = 1 then
                set myAppendData to false
            else
                set myAppendData to true
            end if
            my myWriteToFile(myStartTag & myString & myEndTag & return,
            myTextFile, myAppendData)
        else
            --Handle text style range export by iterating through
            --the text style ranges in the paragraph.
            set myTextStyleRanges to text style ranges of myParagraph
            repeat with myRangeCounter from 1 to (count text style ranges
            of myParagraph)
                set myTextStyleRange to object reference of text style
                range myRangeCounter of myParagraph
                if character -1 of myTextStyleRange is return then
                    set myStartCharacter to index of character 1 of
                    myTextStyleRange
                    set myEndCharacter to index of character -2 of
                    myTextStyleRange
                    set myText to object reference of text from character
                    myStartCharacter to character myEndCharacter of myStory
                    set myString to contents of myText
                else
                    set myString to contents of myTextStyleRange
                end if
                if font style of myTextStyleRange is "Bold" then
                    set myString to "<b>" & myString & "</b>"
                else if font style of myTextStyleRange is "Italic" then
                    set myString to "<i>" & myString & "</i>"
                end if
                my myWriteToFile(myString, myTextFile, true)
            end repeat
            my myWriteToFile(return, myTextFile, true)
        end if
    else
        --Handle table export (assumes that there is only
        --one table per paragraph,
        --and that the table is in the paragraph by itself).
```

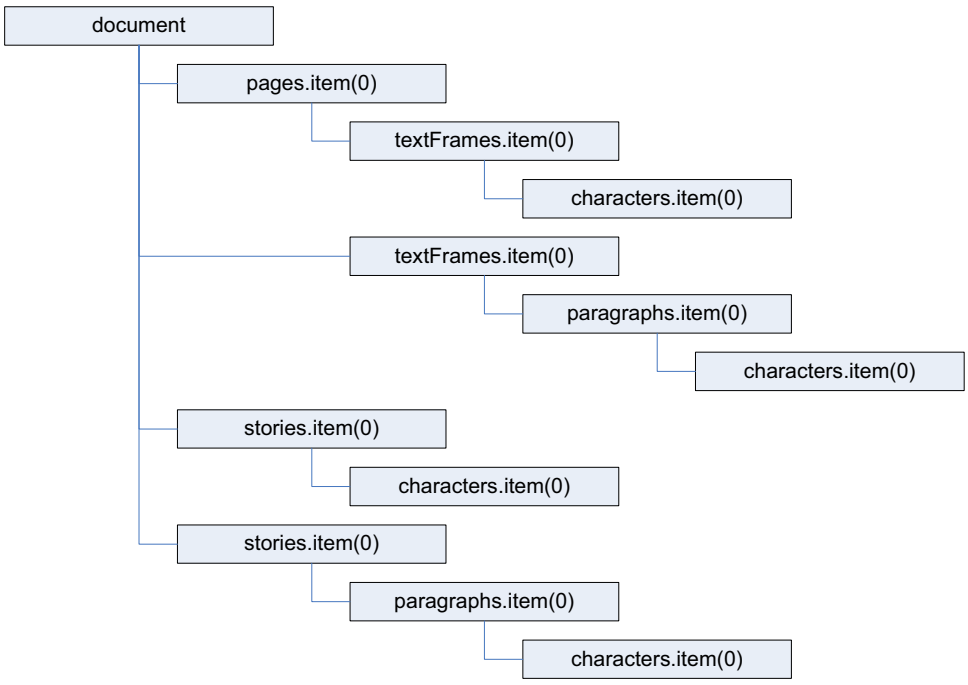
```
                        set myTable to table 1 of myParagraph
                        my myWriteToFile("<table border = 1>", myTextFile, true)
                        repeat with myRowCounter from 1 to (count rows of myTable)
                            my myWriteToFile("<tr>", myTextFile, true)
                            repeat with myColumnCounter from 1 to
                            (count columns of myTable)
                                if myRowCounter = 1 then
                                    set myString to "<th>"
                                    set myString to myString & text 1 of cell myColumnCounter
                                    of row myRowCounter of myTable
                                    set myString to myString & "</th>"
                                else
                                    set myString to "<td>"
                                    set myString to myString & text 1 of cell myColumnCounter
                                    of row myRowCounter of myTable
                                    set myString to myString & "</td>"
                                end if
                                my myWriteToFile(myString, myTextFile, true)
                            end repeat
                            my myWriteToFile("</tr>" & return, myTextFile, true)
                        end repeat
                        my myWriteToFile("</table>" & return, myTextFile, true)
                    end if
                end repeat
            end repeat
        end if
    end if
end tell
```

テキストオブジェクト

次の図は、InDesign のテキストオブジェクトモデルを図式化したものです。この図にもあるように、テキストオブジェクトには レイアウトオブジェクト（テキストフレーム）とテキストストリームオブジェクト（ストーリー、挿入点、文字、単語など）の 2 つの種類があります。



特定のテキストオブジェクトへの参照を取得するには、様々な方法があります。次の図に、新規ドキュメントの最初のページの最初のテキストフレームにある最初の文字を参照する方法をいくつか示します。



テキストストリームオブジェクトの `parent` は、そのオブジェクトが含まれているストーリーです。テキストオブジェクトが含まれているテキストフレーム（またはテキストフレームのコレクション）への参照を取得するには、`parent text frames` プロパティを使用します。

テキストフレームの parent は、通常、そのテキストフレームが含まれているページまたはスプレッドです。グループ内にあるテキストフレームや、別のページアイテム内にペーストされたテキストフレームの parent は、そのテキストフレームが含まれているページアイテムです。アンカー付きフレームに変換されたテキストフレームの parent は、そのアンカー付きフレームが含まれている文字です。

テキスト選択の操作

テキスト関連のスクリプトでよく行われる操作の 1 つが、テキスト選択に対する操作です。現在の選択がテキスト選択かどうかを調べる方法を、次のスクリプトに示します。他の多くのサンプルスクリプトと異なり、このスクリプトは実際には何も行いません。このスクリプトでは、選択をフィルタリングする方法のみが示されています（完全なスクリプトについては、TextSelection を参照してください）。

```
if (count documents) is not equal to 0 then
    --If the selection contains more than one item, the selection
    --is not text selected with the Type tool.
    set mySelection to selection
    if (count mySelection) is not equal to 0 then
        --Evaluate the selection based on its type.
        set myTextClasses to {insertion point, word, text style range,
        line, paragraph, text column, text, story}
        if class of item 1 of selection is in myTextClasses then
            --The object is a text object; display the text object type.
            --A practical script would do something with the selection,
            --or pass the selection on to a function.
            display dialog ("Selection is a text object.")
            --If the selection is inside a note, the parent of the selection
            --will be a note object.
            if class of parent of item 1 of selection is note then
                display dialog ("Selection is inside a note.")
            end if
        else if class of item 1 of selection is text frame then
            display dialog ("Selection is a text frame")
        else
            display dialog ("Selected item is not a text object.")
        end if
    else
        display dialog ("Please select some text and try again.")
    end if
else
    display dialog ("No documents are open.")
end if
```

テキストの移動とコピー

move メソッドを使用して、テキスト内の別の場所にテキストオブジェクトを移動することができます。テキストをコピーするには、duplicate メソッドを使用します（パラメーターは move メソッドと同じです）。次のスクリプトにその例を示します（完全なスクリプトについては、MoveText を参照してください）。

```
--Given a document myDocument" with four text frams on page 1...
set myTextFrameA to text frame 4 of page 1 of myDocument
set myTextFrameB to text frame 3 of page 1 of myDocument
set myTextFrameC to text frame 2 of page 1 of myDocument
set myTextFrameD to text frame 1 of page 1 of myDocument
--Note that moving text removes it from its original location.
tell parent story of myTextFrameD
    --Move WordC between the words in TextFrameC.
    move paragraph 4 to before word -1 of paragraph 1 of parent story of myTextFrameC
    --Move WordB after the word in TextFrameB.
    move paragraph 3 to after insertion point -1 of myTextFrameB
    --Move WordA to before the word in TextFrameA.
    move paragraph 2 to before word 1 in myTextFrameA
end tell
```

フォーマットされたテキストをドキュメント間で転送する場合にも、move メソッドが使用できます。コピーして貼り付けるよりも、move メソッドや duplicate メソッドを使用するほうが優れています。コピーして貼り付けるためには、ドキュメントを表示して、コピーするテキストを選択しなければなりません。move や duplicate を使用するほうが速くて確実です。move や duplicate を使用して、ドキュメント間でテキストを移動する方法を、次のスクリプトに示します （ここでは、myGetBounds 関数は省略しています。この関数については、83 ページの「テキストフレームの作成」または MoveTextBetweenDocuments というチュートリアルスクリプトを参照してください）。

```
--Given a document "mySourceDocument"...
set mySourceStory to story 1 of mySourceDocument
--Create a new document to move the text to.
set myTargetDocument to make document
set myPage to page 1 of myTargetDocument
tell myPage
    set myTextFrame to make text frame with properties
    {geometric bounds:my myGetBounds(myTargetDocument, myPage)}
end tell
set myTargetStory to story 1 of myTargetDocument
set contents of myTargetStory to "This is the target text. Insert the source text after this
paragraph." & return
duplicate paragraph 1 of mySourceStory to after insertion point -1 of myTargetStory
```

テキストをコピーして貼り付ける必要がある場合は、アプリケーションの copy メソッドを使用できます。この場合は、コピーする前にテキストを選択しておく必要があります。繰り返しになりますが、コピーして貼り付ける方法は最後の手段だと考えてください。他の方法を使用するほうが速くて確実であり、ドキュメントを表示する必要がありません （ここでは、myGetBounds 関数は省略しています。この関数については、83 ページの「テキストフレームの作成」または CopyPasteText というチュートリアルスクリプトを参照してください）。

```
--Given an open document with a text frame on page 1
set myDocumentA to document 1
set myTextFrameA to text frame 1 of page 1 of myDocumentA
--Create another example document.
set myDocumentB to make document
set myPageB to page 1 of myDocumentB
tell myPageB
    set myTextFrameB to make text frame with properties {geometric bounds:my
myGetBounds(myDocumentB, myPageB)}
end tell
--Make document A the active document.
set active document to myDocumentA
--Select the text.
select text 1 of parent story of myTextFrameA
copy
--Make document B the active document.
set active document to myDocumentB
--Select the insertion point at which you want to paste the text.
select insertion point -1 of myTextFrameB
paste
```

フォーマットされていないテキストをテキストオブジェクト間でコピーする方法としては、コピーするテキストオブジェクトの contents プロパティを取得して、その文字列を別のテキストオブジェクトの contents プロパティに設定する方法があります。次のスクリプトにその例を示します（完全なスクリプトについては、CopyUnformattedText を参照してください）。

```
tell application "Adobe InDesign CS6"
    set myDocument to document 1
    set myPage to page 1 of myDocument
    tell myPage
        set myTextFrameA to make text frame with properties
        {geometric bounds:{72, 72, 144, 288}}
        set contents of myTextFrameA to "This is a formatted string."
        set font style of text 1 of parent story of myTextFrameA to "Bold"
        set myTextFrameB to make text frame with properties
        {geometric bounds:{228, 72, 300, 288}}
        set contents of myTextFrameB to "This is the destination text frame.
        Text pasted here will retain its formatting."
        set font style of text 1 of myTextFrameB to "Italic"
    end tell
    --Copy from one frame to another using a simple copy.
    select text 1 of myTextFrameA
    copy
    select insertion point -1 of myTextFrameB
    paste
    --Create another text frame on the active page.
    tell myPage
        set myTextFrameC to make text frame with properties
        {geometric bounds:{312, 72, 444, 288}}
    end tell
    set contents of myTextFrameC to "Text copied here will take on
    the formatting of the existing text."
    set font style of text 1 of parent story of myTextFrameC to "Italic"
    --Copy the unformatted string from text frame A to the end of text frame C (note
    --that this doesn't really copy the text it replicates the text string from one
    --text frame in another text frame):
    set contents of insertion point -1 of myTextFrameC to contents of text 1 of parent story of
myTextFrameA
end tell
```

テキストオブジェクトと反復処理

一連のテキストオブジェクトに対して繰り返して処理を行っている間に、テキストの移動、削除、追加を行うと、テキスト参照に問題が発生することがあります。このような問題が発生する例を、次のスクリプトに示します（ここでは、myGetBounds 関数は省略しています。この関数については、83 ページの「テキストフレームの作成」または TextIterationWrong というチュートリアルスクリプトを参照してください）。

```
--The following for loop will fail to format all of the paragraphs
--and then generate an error.
repeat with myParagraphCounter from 1 to (count paragraphs of myStory)
    if contents of word 1 of paragraph myParagraphCounter of myStory is "Delete" then
        tell paragraph myParagraphCounter of myStory to delete
    else
        set point size of paragraph myParagraphCounter of myStory to 24
    end if
end repeat
```

この例では、フォーマットされない段落がいくつか発生することになります。その理由は次のとおりです。このスクリプトのループは、ストーリー内の最初の段落から最後の段落まで、段落に対して繰り返し処理を行います。そうすると、「Delete」という単語で始まる段落があれば削除されます。このスクリプトで第 2 段落が削除されると、第 3 段落がその場所に移動します。次にループカウンターが 3 になると、先ほどまで第 4 段落であった段落が処理されます。第 3 段落であった段落はこの時点で第 2 段落になっているので、処理は行われません。

この問題を回避するには、次のスクリプトのように、テキストオブジェクトを逆順で処理します（ここでは、myGetBounds 関数は省略しています。この関数については、83 ページの「テキストフレームの作成」または TextIterationRight というチュートリアルスクリプトを参照してください）。

```
--By iterating backwards we can avoid the error.
repeat with myParagraphCounter from (count paragraphs of myStory) to 1 by -1
    if contents of word 1 of paragraph myParagraphCounter of myStory is "Delete" then
        tell paragraph myParagraphCounter of myStory to delete
    else
        set point size of paragraph myParagraphCounter of myStory to 24
    end if
end repeat
```

テキストフレームの操作

前の節ではテキストストリームオブジェクトの操作について説明しましたが、この節ではテキストフレーム（InDesign ドキュメント内のテキストを格納するページレイアウトアイテム）について説明します。

テキストフレームのリンク

InDesign スクリプティングで、テキストフレームのリンクをたどっていく際に重要になるのが、テキストフレームの `nextTextFrame` プロパティと `previousTextFrame` プロパティです。これらのプロパティは、InDesign テキストフレームの入力ポートと出力ポートに相当します。次のスクリプトにその例を示します（完全なスクリプトについては、LinkTextFrames を参照してください）。

```
--Given a document "myDocument" with two unlinked text frames on page 1...
set myTextFrameA to text frame 2 of page 1 of document 1
set myTextFrameB to text frame 1 of page 1 of document 1
--Link TextFrameA to TextFrameB using the next text frame property.
set next text frame of myTextFrameA to myTextFrameB
--Add a page.
tell myDocument
    set myNewPage to make page
end tell
--Create another text frame on the new page.
tell myNewPage
    set myTextFrameC to make text frame with properties
    {geometric bounds:{72, 72, 144, 144}}
    --Link TextFrameC to TextFrameB using the previousTextFrame property.
    set previous text frame of myTextFrameC to myTextFrameB
    --Fill the text frames with placeholder text.
    set contents of myTextFrameA to placeholder text
end tell
```

テキストフレームのリンク解除

テキストフレームのリンクを解除する方法を、次のスクリプト例に示します（完全なスクリプトについては、UnlinkTextFrames を参照してください）。

```
--Unlink the two text frames.
set next text frame of myTextFrameA to nothing
```

ストーリーからのフレームの削除

InDesign では、ストーリーからフレームを削除しても、フレーム内のテキストは削除されません（そのフレームがストーリー内の唯一のフレームである場合を除く）。ストーリー内の他のフレームを維持したまま、特定のフレームとそこに含まれているテキストをストーリーから削除する方法を、次のスクリプトに示します（完全なスクリプトについては、BreakFrame を参照してください）。

```
set myObjectList to {}
--Script does nothing if no documents are open or if no objects are selected.
tell application "Adobe InDesign CS6"
    if (count documents) is not equal to 0 then
        set mySelection to selection
        if (count mySelection) is not equal to 0 then
            --Process the objects in the selection to create a list of
            --qualifying objects (text frames).
            repeat with myCounter from 1 to (count mySelection)
                if class of item myCounter of mySelection is text frame then
                    set myObjectList to myObjectList & item myCounter of mySelection
                else if class of item myCounter of mySelection is in {text,
                insertion point, character, word, line, text style range, paragraph,
                text column} then
                    set myObject to item 1 of parent text frames of item myCounter
                    of mySelection
                    set myObjectList to myObjectList & myObject
                end if
            end repeat
            --If the object list is not empty, pass it on to the handler
            --that does the real work.
            if (count myObjectList) is not equal to 0 then
                my myBreakFrames(myObjectList)
            end if
        end if
    end if
end tell
```

このスクリプトで参照されている myBreakFrames ハンドラーを次に示します。

```
on myBreakFrames(myObjectList)
    repeat with myCounter from 1 to (count myObjectList)
        my myBreakFrame(item myCounter of myObjectList)
    end repeat
end myBreakFrames
on myBreakFrame(myTextFrame)
    tell application "Adobe InDesign CS6"
        if next text frame of myTextFrame is not equal to nothing and previous text frame of
myTextFrame is not equal to nothing then
            set myNewFrame to duplicate myTextFrame
            if contents of myTextFrame is not equal to "" then
                tell text 1 of myTextFrame to delete
            end if
            delete myTextFrame
        end if
    end tell
end myBreakFrame
```

ストーリー内のすべてのフレームの分割

ストーリー内のすべてのフレームを独立した別個のストーリーに分割して、リンクされていないテキストフレームが各ストーリーに 1 つだけ含まれるようにする方法を、次のスクリプトに示します（完全なスクリプトについては、SplitStory を参照してください）。

```
on mySplitStory(myStory)
    tell application "Adobe InDesign CS6"
        tell document 1
            local myTextContainers
            set myTextContainers to text containers in myStory
            if (count myTextContainers) is greater than 1 then
                repeat with myCounter from (count myTextContainers) to 1 by -1
                    set myTextFrame to item myCounter of myTextContainers
                    tell myTextFrame
                        duplicate
                        if text 1 of myTextFrame is not equal to "" then
                            tell text 1 of myTextFrame
                                delete
                            end tell
                        end if
                        delete
                    end tell
                end repeat
            end if
        end tell
    end tell
end mySplitStory
```

アンカー付きフレームの作成

アンカー付きフレーム（インラインフレームとも呼ばれます）を作成するには、テキストの特定の位置（通常は挿入点）にテキストフレーム（または長方形、楕円形、多角形、グラフィックの線）を作成します。次のスクリプトにその例を示します（完全なスクリプトについては、AnchoredFrame を参照してください）。

```
--Given a document "myDocument" with a text frame on page 1...
set myPage to page 1 of myDocument
set myTextFrame to text frame 1 of myPage
tell insertion point 1 of paragraph 1 of myTextFrame
    set myInlineFrame to make text frame
end tell
--Recompose the text to make sure that getting the
--geometric bounds of the inline graphic will work.
tell text 1 of myTextFrame to recompose
--Get the geometric bounds of the inline frame.
set myBounds to geometric bounds of myInlineFrame
--Set the width and height of the inline frame. In this example, we'll
--make the frame 24 points tall by 72 points wide.
set e1 to item 1 of myBounds
set e2 to item 2 of myBounds
set e3 to (item 1 of myBounds) + 24
set e4 to (item 2 of myBounds) + 72
set geometric bounds of myInlineFrame to {e1, e2, e3, e4}
set contents of myInlineFrame to "This is an inline frame."
tell insertion point 1 of paragraph 2 of myTextFrame
    set myAnchoredFrame to make text frame
end tell
--Recompose the text to make sure that getting the
--geometric bounds of the inline graphic will work.
tell text 1 of myTextFrame to recompose
--Get the geometric bounds of the inline frame.
set myBounds to geometric bounds of myAnchoredFrame
--Set the width and height of the inline frame. In this example, we'll
--make the frame 24 points tall by 72 points wide.
set e1 to item 1 of myBounds
```

```
set e2 to item 2 of myBounds
set e3 to (item 1 of myBounds) + 24
set e4 to (item 2 of myBounds) + 72
set geometric bounds of myAnchoredFrame to {e1, e2, e3, e4}
set contents of myAnchoredFrame to "This is an anchored frame."
tell anchored object settings of myAnchoredFrame
    set anchored position to anchored
    set anchor point to top left anchor
    set horizontal reference point to anchor location
    set horizontal alignment to left align
    set anchor xoffset to 72
    set vertical reference point to line baseline
    set anchor yoffset to 24
    set anchor space above to 24
end tell
```

テキストフレームをコンテンツに合わせる

テキストフレームにルールを設定し、ユーザーのテキスト入力に応じてテキストフレームを伸縮させる方法を、次のスクリプトに示します（完全なスクリプトについては、PersistedTextFrameFill を参照してください）。

```
tell application "Adobe InDesign CS6"
set myDocument to document 1
set myPage to page 1 of myDocument
tell myPage
    set myTextFrame to text frame 4 of myPage
    --Now add text at the end of the text frame.
    set myString to return & "this is some overset text"
    tell insertion point -1 of myTextFrame to set contents to myString
    set myString to contents of text 1 of myTextFrame
    display dialog
    ("The last paragraph in this alert should be ¥"This is some overset text¥". Is it?"
        & return & myString)
    set myString to contents of parent story of myTextFrame
    display dialog
    ("The last paragraph in this alert should be ¥"This is some overset text¥". Is it?"
        & return & myString)
end tell
set myPage to page 1 of myDocument
tell myPage
    set myTextFrame2 to text frame 3 of myPage
    --Set auto sizing dimension of the text frame
    tell text frame preferences of myTextFrame2
        set auto sizing dimension to height and width proportionally
    end tell
    --Now add text at the end of the text frame.
    set myString to return & "this is some overset text"
    tell insertion point -1 of myTextFrame2 to set contents to myString
end tell
set myPage to page 1 of myDocument
tell myPage
    set myTextFrame3 to text frame 2 of myPage
    --Set  auto sizing dimension of the text frame
    --useMinimumHeightForAutoSizing and useNoLineBreaksForAutoSizing
    tell text frame preferences of myTextFrame3
        set auto sizing dimension to height only
        set use minimum height for auto sizing to true
        set use no line breaks for auto sizing to true
    end tell
    --Now add text at the end of the text frame.
```

```
    set myString to return & "this is some overset text"
    tell insertion point -1 of myTextFrame3 to set contents to myString
end tell
set myPage to page 1 of myDocument
tell myPage
    set myTextFrame4 to text frame 1 of myPage
    --Set  auto size dimension of the text frame and auto sizing reference point
    tell text frame preferences of myTextFrame4
        set auto sizing dimension to width only
        set auto sizing reference point to top left position
    end tell
    --Now add text at the end of the text frame.
    set myString to return & "this is some overset text"
    tell insertion point -1 of myTextFrame4 to set contents to myString
end tell
```

テキストのフォーマット

この章の前の節では、ドキュメントにテキストを追加し、テキストフレームをリンクし、ストーリーやテキストオブジェクトを操作しました。この節では、テキストにフォーマットを適用します。InDesign のすべてのフォーマット機能は、スクリプティングで操作することができます。

テキストデフォルトの設定

テキストデフォルトは、アプリケーションとドキュメントの両方で設定できます。アプリケーションのテキストデフォルトは、新しく作成されるすべてのドキュメントのテキストデフォルトとして使用されます。ドキュメントのテキストデフォルトは、そのドキュメントで新しく作成されるすべてのテキストオブジェクトのフォーマットとして使用されます （完全なスクリプトについては、TextDefaults を参照してください）。

```
set horizontal measurement units of view preferences to points
set vertical measurement units of view preferences to points
--To set the text formatting defaults for a document, replace "app"
--in the following lines with a reference to a document.
tell text defaults
    set alignToBaseline to true
    --Because the font might not be available, it's usually best
    --to apply the font within a try...catch structure. Fill in the
    --name of a font on your system.
    try
        set appliedFont to font "Minion Pro"
    end try
    --Because the font style might not be available, it's usually best
    --to apply the font style within a try...catch structure.
    try
        set font style to "Regular"
    end try
    --Because the language might not be available, it's usually best
    --to apply the language within a try...catch structure.
    try
        set applied language to "English: USA"
    end try
    set autoLeading to 100
    set balsanceRaggedLines to false
    set baselineShift to 0
    set capitalization to normal
    set composer to "Adobe Paragraph Composer"
    set desiredGlyphScaling to 100
    set desiredLetterSpacing to 0
    set desiredWordSpacing to 100
    set drop cap characters to 0
    if drop cap characters is not equal to 0 then
        dropCapLines to 3
```

```
    --Assumes that the application has a default character style named "myDropCap"
    set drop cap style to character style "myDropCap"
end if
set fill color to "Black"
set fill tint to 100
set first line indent to 14
set grid align first line only to false
set horizontal scale to 100
set hyphenate after first to 3
set hyphenate before last to 4
set hyphenate capitalized words to false
set hyphenate ladder limit to 1
set hyphenate words longer than to 5
set hyphenation to true
set hyphenation zone to 36
set hyphen weight to 9
set justification to left align
set keep all lines together to false
set keep lines together to true
set keep first lines to 2
set keep last lines to 2
set keep with next to 0
set kerning method to "Optical"
set leading to 14
set left indent to 0
set ligatures to true
set maximum glyph scaling to 100


set maximum letter spacing to 0
set maximum word spacing to 160
set minimum glyph scaling to 100
set minimum letter spacing to 0
set minimum word spacing to 80
set no break to false
set OTF contextual alternate to true
set OTF discretionary ligature to false
set OTF figure style to proportional oldstyle
set OTF fraction to true
set OTF historical to false
set OTF ordinal to false
set OTF slashed zero to false
set OTF swash to false
set OTF titling to false
set overprint fill to false
set overprint stroke to false
set pointSize to 11
set position to normal
set right indent to 0
set rule above to false
if rule above = true then
    set rule above color to color "Black"
    set rule above gap color to swatch "None"
    set rule above gap overprint to false
    set rule above gap tint to 100
    set rule above left indent to 0
    set rule above line weight to 0.25
    set rule above offset to 14
    set rule above overprint to false
    set rule above right indent to 0
    set rule above tint to 100
    set rule above type to stroke style "Solid"
    set rule above width to column width
end if
set rule below to false
if rule below = true then
```

```
            set rule below color to color "Black"
            set rule below gap color to swatch "None"
            set rule below gap overprint to false
            set rule below gap tint to 100
            set rule below left indent to 0
            set rule below line weight to 0.25
            set rule below offset to 14
            set rule below overprint to false
            set rule below right indent to 0
            set rule below tint to 100
            set rule below type to stroke style "Solid"
            set rule below width to column width
        end if
        set single word justification to left align
        set skew to 0
        set space after to 0
        set space before to 0
        set start paragraph to anywhere
        set strike thru to false
        if strike thru = true then
            set strike through color to color "Black"
            set strike through gap color to swatch "None"
            set strike through gap overprint to false


            set strike through gap tint to 100
            set strike through offset to 3
            set strike through overprint to false
            set strike through tint to 100
            set strike through type to stroke style "Solid"
            set strike through weight to 0.25
        end if
        set stroke color to "None"
        set stroke tint to 100
        set stroke weight to 0
        set tracking to 0
        set underline to false
        if underline = true then
            set underline color to color "Black"
            set underline gap color to swatch "None"
            set underline gap overprint to false
            set underline gap tint to 100
            set underline offset to 3
            set underline overprint to false
            set underline tint to 100
            set underline type to stroke style "Solid"
            set underline weight to 0.25
        end if
        set vertical scale to 100
    end tell
```

フォントの操作

InDesign アプリケーションオブジェクトのフォントコレクションには、InDesign で利用できるすべてのフォントが含まれています。これとは対照的に、ドキュメントのフォントコレクションには、そのドキュメントで使用されているフォントのみが含まれています。ドキュメントのフォントコレクションには、環境に無いフォント（ドキュメントで使用されているが InDesign では利用できないフォント）も含まれます。アプリケーションフォントとドキュメントフォントの違いを調べる方法を、次のスクリプトに示します（ここでは、`myGetBounds` 関数は省略しています。完全なスクリプトについては、FontCollections を参照してください）。

```
set myApplicationFonts to the name of every font
set myDocument to active document
tell myDocument
    set myPage to page 1
    tell myPage
        set myTextFrame to make text frame with properties {geometric bounds:my
myGetBounds(myDocument, myPage)}
    end tell
    set myStory to parent story of myTextFrame
    set myDocumentFonts to name of every font
end tell
set myString to "Document Fonts:" & return
repeat with myCounter from 1 to (count myDocumentFonts)
    set myString to myString & (item myCounter) of myDocumentFonts & return
end repeat
set myString to myString & return & "Application Fonts:" & return
repeat with myCounter from 1 to (count myApplicationFonts)
    set myString to myString & (item myCounter) of myApplicationFonts & return
end repeat
set contents of myStory to myString
```

注意 : フォント名は通常、*familyName*<tab>*fontStyle* という形式で示されます。*familyName* はフォントファミリー名、<tab> はタブ文字、*fontStyle* はフォントスタイル名です。例えば、次のようになります。

```
"Adobe Caslon Pro<tab>Semibold Italic"
```

フォントの適用

テキストのある範囲にローカルフォントを適用するには、appliedFont プロパティを使用します。次のスクリプトにその例を示します（チュートリアルスクリプトの ApplyFont より）。

```
--Given a font name "myFontName" and a text object "myText"...
set applied font of myText to myFontName
```

また、次のスクリプトのように、フォントファミリー名とフォントスタイルを指定してフォントを適用することもできます。

```
tell myText
set applied font to "Adobe Caslon Pro"
set font style to "Semibold Italic"
end tell
```

テキストプロパティの変更

InDesign のテキストオブジェクトには、そのフォーマット属性に対応する多数のプロパティが用意されています。単なる挿入点にも、テキストのフォーマットに影響を与える多数のプロパティ（その挿入点が含まれている段落とほぼ同じプロパティ）が用意されています。SetTextProperties というチュートリアルスクリプトには、テキストオブジェクトのすべてのプロパティを設定する方法が示されています。次のスクリプトにその例を示します。

```
--Given a document "myDocument" containing a story...
set myStory to story 1 of myDocument
tell character 1 of myStory
    set align to baseline to false
    set applied character style to character style "[None]" of myDocument
    set applied font to "Minion ProRegular"
    set applied language to "English: USA"
    set applied numbering list to "[Default]"
    set applied paragraph style to paragraph style "[No Paragraph Style]" of myDocument
    set auto leading to 120
    set balance ragged lines to no balancing
    set baseline shift to 0
    set bullets alignment to left align
    set bullets and numbering list type to no list
    set bullets character style to character style "[None]" of myDocument
    set bullets text after to "^t"
    set capitalization to normal
    set composer to "Adobe Paragraph Composer"
    set desired glyph scaling to 100
    set desired letter spacing to 0
    set desired word spacing to 100
    set drop cap characters to 0
    set drop cap lines to 0
    set drop cap style to character style "[None]" of myDocument
    set dropcap detail to 0
    set fill color to color "Black" of myDocument
    set fill tint to -1
    set first line indent to 0
    set font style to "Regular"
    set gradient fill angle to 0
    set gradient fill length to -1
    set gradient fill start to [0, 0]
    set gradient stroke angle to 0
    set gradient stroke length to -1
    set gradient stroke start to [0, 0]
    set grid align first line only to false
    set horizontal scale to 100
    set hyphen weight to 5
    set hyphenate across columns to true
    set hyphenate after first to 2
    set hyphenate before last to 2
    set hyphenate capitalized words to true
    set hyphenate ladder limit to 3
    set hyphenate last word to true
    set hyphenate words longer than to 5
    set hyphenation to true
    set hyphenation zone to 3
    set ignore edge alignment to false
    set justification to left align
    set keep all lines together to false
    set keep first lines to 2
    set keep last lines to 2
    set keep lines together to false
    set keep rule above in frame to false
    set keep with next to 0
    set kerning method to "Optical"
    set last line indent to 0
    set leading to 12
    set left indent to 0
    set ligatures to true
    set maximum glyph scaling to 100
    set maximum letter spacing to 0
    set maximum word spacing to 133
    set minimum glyph scaling to 100
    set minimum letter spacing to 0
    set minimum word spacing to 80
```

```
set no break to false
set numbering alignment to left align
set numbering apply restart policy to true
set numbering character style to character style "[None]" of myDocument
set numbering continue to true
set numbering expression to "^#.^t"
set numbering format to "1, 2, 3, 4..."
set numbering level to 1
set numbering start at to 1
set OTF contextual alternate to true
set OTF discretionary ligature to false
set OTF figure style to proportional lining
set OTF fraction to false
set OTF historical to false
set OTF locale to true
set OTF mark to true
set OTF ordinal to false
set OTF slashed zero to false


set OTF stylistic sets to 0
set OTF swash to false
set OTF titling to false
set overprint fill to false
set overprint stroke to false
set point size to 72
set position to normal
set positional form to none
set right indent to 0
set rule above to false
set rule above color to "Text Color"
set rule above gap color to swatch "None" of myDocument
set rule above gap overprint to false
set rule above gap tint to 100
set rule above left indent to 0
set rule above line weight to 0.25
set rule above offset to 14
set rule above overprint to false
set rule above right indent to 0
set rule above tint to 100
set rule above type to stroke style "Solid" of myDocument
set rule above width to column width
set rule below to false
set rule below color to "Text Color"
set rule below gap color to swatch "None" of myDocument
set rule below gap overprint to false
set rule below gap tint to 100
set rule below left indent to 0
set rule below line weight to 0.25
set rule below offset to 14
set rule below overprint to false
set rule below right indent to 0
set rule below tint to 100
set rule below type to stroke style "Solid" of myDocument
set rule below width to column width
set single word justification to left align
set skew to 0
set space after to 0
```

```
    set space before to 0
    set start paragraph to anywhere
    set strike thru to false
    set strike through color to color "Black" of myDocument
    set strike through gap color to swatch "None" of myDocument
    set strike through gap overprint to false
    set strike through gap tint to 100
    set strike through offset to 3
    set strike through overprint to false
    set strike through tint to 100
    set strike through type to stroke style "Solid" of myDocument
    set strike through weight to 0.25
    set stroke color to swatch "None" of myDocument
    set stroke tint to -1


    set stroke weight to 1
    set tracking to 0
    set tracking to 0
    set underline to false
    set underline color to color "Black" of myDocument
    set underline gap color to swatch "None" of myDocument
    set underline gap overprint to false
    set underline gap tint to 100
    set underline offset to 3
    set underline overprint to false
    set underline tint to 100
    set underline type to stroke style "Solid" of myDocument
    set underline weight to 0.25
    set vertical scale to 100
end tell
```

テキストカラーの変更

テキスト文字の塗りと線にカラーを適用できます。次のスクリプトにその例を示します（チュートリアルスクリプトの TextColors より）。

```
set myStory to story 1 of myDocument
set myColorA to color "DGC1_664a" of myDocument
set myColorB to color "DGC1_664b" of myDocument
tell paragraph 1 of myStory
    set point size to 72
    set justification to center align
    --Apply a color to the fill of the text.
    set fill color to myColorA
    set stroke color to myColorB
end tell
tell paragraph 2 of myStory
    set stroke weight to 3
    set point size to 144
    set justification to center align
    set fill color to myColorB
    set stroke color to myColorA
    set stroke weight to 3
end tell
```

スタイルの作成と適用

前述のいくつかの例でローカルフォーマットを適用する方法を示しましたが、ここでは、文字スタイルや段落スタイルを使用してテキストをフォーマットする方法について説明します。スタイルを使用すると、フォーマットされたテキストとスタイルの間にリンクが作成されます。そのため、スタイルの再定義、特定のスタイルでフォーマットされたテキストの収集、テキストの検索や変更などが簡単に行えます。段落スタ

イルや文字スタイルは、テキストのフォーマット作業を効率化する上で重要な役割を果たします。段落と文字スタイルはテキストにフォーマットを設定する場合に生産性の鍵となるものであり、テキストにフォーマットを適用するどんなスクリプトにとっても中心的な存在になるはずです。

段落スタイルや文字スタイルを作成して適用する方法を、次のスクリプトに示します（完全なスクリプトについては、CreateStyles を参照してください）。

```
--Given a document "myDocument" containing a story and the color "Red"...
tell myDocument
    set myColor to color "Red"
    set myStory to story 1 of myDocument
    set contents of myStory to "Normal text. Text with a character style applied to it.
    More normal text."
    --Create a character style named "myCharacterStyle" if
    --no style by that name already exists.
    try
        set myCharacterStyle to character style "myCharacterStyle"
    on error
        --The style did not exist, so create it.
        set myCharacterStyle to make character style with properties
        {name:"myCharacterStyle"}
    end try
    --At this point, the variable myCharacterStyle contains a reference to a character
    --style object, which you can now use to specify formatting.
    set fill color of myCharacterStyle to myColor
    --Create a paragraph style named "myParagraphStyle" if
    --no style by that name already exists.
    try
        set myParagraphStyle to paragraph style "myParagraphStyle"
    on error
        --The paragraph style did not exist, so create it.
        set myParagraphStyle to make paragraph style with properties
        {name:"myParagraphStyle"}
    end try
    --At this point, the variable myParagraphStyle contains a reference to a paragraph
    --style object, which you can now use to specify formatting.
    --(Note that the story object does not have the apply paragraph style method.)
    tell text 1 of myStory
        apply paragraph style using myParagraphStyle
        tell text from character 13 to character 54
            apply character style using myCharacterStyle
        end tell
    end tell
end tell
```

テキストオブジェクトの `appliedParagraphStyle` プロパティを使用せずに `applyParagraphStyle` メソッドを使用する利点は、既存のフォーマットをオーバーライドできる点にあります。スタイルにプロパティを設定しても、ローカルフォーマットは変更されずに保持されます。

新規ドキュメントの作成時に、スタイルが存在するかどうかを確認しているのは、アプリケーションのデフォルトスタイルとして同じ名前のスタイルが存在している可能性があるからです。その場合、同じ名前で新しいスタイルを作成しようとするとエラーが発生します。

先頭文字スタイル（Nested Style）を使用すると、文字スタイルのフォーマットを特定のパターンに従って段落に適用することができます。先頭文字スタイルを含む段落スタイルの作成方法を、次のスクリプトに示します（完全なスクリプトについては、NestedStyles を参照してください）。

```
--Given a document "myDocument"...
--Get the last paragraph style.
set myParagraphStyle to paragraph style -1 of myDocument
--Get the last character style
set myCharacterStyle to character style -1 of myDocument
--At this point, the variable myParagraphStyle contains a reference to a paragraph
--style object. Next, add a nested style to the paragraph style.
tell myParagraphStyle
    set myNestedStyle to make nested style with properties {applied character
style:myCharacterStyle, delimiter:".", inclusive:true, repetition:1}
end tell
set myPage to page 1 of myDocument
set myTextFrame to text frame 1 of myPage
--Apply the paragraph style to the story so that we can see the
--effect of the nested style we created.
--(Note that the story object does not have the apply paragraph style method.)
set myStory to parent story of myTextFrame
tell text 1 of myStory
    apply paragraph style using myParagraphStyle
end tell
```

スタイルの削除

ユーザーインターフェイスでスタイルを削除する際には、タグ付けされたスタイルでテキストのフォーマット方法を選択することができますが、InDesign スクリプティングでもこれと同じことが行えます。次のスクリプトにその例を示します（チュートリアルスクリプトの RemoveStyle より）。

```
--Given a document "myDocment" with paragraph styles named
--"myParagraphStyleA" and "myParagraphStyleB", remove the
--paragraph style "myParagraphStyleA" and replace with
--"myParagraphStyleB."
tell myDocument
    delete paragraph style "myParagraphStyleA" replacing with paragraph style "myParagraphStyleB"
end tell
```

段落スタイルや文字スタイルの読み込み

他の InDesign ドキュメントから文字スタイルや段落スタイルを読み込むことができます。次のスクリプトにその例を示します（チュートリアルスクリプトの ImportTextStyles より）。

```
--You'll have to fill in a valid file path for your system.
set myFilePath to "Macintosh HD:scripting:styles.indd"
--Create a new document.
set myDocument to make document
tell myDocument
    --Import the styles from the saved document.
    --importStyles parameters:
    --Format options for text styles are:
    --    paragraph styles format
    --    character styles format
    --    text styles format
    --From as file or string
    --Global Strategy options are:
    --    do not load the style
    --    load all with overwrite
    --    load all with rename
    import styles format text styles format from myFilePath global strategy load all with
overwrite
end tell
```

テキストの検索と変更

検索／変更機能は、テキストを操作するための最も強力なツールの 1 つであり、スクリプティングに完全に対応しています。スクリプトを使用すれば、ユーザーインターフェイスではできないことも可能になります。InDesign では、次の 3 つの方法でテキストを検索できます。

- テキストやテキストフォーマットを検索して、別のテキストやテキストフォーマットに変更できます。この場合は、`findTextPreferences` オブジェクトや `changeTextPreferences` オブジェクトを使用して、`findText` メソッドや `changeText` メソッドのパラメーターを指定します。
- 正規表現を使用したテキスト検索（grep）を実行できます。この場合は、`findGrepPreferences` オブジェクトや `changeGrepPreferences` オブジェクトを使用して、`findGrep` メソッドや `changeGrep` メソッドのパラメーターを指定します。
- 特定の字形（とそのフォーマット）を検索して、別の字形やフォーマットに置換できます。この場合は、`findGlyphPreferences` オブジェクトや `changeGlyphPreferences` オブジェクトを使用して、`findGlyph` メソッドや `changeGlyph` メソッドのパラメーターを指定します。

どの検索／変更メソッドにも、`ReverseOrder` という 1 つのオプションパラメーターがあります。このパラメーターで、検索結果を返す順序を指定できます。検索／変更操作の結果に基づいて、ストーリーのテキストを追加したり削除したりする場合は、この章の以前の節で説明したテキスト参照の問題に注意する必要があります。この問題が発生する場合は、返されたテキストオブジェクトのコレクションを逆順で処理するようにループを設定するか、または、検索結果を逆順で返すように設定して、そのコレクションを通常の順序で処理します。

検索／変更の環境設定

テキストを検索する前に、検索／変更の環境設定を消去することで、前回の設定を引き継がないようにすることができます。また、必要に応じていくつかの検索／変更環境設定を指定して、対象となるテキスト、フォーマット、正規表現、字形などを指定することができます。検索／変更操作を行う一般的な手順を次に示します。

1. 検索／変更の環境設定を消去します。検索／変更操作の種類に応じて、次の 3 つのいずれかを行います。

▷
```
--find/change text preferences
tell application "Adobe InDesign CS6"
set find text preferences to nothing
set change text preferences to nothing
end tell
```

▷
```
--find/change grep preferences
tell application "Adobe InDesign CS6"
set find grep preferences to nothing
set change grep preferences to nothing
end tell
```

▷
```
--find/change glyph preferences
tell application "Adobe InDesign CS6"
set find glyph preferences to nothing
set change glpyh preferences to nothing
end tell
```

2. 検索パラメーターを設定します。
3. 検索／変更操作を実行します。
4. 検索／変更の環境設定を再び消去します。

テキストの検索と変更

指定したテキスト文字列を検索する方法を、次のスクリプトに示します。このスクリプトではドキュメント全体を検索していますが、ストーリー、テキストフレーム、段落、テキスト列などのテキストオブジェクトを検索することもできます。`findText` メソッドとそのパラメーターは、すべてのテキストオブジェクトで共通しています（完全なスクリプトについては、FindText を参照してください）。

```
--Clear the find/change preferences.
set find text preferences to nothing
set change text preferences to nothing
--Search the document for the string "Text".
set find what of find text preferences to "text"
--Set the find options.
set case sensitive of find change text options to false
set include footnotes of find change text options to false
set include hidden layers of find change text options to false
set include locked layers for find of find change text options to false
set include locked stories for find of find change text options to false
set include master pages of find change text options to false
set whole word of find change text options to false
tell active document
    set myFoundItems to find text
    display dialog ("Found " & (count myFoundItems) & " instances of the search string.")
end tell
```

指定したテキスト文字列を検索して別の文字列に置換する方法を、次のスクリプトに示します（完全なスクリプトについては、ChangeText を参照してください）。

```
--Clear the find/change preferences.
set find text preferences to nothing
set change text preferences to nothing
--Search the document for the string "copy".
set find what of find text preferences to "copy"
set change to of change text preferences to "text"
--Set the find options.
set case sensitive of find change text options to false
set include footnotes of find change text options to false
set include hidden layers of find change text options to false
set include locked layers for find of find change text options to false
set include locked stories for find of find change text options to false
set include master pages of find change text options to false
set whole word of find change text options to false
tell active document
    set myFoundItems to change text
    display dialog ("Found " & (count myFoundItems) & " instances of the search string.")
end tell
```

テキストフォーマットの検索と変更

テキストフォーマットの検索や変更を行うには、`findTextPreferences` オブジェクトや `changeTextPreferences` オブジェクトのその他のプロパティを設定します。次のスクリプトにその例を示します（チュートリアルスクリプトの FindChangeFormatting より）。

```
--Clear the find/change preferences.
set find text preferences to nothing
set change text preferences to nothing
--Set the find options.
set case sensitive of find change text options to false
set include footnotes of find change text options to false
set include hidden layers of find change text options to false
set include locked layers for find of find change text options to false
set include locked stories for find of find change text options to false
set include master pages of find change text options to false
set whole word of find change text options to false
set point size of find text preferences to 24
--The following line will only work if your default font has a font style named "Bold"
--if not, change the text to a font style used by your default font.
set point size of change text preferences to 48
--Search the document. In this example, we'll use the
--InDesign search metacharacter "^9" to find any digit.
tell document 1
    set myFoundItems to change text
end tell
display dialog ("Changed " & (count myFoundItems) & " instances of the search string.")
--Clear the find/change preferences after the search.
set find text preferences to nothing
set change text preferences to nothing
```

grep の使用

InDesign の `findGrep` メソッドや `changeGrep` メソッドを使用すれば、正規表現を利用した検索／変更操作が行えます。正規表現を利用した検索／変更操作では、指定したフォーマットのテキストを検索したり、テキストのフォーマットを `changeGrepPreferences` オブジェクトのプロパティで指定したフォーマットに置換することができます。これらのメソッドや関係する環境設定オブジェクトの使用方法を、次のスクリプトに示します（完全なスクリプトについては、FindGrep を参照してください）。

```
--Clear the find/change preferences.
set find grep preferences to nothing
set change grep preferences to nothing
--Set the find options.
set include footnotes of find change grep options to false
set include hidden layers of find change grep options to false
set include locked layers for find of find change grep options to false
set include locked stories for find of find change grep options to false
set include master pages of find change grep options to false
--Regular expression for finding an email address.
set find what of find grep preferences to "(?i)[A-Z0-9]*?@[A-Z0-9]*?[.]..."
--Apply the change to 24-point text only.
set point size of find grep preferences to 24
set underline of change grep preferences to true
tell myDocument
    change grep
end tell
--Clear the find/change preferences after the search.
set find grep preferences to nothing
set change grep preferences to nothing
```

注意 : `findChangeTextOptions` オブジェクトで使用できる `wholeWord` と `caseSensitive` の 2 つのプロパティは、`findChangeGrepOptions` オブジェクトでは使用できません。これは、正規表現の文字列で同等のオプションを表現できるからです。大文字と小文字を区別しない場合は `(?i)` を使用し、区別する場合は `(?-i)` を使用します。単語の先頭に一致するには `¥>` を使用し、単語の末尾に一致するには `¥<` を使用し、単語の境界に一致するには `¥b` を使用します。

grep 検索／変更の利用例として、マークアップテキスト（タグでフォーマットが指定されているプレーンテキスト）を InDesign 形式のテキストに変換する処理があります。簡単なテキストマークアップスキームとしては、PageMaker の段落タグがあります（これは、PageMaker のタグ付きテキスト形式のファイルとは異な

ります）。このスキームでマークアップされたテキストファイルでは、次のように、段落スタイルの名前が段落の先頭に示されます。

```
<heading1>This is a heading.
<body_text>This is body text.
```

grep 検索とテキスト検索／変更を組み合わせれば、テキストにフォーマットを適用して、マークアップタグを削除することができます。次のスクリプトにその例を示します（チュートリアルスクリプトの ReadPMTags より）。

```
set myDocument to document 1
set myStory to story 1 of myDocument
myReadPMTags(myStory)
```

このスクリプトで参照されている myReadPMTags ハンドラーを次に示します。

```
on myReadPMTags(myStory)
    local myFoundTags, myFoundItems, myFoundTag, myString, myStyleName, myStyle
    tell application "Adobe InDesign CS6"
        set myDocument to parent of myStory
        --Reset the find grep preferences to ensure that
        --previous settings do not affect the search.
        set find grep preferences to nothing
        set change grep preferences to nothing
        --Set the find options.
        set include footnotes of find change grep options to false
        set include hidden layers of find change grep options to false
        set include locked layers for find of find change grep options to false
        set include locked stories for find of find change grep options to false
        set include master pages of find change grep options to false
        --Find the tags.
        set find what of find grep preferences to "(?i)^<¥¥s*¥¥w+¥¥s*>"
        tell myStory
            set myFoundItems to find grep
        end tell
        if (count myFoundItems) is not equal to 0 then
            set myFoundTags to {}
            repeat with myCounter from 1 to (count myFoundItems)
                set myFoundTag to contents of item myCounter of myFoundItems
                if myFoundTags does not contain myFoundTag then
                    copy myFoundTag to end of myFoundTags
                end if
            end repeat
            --At this point, we have a list of tags to search for.
            repeat with myCounter from 1 to (count myFoundTags)
                set myString to item myCounter of myFoundTags
                --Find the tag using find what.
                set find what of find text preferences to myString
                --Extract the style name from the tag.
                set myStyleName to text 2 through ((count characters of myString) - 1)
                of myString
                tell myDocument
                    --Create the style if it does not already exist.
                    try
                        set myStyle to paragraph style myStyleName
                    on error
                        set myStyle to make paragraph style with properties
                        {name:myStyleName}
                    end try
                end tell
                --Apply the style to each instance of the tag.
                set applied paragraph style of change text preferences to myStyle
                tell myStory
                    change text
                end tell
                --Reset the change text preferences.
```

```
                set change text preferences to nothing
                --Set the change to property to an empty string.
                set change to of change text preferences to ""
                --Search to remove the tags.
                tell myStory
                    change text
                end tell
                --Reset the find/change preferences again.
                set change text preferences to nothing
            end repeat
        end if
        --Reset the findGrepPreferences.
        set find grep preferences to nothing
    end tell
end myReadPMTags
```

字形検索の使用

特定のフォントに含まれている個々の文字を検索／変更するには、`findGlyph` メソッドや `changeGlyph` メソッドを使用します。パラメーターは `findGlyphPreferences` オブジェクトや `changeGlyphPreferences` オブジェクトで設定します。次のスクリプトでは、サンプルドキュメントの中で字形を検索し、変更しています（完全なスクリプトについては、FindChangeGlyph を参照してください）。

```
--Clear glyph search preferences.
set find glyph preferences to nothing
set change glyph preferences to nothing
set myDocument to document 1
--You must provide a font that is used in the document for the
--applied font property of the find glyph preferences object.
set applied font of find glyph preferences to applied font of character 1 of story 1 of myDocument
--Provide the glyph ID, not the glyph Unicode value.
set glyph ID of find glyph preferences to 374
--The applied font of the change glyph preferences object can be
--any font available to the application.
set applied font of change glyph preferences to "ITC Zapf DingbatsMedium"
set glyph ID of change glyph preferences to 85
tell myDocument
    change glyph
end tell
--Clear glyph search preferences.
set find glyph preferences to nothing
set change glyph preferences to nothing
```

表の操作

`convertTextToTable` メソッドを使用して、既存のテキストから表を作成したり、ストーリー内の任意の挿入点に空の表を作成することができます。次のスクリプトでは、3 つの異なる方法で表を作成しています（完全なスクリプトについては、MakeTable を参照してください）。

```
--Given a document "myDocument" containing a story...
set myStory to story 1 of myDocument
tell myStory
    set myStartCharacter to index of character 1 of paragraph 7
    set myEndCharacter to index of character -2 of paragraph 7
    set myText to object reference of text from character
    myStartCharacter to character myEndCharacter
    --The convertToTable method takes three parameters:
    --[column separator as string]
    --[row separator as string]
    --[number of columns as integer] (only used if the column separator
    --and row separator values are the same)
    --In the last paragraph in the story, columns are separated by commas
    --and rows are separated by semicolons, so we provide those characters
    --to the method as parameters.
    tell myText
        set myTable to convert to table column separator "," row separator ";"
    end tell
    set myStartCharacter to index of character 1 of paragraph 2
    set myEndCharacter to index of character -2 of paragraph 5
    set myText to object reference of text from character myStartCharacter
    to character myEndCharacter
    --In the second through the fifth paragraphs, colums are separated by
    --tabs and rows are separated by returns. These are the default delimiter
    --parameters, so we don't need to provide them to the method.
    tell myText
        set myTable to convert to table column separator tab row separator return
    end tell
    --You can also explicitly add a table--you don't have to convert text to a table.
    tell insertion point -1
        set myTable to make table
        set column count of myTable to 3
        set body row count of myTable to 3
    end tell
end tell
```

表セルの結合方法を、次のスクリプトに示します（完全なスクリプトについては、MergeTableCells を参照してください）。

```
--Given a document "myDocument" containing a story...
tell story 1 of myDocument
    tell table 1
        --Merge all of the cells in the first column.
        merge cell 1 of column 1 with cell -1 of column 1
        --Convert column 2 into 2 cells (rather than 4).
        merge cell 3 of column 2 with cell -1 of column 2
        merge cell 1 of column 2 with cell 2 of column 2
        --Merge the last two cells in row 1.
        merge cell -2 of row 1 with cell -1 of row 1
        --Merge the last two cells in row 3.
        merge cell -2 of row 3 with cell -1 of row 3
    end tell
end tell
```

表セルの分割方法を、次のスクリプトに示します（完全なスクリプトについては、SplitTableCells を参照してください）。

```
tell table 1 of story 1 of document 1
    split cell 1 using horizontal
    split column 1 using vertical
    split cell 1 using vertical
    split row -1 using horizontal
    split cell -1 using vertical
    --Fill the cells with row:cell labels.
    repeat with myRowCounter from 1 to (count rows)
        set myRow to row myRowCounter
        repeat with myCellCounter from 1 to (count cells of myRow)
            set myString to "Row: " & myRowCounter & " Cell: " & myCellCounter
            set contents of text 1 of cell myCellCounter of row myRowCounter to myString
        end repeat
    end repeat
end tell
```

表にヘッダー行とフッター行を作成する方法を、次のスクリプトに示します（完全なスクリプトについては、HeaderAndFooterRows を参照してください）。

```
--Given a document containing a story that contains a table...
tell table 1 of story 1 of document 1
    --Convert the first row to a header row.
    set row type of row 1 to header row
    --Convert the last row to a footer row.
    set row type of row -1 to footer row
end tell
```

表にフォーマットを適用する方法を、次のスクリプトに示します（完全なスクリプトについては、TableFormatting を参照してください）。

```
set myTable to table 1 of story 1 of myDocument
tell myTable
    --Convert the first row to a header row.
    set row type of row 1 to header row
    --Use a reference to a swatch, rather than to a color.
    set fill color of row 1 to swatch "DGC1_446b" of myDocument
    set fill tint of row 1 to 40
    set fill color of row 2 to swatch "DGC1_446a" of myDocument
    set fill tint of row 2 to 40
    set fill color of row 3 to swatch "DGC1_446a" of myDocument
    set fill tint of row 3 to 20
    set fill color of row 4 to swatch "DGC1_446a" of myDocument
    set fill tint of row 4 to 40
    tell every cell in myTable
        set top edge stroke color to swatch "DGC1_446b" of myDocument
        set top edge stroke weight to 1
        set bottom edge stroke color to swatch "DGC1_446b" of myDocument
        set bottom edge stroke weight to 1
        --When you set a cell stroke to a swatch, make certain that
        --you also set the stroke weight.
        set left edge stroke color to swatch "None" of myDocument
        set left edge stroke weight to 0
        set right edge stroke color to swatch "None" of myDocument
        set right edge stroke weight to 0
    end tell
end tell
```

表に反復行フォーマットを追加する方法を、次のスクリプトに示します（完全なスクリプトについては、AlternatingRows を参照してください）。

```
--Given a document "myDocument" containing a story that
--contains a table...
set myTable to table 1 of story 1 of myDocument
tell myTable
    --Apply alternating fills to the table.
    set alternating fills to alternating rows
    set start row fill color to swatch "DGC1_446a" of myDocument
    set start row fill tint to 60
    set end row fill color to swatch "DGC1_446b" of myDocument
    set end row fill tint to 50
end tell
```

テキストや表セルが選択されている場合の選択の処理方法を、次のスクリプトに示します。このスクリプトでは、選択状況に応じて警告を表示していますが、実際のスクリプトでは選択されているアイテムに応じて何らかの操作を行うことになります （完全なスクリプトについては、TableSelection を参照してください）。

```
--Check to see if any documents are open.
if (count documents) is not equal to 0 then
    --If the selection contains more than one item, the selection
    --is not text selected with the Type tool.
    set mySelection to selection
    if (count mySelection) is not equal to 0 then
        --Evaluate the selection based on its type.
        set myTextClasses to {insertion point, word, text style range,
        line, paragraph, text column, text, story}
        if class of item 1 of selection is in myTextClasses then
            --The object is a text object; display the text object type.
            --A practical script would do something with the selection,
            --or pass the selection on to a function.
            if class of parent of item 1 of mySelection is cell then
                display dialog ("The selection is inside a table cell")
            else
                display dialog ("The selection is not in a table")
            end if
        else if class of item 1 of selection is cell then
            display dialog ("The selection is a table cell")
        else if class of item 1 of selection is row then
            display dialog ("The selection is a table row")
        else if class of item 1 of selection is column then
            display dialog ("The selection is a table column")
        else if class of item 1 of selection is table then
            display dialog ("The selection is a table.")
        else
            display dialog ("The selection is not in a table")
        end if
    else
        display dialog ("Please select some text and try again.")
    end if
else
    display dialog ("Nothing is selected. Please select some text and try again.")
end if
```

パステキストの追加

パステキストは、長方形、楕円形、多角形、グラフィックの線、テキストフレームに追加できます。ページアイテムにパステキストを追加する方法を、次のスクリプトに示します（完全なスクリプトについては、PathText を参照してください）。

```
--Given a document "myDocument" with a rectangle on page 1...
set myRectangle to rectangle 1 of page 1 of myDocument
tell myRectangle
    set myTextPath to make text path with properties {contents:"This is path text."}
end tell
```

テキストパスを別のテキストパスやテキストフレームにリンクするには、nextTextFrame プロパティや previousTextFrame プロパティを使用します。これは、テキストフレームのリンクの場合と同じです（98 ページの「テキストフレームの操作」を参照）。

自動修正の使用

自動修正機能を使用すると、テキストが入力されたときにテキストを自動的に修正することができます。次のスクリプトにその例を示します（完全なスクリプトについては、Autocorrect を参照してください）。

```
tell auto correct preferences
    set autocorrect to true
    set auto correct capitalization errors to true
    --Add a word pair to the autocorrect list. Each auto correct table
    --is linked to a specific language.
end tell
set myAutoCorrectTable to auto correct table "English: USA"
--To safely add a word pair to the auto correct table, get the current
--word pair list, then add the new word pair to that array, and then
--set the autocorrect word pair list to the array.
set myWordPairList to {}
set myWordPairList to myWordPairList & auto correct word pair list of myAutoCorrectTable
--Add a new word pair to the array.
set myWordPairList to myWordPairList & {{"paragarph", "paragraph"}}
--Update the word pair list.
set auto correct word pair list of auto correct table "English: USA" to myWordPairList
--To clear all autocorrect word pairs in the current dictionary:
--myAutoCorrectTable.autoCorrectWordPairList to {{}}
```

脚注の追加

ストーリーに脚注を追加する方法を、次のスクリプトに示します（myGetRandom 関数を含む完全なスクリプトについては、Footnotes を参照してください）。

```
set myDocument to document 1
tell footnote options of myDocument
    set separator text to tab
    set marker positioning to superscript marker
end tell
set myStory to story 1 of myDocument
--Add four footnotes at random locations in the story.
local myStoryLength, myRandomNumber
set myStoryLength to (count words of myStory)
repeat with myCounter from 1 to 5
    set myRandomNumber to my myGetRandom(1, myStoryLength)
    tell insertion point -1 of word myRandomNumber of myStory
        set myFootnote to make footnote
    end tell
    --Note: when you create a footnote, it contains text--the footnote marker
    --and the separator text (if any). If you try to set the text of the footnote
    --by setting the footnote contents, you will delete the marker. Instead, append
    --the footnote text, as shown below.
    tell insertion point -1 of myFootnote
        set contents to "This is a footnote."
    end tell
end repeat
```

段抜きと段分割

段抜きや段分割の属性を適用して、段落のレイアウトを複数の段にまたがるようにしたり、段を分割したりすることができます。段落の段抜きおよび段分割のスタイルを設定する方法を、次のスクリプトに示します（完全なスクリプトについては、SpanColumns を参照してください）。

```
set myDocument to active document
set myPage to page 1 of myDocument
set myTextFrame to item 1 of text frames of myPage
tell myTextFrame
    set text column count of text frame preferences to 3
    set myStory to parent story
    tell item 1 of paragraphs of myStory
        --split column
        set span column type to split columns
        set span split column count to 2
        set split column outside gutter to 0
        set split column inside gutter to 1
    end tell
    set mySpanIndex to (count of paragraphs of myStory) div 2 + 1
    tell item mySpanIndex of paragraphs of myStory
        --span columns
        set span column type to span columns
        set span split column count to all
    end tell
end tell
```

テキスト環境設定の指定

一般的なテキスト環境設定の指定方法を、次のスクリプトに示します（完全なスクリプトについては、TextPreferences を参照してください）。

```
tell text preferences
    set abut text to text wrap to true
    --baseline shift key increment can range from .001 to 200 points.
    set baseline shift key increment to 1
    set highlight custom spacing to false
    set highlight hj violations to true
    set highlight keeps to true
    set highlight substituted fonts to true
    set highlight substituted glyphs to true
    set justify text wraps to true
    --kerning key increment value is 1/1000 of an em.
    set kerning key increment to 10
    --leading key increment value can range from .001 to 200 points.
    set leading key increment to 1
    set link text files when importing to false
    set show invisibles to true
    set small cap to 60
    set subscript position to 30
```

```
    set subscript size to 60
    set superscript position to 30
    set superscript size to 60
    set typographers quotes to false
    set use optical size to false
    set use paragraph leading to false
    set z order text wrap to false
end tell
--Text editing preferences are application-wide.
tell text editing preferences
    set allow drag and drop text in story to true
    set drag and drop text in layout to true
    set smart cut and paste to true
    set triple click selects line to false
end tell
```

リンクストーリーの操作

リンクストーリーを使用すると、1 つのドキュメントに含まれる同一のストーリーまたはテキストコンテンツの複数のバージョンを簡単に管理することができます。これにより、例えば縦組みと横組みの両方のレイアウト用のデザインが必要になるといった、新たなデジタルパブリッシングワークフローへの対応が容易になります。リンクストーリーは、一般的なリンクと同じように動作します。ストーリーを親として指定し、同じストーリーをドキュメント内の別の場所に子ストーリーとして配置できます。親ストーリーを更新するとリンクパネル内で子ストーリーにフラグが付くので、子ストーリーを更新して親ストーリーに同期させることができます。

リンクストーリーの作成

リンクストーリーは複数の方法で作成できます。既存のテキストフレームを親ストーリーにリンクする方法と、指定した場所にテキストフレームを作成して親ストーリーにリンクするよう page オブジェクト、spread オブジェクト、master spread オブジェクトまたは document オブジェクトに対して指示する方法があります。

既存のテキストフレームを親ストーリーにリンクするには、次のスクリプトを使用します（完全なスクリプトについては、CreateLinkedStories を参照してください）。

```
place and link childTextFrame1 parent story parentStory
```

親ストーリーのリンクストーリーの作成を page オブジェクトに対して指示するために、place and link メソッドには 2 つのパラメーターが用意されており、これらのパラメーターでリンクストーリーを配置するレイヤーと位置を指定できます。次のスクリプトにその例を示します（完全なスクリプトについては、CreateLinkedStories を参照してください）。

```
tell myDocument
    set myNewPage to make page
    tell myNewPage
        place and link parent story parentStory place point {originX, originY} without showing
options
    end tell
end tell
```

親ストーリーのリンクストーリーの作成を spread オブジェクトに対して指示する場合、link メソッドでは Page オブジェクトの link メソッドと同じパラメーターを取ります。次のスクリプトにその例を示します（完全なスクリプトについては、CreateLinkedStories を参照してください）。

```
tell myDocument
    set myNewSpread to make spread
    tell myNewSpread
    place and link parent story parentStory place point {originX, originY} without showing options
    end tell
end tell
```

また、link メソッドを document オブジェクトで呼び出すことも可能です。このメソッドは親ストーリーを唯一のパラメーターとして取ります。このメソッドではリンクストーリーは作成されませんが、プレースガンが読み込まれるので、ユーザーがリンクストーリーの場所を自分で決めることができます。

リンクストーリーのオプションの変更

リンクストーリーのオプションを変更するには、次のスクリプトを使用します（完全なスクリプトについては、CreateLinkedStories を参照してください）。

```
tell linked story options of myDocument
    set remove forced line breaks to true
    set update while saving to true
    set warn on update of edited story to true
end tell
```

よくある質問

特定のリンクがリンクストーリーのリンクであるかどうかは、どのように判断することができますか？

file path of myLink を使用します。これが空の場合、そのリンクはリンクストーリーのリンクです。

リンクストーリーのリンクでは、リンクソースをどのような方法で取得できますか？

次のスクリプトを使用して、リンクソースを選択します。

```
tell myLink to edit original
```

次に、次のスクリプトでアクティブな選択を検出します。これがリンクソースです。

```
set mySelection to selection of application "Adobe InDesign CS6"
```

7 ユーザーインターフェイス

章の更新ステータス	
CS6	変更なし

AppleScript で作成できるダイアログボックスは、「はい」か「いいえ」で答えられる単純な質問をするためのダイアログボックスか、テキストを入力するためのダイアログボックスに限られていますが、より複雑なダイアログボックスを使用したいこともあります。InDesign スクリプティングを使用すれば、ポップアップリスト、テキスト入力フィールド、数値入力フィールドなどの一般的なユーザーインターフェイスコントロールを備えたダイアログボックスを作成できます。ユーザー（または自分）から情報を受け取って、それに基づいて処理を実行したい場合は、`dialog` オブジェクトが役に立ちます。

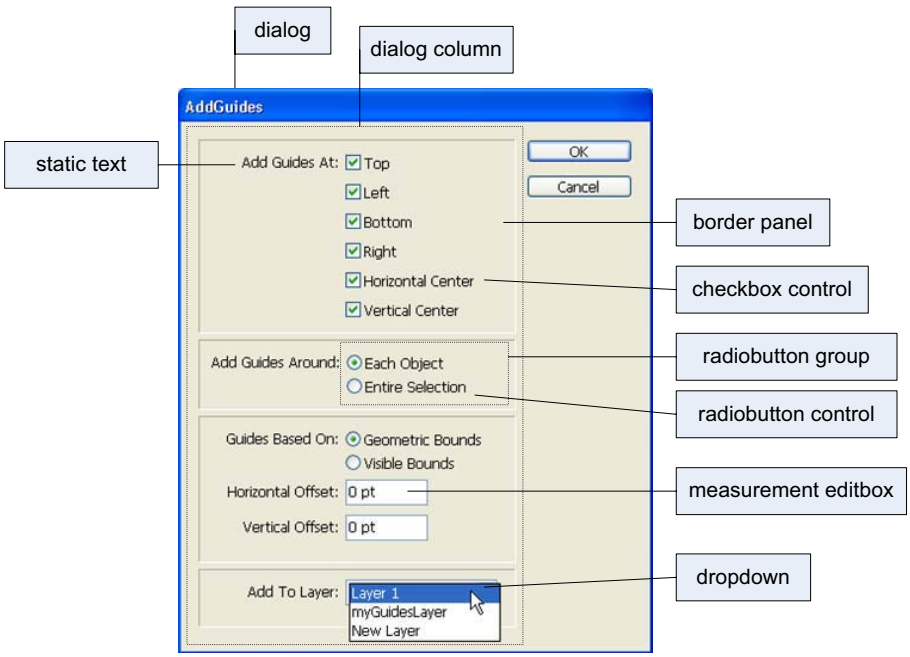
この章では、InDesign のダイアログボックスを使用したスクリプトについて説明します。最初は簡単なサンプルスクリプトから始めて、徐々に複雑なスクリプトへと進んでいきます。

注意 : JavaScript で InDesign スクリプトを作成する場合は、Adobe ScriptUI コンポーネントを使用してユーザーインターフェイスを作成することもできます。この章には、ScriptUI スクリプティングのチュートリアルも含まれています。詳しくは、『Adobe JavaScript Tools Guide』を参照してください。

ここでは、読者が『Adobe InDesign スクリプティングチュートリアル』を既に読んでおり、スクリプトを作成して実行する方法を理解しているものとして説明を行います。

ダイアログボックスの概要

InDesign のダイアログボックスは、InDesign の他のスクリプティングオブジェクトと同じく、1 つのオブジェクトです。次の図に示すように、ダイアログボックスには様々な要素（「ウィジェット」と呼ばれます）を含めることができます。この図に示されている各要素について、この図の次の表で説明します。



ダイアログボックスの要素	InDesign での名前
テキスト編集フィールド	Text editbox control
数値入力フィールド	Real editbox, integer editbox, measurement editbox, percent editbox, angle editbox
ポップアップメニュー	Drop-down control
テキスト編集フィールドとポップアップメニューを組み合わせたコントロール	Combo-box control
チェックボックス	Check-box control
ラジオボタン	Radio-button control

これらのコントロールは、dialog オブジェクトに直接格納されるのではなく、dialog column というオブジェクトに格納されます。dialog column を使用することで、ダイアログボックス内のコントロールの位置を制御することができます。dialog column の中に、別の dialog column や border panel を含めて、ダイアログボックスをさらに細かく分割することができます（必要に応じて、dialog column や border panel の中に別の dialog column や border panel を順次含めていくことができます）。

InDesign の他のスクリプティングオブジェクトと同じように、ダイアログボックスの各パーツにも固有のプロパティがあります。例えば、checkbox control には、表示されるテキストを表すプロパティ（static label）や、コントロールの状態を表すプロパティ（checked state）があります。また、dropdown コントロールには、メニューに表示されるオプションのリストを表すプロパティ（string list）があります。

スクリプトの中でダイアログボックスを使用するには、まず dialog オブジェクトを作成し、そこに様々なコントロールを配置して、ダイアログボックスを表示します。そして、スクリプトの処理に必要な値をそれらのコントロールから収集します。ダイアログボックスは、破棄されるまで InDesign のメモリ内に維持されます。したがって、ダイアログボックスをメモリ内に保持しておき、そのプロパティに保存されているデータを複数のスクリプトで使用することが可能です。ただし、ダイアログボックスを保持するためにメモリが消費されるので、必要がなくなったら破棄するようにしてください。一般的には、スクリプトが処理を終了する前にダイアログボックスオブジェクトを破棄します。

最初の InDesign ダイアログボックスの作成

InDesign ダイアログボックスは、簡単な手順で作成できます。ダイアログボックスを追加し、ダイアログボックスにダイアログコラムを追加し、ダイアログコラムにコントロールを追加します。次のスクリプトにその例を示します（完全なスクリプトについては、SimpleDialog を参照してください）。

```
set myDialog to make dialog with properties {name:"Simple Dialog"}
tell myDialog
    tell (make dialog column)
        make static text with properties {static label:"This is a very
        simple dialog box."}
    end tell
end tell
--Show the dialog box.
set myResult to show myDialog
--If the user clicked OK, display one message;
--if they clicked Cancel, display a different message.
if myResult is true then
    display dialog ("You clicked the OK button")
else
    display dialog ("You clicked the Cancel button")
end if
--Remove the dialog box from memory.
destroy myDialog
```

「Hello World」へのユーザーインターフェイスの追加

次の例は、『Adobe InDesign スクリプティングチュートリアル』で作成した Hello World チュートリアルスクリプトに、簡単なユーザーインターフェイスを追加したものです。ダイアログボックスに表示されるオプションを使用して、表示するテキストを指定したり、テキストのポイントサイズを変更したりすることができます。

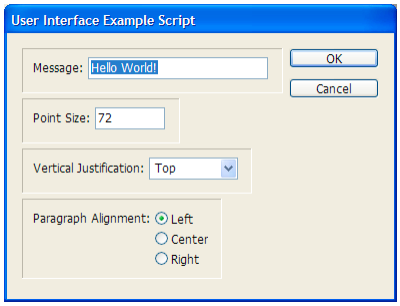
```
set myDialog to make dialog
tell myDialog
    set name to "Simple User Interface Example Script"
    set myDialogColumn to make dialog column
    tell myDialogColumn
        --Create a text entry field.
        set myTextEditField to make text editbox with properties
        {edit contents:"Hello World!", min width:180}
        --Create a number (real) entry field
        set myPointSizeField to make measurement editbox with properties
        {edit value:72, edit units:points}
    end tell
end tell
set myResult to show myDialog
if myResult is true then
    --Get the settings from the dialog box.
    --Get the point size from the point size field.
    set myPointSize to edit value of myPointSizeField
    --Get the example text from the text edit field.
    set myString to edit contents of myTextEditField
    --Remove the dialog box from memory.
    destroy myDialog
    my myMakeDocument(myPointSize, myString)
else
    destroy myDialog
end if
```

このスクリプトで参照されている `myMakeDocument` ハンドラーを次に示します。

```
on myMakeDocument(myPointSize, myString)
    tell application "Adobe InDesign CS6"
        set myDocument to make document
        tell view preferences of myDocument
        end tell
        tell page 1 of myDocument
            --Create a text frame.
            set myTextFrame to make text frame
            set geometric bounds of myTextFrame to my myGetBounds
            (myDocument, page 1 of myDocument)
            --Apply the settings from the dialog box to the text frame.
            set contents of myTextFrame to myString
            --Set the point size of the text in the text frame.
            set point size of text 1 of myTextFrame to myPointSize
        end tell
    end tell
end myMakeDocument
```

さらに複雑なユーザーインターフェイスの作成

次の例では、様々なコントロールをサンプルダイアログボックスに追加します。この例では、次のようなダイアログボックスが作成されます。



完全なスクリプトについては、ComplexUI を参照してください。

```
set myDialog to make dialog
--This example dialog box uses border panels and dialog columns to
--separate and organize the user interface items in the dialog.
    tell myDialog
        set name to "User Interface Example Script"
        set myDialogColumn to make dialog column
        tell myDialogColumn
            set myBorderPanel to make border panel
            tell myBorderPanel
                set myDialogColumn to make dialog column
                tell myDialogColumn
                    make static text with properties {static label:"Message:"}
                end tell
                set myDialogColumn to make dialog column
                tell myDialogColumn
                    set myTextEditField to make text editbox with properties
                    {edit contents:"Hello World!", min width:180}
                end tell
            end tell
            set myBorderPanel to make border panel
            tell myBorderPanel
                set myDialogColumn to make dialog column
                tell myDialogColumn
                    make static text with properties {static label:"Point Size:"}
                end tell
                set myDialogColumn to make dialog column
                tell myDialogColumn
                    set myPointSizeField to make measurement editbox with
                    properties {edit value:72, edit units:points}
                end tell
            end tell
            set myBorderPanel to make border panel
            tell myBorderPanel
                set myDialogColumn to make dialog column
                tell myDialogColumn
                    make static text with properties
                    {static label:"Vertical Justification:"}
                end tell
                set myDialogColumn to make dialog column
                tell myDialogColumn
                    set myVerticalJustificationMenu to make dropdown with properties
                    {string list:{"Top", "Center", "Bottom"}, selected index:0}
                end tell
            end tell
            set myBorderPanel to make border panel
```

```
            tell myBorderPanel
                make static text with properties {static label:"Paragraph Alignment:"}
                set myParagraphAlignmentGroup to make radiobutton group
                tell myParagraphAlignmentGroup
                    set myLeftRadioButton to make radiobutton control with
                    properties {static label:"Left", checked state:true}
                    set myCenterRadioButton to make radiobutton control with
                    properties {static label:"Center"}
                    set myRightRadioButton to make radiobutton control with
                    properties {static label:"Right"}
                end tell
            end tell
        end tell
    end tell
    set myResult to show myDialog
    if myResult is true then
        --Get the settings from the dialog box.
        --Get the point size from the point size field.
        set myPointSize to edit value of myPointSizeField
        --Get the example text from the text edit field.
        set myString to edit contents of myTextEditField
        --Get the vertical justification setting from the pop-up menu.
        if selected index of myVerticalJustificationMenu is 0 then
            set myVerticalJustification to top align
        else if selected index of myVerticalJustificationMenu is 1 then
            set myVerticalJustification to center align
        else
            set myVerticalJustification to bottom align
        end if
        --Get the paragraph alignment setting from the radiobutton group.
        get properties of myParagraphAlignmentGroup


        if selected button of myParagraphAlignmentGroup is 0 then
            set myParagraphAlignment to left align
        else if selected button of myParagraphAlignmentGroup is 1 then
            set myParagraphAlignment to center align
        else
            set myParagraphAlignment to right align
        end if
        --Remove the dialog box from memory.
        destroy myDialog
        my myMakeDocument(myPointSize, myString,
        myParagraphAlignment, myVerticalJustification)
    else
        destroy myDialog
    end if
end tell
```

このスクリプトで参照されている `myMakeDocument` ハンドラーを次に示します。

```
on myMakeDocument(myPointSize, myString, myParagraphAlignment, myVerticalJustification)
    tell application "Adobe InDesign CS6"
        set myDocument to make document
        tell view preferences of myDocument
        end tell
        set myPage to page 1 of myDocument
        tell myPage
            set myTextFrame to make text frame
            set geometric bounds of myTextFrame to my myGetBounds(myDocument, myPage)
            --Apply the settings from the dialog box to the text frame.
            set contents of myTextFrame to myString
            --Apply the vertical justification setting.
            set vertical justification of text frame preferences of
            myTextFrame to myVerticalJustification
            --Apply the paragraph alignment ("justification").
            --"text 1 of myTextFrame" is all of the text in the text frame.
            set justification of text 1 of myTextFrame to myParagraphAlignment
            --Set the point size of the text in the text frame.
            set point size of text 1 of myTextFrame to myPointSize
        end tell
    end tell
end myMakeDocument
```

ScriptUI の使用

JavaScript では、ScriptUI というアドビ システムズ社独自のスクリプティングコンポーネントを使用して、ユーザーインターフェイス要素の作成や定義が行えます。ScriptUI を使用すれば、InDesign の組み込みの `dialog` オブジェクトよりも格段に複雑なインタラクティブダイアログボックスを作成できます。また、フローティングパレットやプログレスバーも作成できます。

AppleScript や VBScript でも、ScriptUI を使用して定義されたユーザーインターフェイス要素にアクセスすることができます。do script メソッド（AppleScript の場合）や DoScript メソッド（VBScript の場合）を使用すれば、他のスクリプト言語で書かれたスクリプトを実行できます。

ScriptUI を使用したプログレスバーの作成

JavaScript で ScriptUI を使用してプログレスバーを作成する方法を、次のスクリプトに示します（完全なスクリプトについては、ProgressBar を参照してください）。このスクリプトで作成したプログレスバーは、AppleScript で使用することが可能です。

```
#targetengine "session"
//Because these terms are defined in the "session" engine,
//they will be available to any other JavaScript running
//in that instance of the engine.
var myMaximumValue = 300;
var myProgressBarWidth = 300;
var myIncrement = myMaximumValue/myProgressBarWidth;
myCreateProgressPanel(myMaximumValue, myProgressBarWidth);
function myCreateProgressPanel(myMaximumValue, myProgressBarWidth){
    myProgressPanel = new Window('window', 'Progress');
    with(myProgressPanel){
        myProgressPanel.myProgressBar = add('progressbar', [12, 12,
    myProgressBarWidth, 24], 0, myMaximumValue);
    }
}
```

前のスクリプトで作成したプログレスバーを AppleScript で呼び出す方法を、次のスクリプトに示します（完全なスクリプトについては、CallProgressBar を参照してください）。

```
set myDocument to make document
--Add pages to the active document.
--If you don't do this, the progress bar will
--go by too quickly.
--Note that the JavaScripts must use the "session"
--engine for this to work.
set myJavaScript to "#targetengine ¥"session¥"" & return
set myJavaScript to myJavaScript & "myCreateProgressPanel(100, 400);" & return
set myJavaScript to myJavaScript & "myProgressPanel.show();" & return
do script myJavaScript language javascript
repeat with myCounter from 1 to 40
    set myJavaScript to "#targetengine ¥"session¥"" & return
    set myJavaScript to myJavaScript & "myProgressPanel.myProgressBar.value = "
    set myJavaScript to myJavaScript & myCounter & "/myIncrement;" & return
    do script myJavaScript language javascript
    tell myDocument to make page
    if myCounter = 100 then
        set myJavaScript to "#targetengine ¥"session¥"" & return
        set myJavaScript to myJavaScript &
        "myProgressPanel.myProgressBar.value = 0;" & return
        set myJavaScript to myJavaScript & "myProgressPanel.hide();" & return
        do script myJavaScript language javascript
        close myDocument saving no
    end if
end repeat
```

8 イベント

章の更新ステータス	
CS6	変更なし

InDesign スクリプティングでは、アプリケーションやドキュメントの一般的なイベント（ファイルのオープン、新規ファイルの作成、プリント、ディスク上のテキストファイルやグラフィックファイルの読み込みなど）に応答することができます。アプリケーションで発生するイベントは、`event` オブジェクトによって表されます。イベントにスクリプトを割り当てるには、`event listener` というスクリプトオブジェクトを使用します。イベントに割り当てるスクリプトは、通常のスクリプトとほとんど同じですが、通常のスクリプトはユーザーが（スクリプトパネルから）実行するのに対して、イベントに割り当てるスクリプトは自動的に（目的のイベントが発生したときに）実行される点が異なります。

この章では、InDesign のイベントを使用したスクリプトについて説明します。最初は簡単なサンプルスクリプトから始めて、徐々に複雑なスクリプトへと進んでいきます。

ここでは、読者が『Adobe InDesign スクリプティングチュートリアル』を既に読んでおり、スクリプトを作成、インストール、実行する方法を理解しているものとして説明を行います。

メニューに関連するイベントについては、第 9 章「メニュー」を参照してください。

InDesign のイベントスクリプトモデルは、Worldwide Web Consortium（W3C）勧告のドキュメントオブジェクトモデルイベントに似ています。詳しくは、http://www.w3c.org を参照してください。

イベントスクリプトモデルについて

InDesign のイベントスクリプトモデルは、アプリケーションで発生する各種のイベントに相当する一連のオブジェクトによって構成されています。その中心となるのが `event` オブジェクトです。これは、InDesign のユーザーインターフェイスで行われる特定のアクション（またはスクリプトの処理に相当するアクション）を表します。

イベントに応答するには、そのイベントを受け取るオブジェクトに `event listener` を登録します。そのオブジェクトが指定のイベントを受け取ると、`event listener` のハンドラー関数に定義されているスクリプト関数（ディスク上のスクリプトファイルへの参照）が実行されます。

使用可能なイベントを確認するには、AppleScript 用語説明ビューアを使用します。AppleScript エディタの AppleScript 用語説明ビューアで、Basics Suite のイベントクラスを確認します。

イベントプロパティとイベントの伝達について

ユーザーやスクリプトによってアクションが実行されてイベントが発生すると、そのイベントに応答可能なスクリプティングオブジェクトにイベントが伝達されます。特定のイベントに対する `event listener` が登録されているオブジェクトにそのイベントが伝達されると、その `event listener` が呼び出されます。1 つのイベントが複数のオブジェクトに伝達されて、複数のオブジェクトで処理が行われる場合もあります。

イベントの伝達には、次の 2 種類があります。

- **伝達しない** — イベントターゲットに登録されている `event listener` のみが呼び出されます。伝達しないイベントの例としては、`beforeDisplay` イベントがあります。

- **バブル** — target から伝達が開始されます。まず、target に登録されている応答可能な event listener が呼び出されます。次に、スクリプトオブジェクトモデルの上方向にイベントが伝達されます。スクリプトオブジェクトモデルで target の上位にあるオブジェクトに登録されている、そのイベントに応答可能な event listener が呼び出されます。

次の表に、event のプロパティの詳細と、スクリプトオブジェクトモデルでのイベント伝達に与える影響を示します。

プロパティ	説明
Bubbles	true の場合、event が発生したオブジェクトの上位にあるスクリプティングオブジェクトにその event が伝達されます。
Cancelable	true の場合、target における event のデフォルトの動作をキャンセルできます。これを行うには、prevent default コマンドを使用します。
CurrentTarget	event を処理している現在のスクリプティングオブジェクト。この表の Target を参照してください。
DefaultPrevented	true の場合、現在の target における event のデフォルト動作が実行されず、アクションがキャンセルされます。この表の Target を参照してください。
EventPhase	event の伝達プロセスの現在のステージ。
EventType	event のタイプを表す文字列（「beforeNew」など）。
PropagationStopped	true の場合、current target 以降への伝達が中止されます（この表の Target を参照してください）。イベントの伝達を中止するには、stop propagation コマンドを使用します。
Target	event を発生させたオブジェクト。例えば、beforeImport イベントの Target はドキュメントであり、beforeNew イベントの Target はアプリケーションです。
TimeStamp	event が発生した日時。

イベントリスナーの操作

event listener を作成するときには、イベントのタイプとイベントハンドラー（ハンドラーまたはファイル参照）を指定します。特定のイベントに対して event listener を追加する方法を、次のスクリプトに示します（完全なスクリプトについては、AddEventListener を参照してください）。

```
--Registers an event listener on the afterNew event.
tell application "Adobe InDesign CS6"
    make event listener with properties {event type:"afterNew",
    handler:my myDisplayEventType}
end tell
```

このスクリプトで参照されているハンドラーを次に示します。

```
on myDisplayEventType()
    tell application "Adobe InDesign CS6"
        --"evt" is the event passed to this script by the event listener.
        set myEvent to evt
        display dialog ("This event is the "& event type of myEvent & "event.")
    end tell
end myMessage
```

別のスクリプトファイルで定義されているイベントハンドラーではなく、スクリプトの中で定義されているイベントハンドラーを使用する場合、InDesign のスクリプトパネルから実行するとスクリプトは正常に動作しますが、スクリプト編集アプリケーション（Apple 社のスクリプトエディタなど）から実行すると動作しません。

このスクリプトで作成した event listener を削除するには、次のスクリプトを実行します（チュートリアルスクリプトの RemoveEventListener より）。

```
tell application "Adobe InDesign CS6"
    remove event listener event type "afterNew" handler myDisplayEventType
end tell
```

1 つの event listener でイベントが処理されても、そのイベントを監視している他の event listener があれば、そこでも処理が行われる場合があります（イベントの伝達方法によって異なります）。例えば、afterOpen イベントは、アプリケーションとドキュメントの両方の event listener で監視されている可能性があります。

event listener は、現在の InDesign セッションを超えて保持されることはありません。すべての InDesign セッションで event listener を利用できるようにしたい場合は、スタートアップスクリプト用のフォルダーである Startup Scripts にスクリプトを追加します （スクリプトのインストールについて詳しくは、『Adobe InDesign スクリプティングチュートリアル』の「スクリプトのインストール」を参照してください）。ドキュメントに event listener スクリプトを追加しても、ドキュメントとともに保存されたり、IDML に書き出されたりすることはありません。

注意 : event listener を定義するスクリプトで問題が発生した場合は、その event listener を削除するスクリプトを実行するか、InDesign を終了して再起動します。

スクリプトオブジェクトモデルの中で 1 つのイベントが伝達されて、複数の event listener が呼び出される場合があります。次のサンプルスクリプトでは、様々なオブジェクトに登録されている event listener が 1 つのイベントによって呼び出されます（完全なスクリプトについては、MultipleEventListeners を参照してください）。

```
--Shows that an event can trigger multiple event listeners.
tell application "Adobe InDesign CS6"
    set myDocument to make document
    --You'll have to fill in a valid file path for your system
    make event listener with properties {event type:"beforeImport",
    handler:my myEventInfo}
    tell myDocument
        make event listener with properties {event type:"beforeImport",
        handler:my myEventInfo}
    end tell
end tell
```

このスクリプトで参照されているハンドラーを次に示します。

```
on myEventInfo(myEvent)
    tell application "Adobe InDesign CS6"
        set myString to "Current Target: " & name of current target of myEvent
        display dialog (myString)
    end tell
end myEventInfo
```

このスクリプトを実行してファイルを配置すると、ドキュメントの名前を示す警告と、アプリケーションの名前を示す警告が順に表示されます。このスクリプトで追加したイベントリスナーを削除するには、RemoveMultipleEventListeners スクリプトを実行します。

次のサンプルスクリプトでは、ドキュメントの各イベントの event listener を作成して、簡単なダイアログボックスにイベントに関する情報を表示しています。完全なスクリプトについては、EventListenersOn を参照してください。

```
tell application "Adobe InDesign CS6"
set myEventNames to {"beforeNew", "afterNew", "beforeQuit", "afterQuit", "beforeOpen",
"afterOpen", "beforeClose", "afterClose", "beforeSave", "afterSave", "beforeSaveAs",
"afterSaveAs", "beforeSaveACopy", "afterSaveACopy", "beforeRevert", "afterRevert", "beforePrint",
"afterPrint", "beforeExport", "afterExport", "beforeImport", "afterImport", "beforePlace",
"afterPlace"}
    repeat with myEventName in myEventNames
        make event listener with properties {event type:myEventName,
        handler:"yukino:IDEventHandlers:GetEventInfo.applescript"}
    end repeat
end tell
```

このスクリプトで参照されているスクリプトを次に示します。このファイル参照で示されているディスク上の場所に、該当するスクリプトファイルが存在している必要があります。完全なスクリプトについては、`GetEventInfo.applescript` を参照してください。

```
main(evt)
on main(myEvent)
    tell application "Adobe InDesign CS6"
        set myString to "Handling Event: " & event type of myEvent & return
        set myString to myString & "Target: " & name of target of myEvent & return
        set myString to myString & "Current: " & name of current target of
        myEvent & return
        set myString to myString & "Phase: " & my myGetPhaseName(event phase o
        f myEvent) & return
        set myString to myString & "Bubbles: " & bubbles of myEvent & return
        set myString to myString & "Cancelable: " & cancelable of
        myEvent & return
        set myString to myString & "Stopped: " & propagation stopped of
        myEvent & return
        set myString to myString & "Canceled: " & default prevented of
        myEvent & return
        set myString to myString & "Time: " & time stamp of myEvent & return
        display dialog (myString)
    end tell
end main
--Function returns a string corresponding to the event phase.
on myGetPhaseName(myEventPhase)
    tell application "Adobe InDesign CS6"
        if myEventPhase is at target then
            set myString to "At Target"
        else if myEventPhase is bubbling phase then
            set myString to "Bubbling"
        else if myEventPhase is done then
            set myString to "Done"
        else if myEventPhase is not dispatching then
            set myString to "Not Dispatching"
        else
            set myString to "Unknown Phase"
        end if
        return myString
    end tell
end myGetPhaseName
```

アプリケーションオブジェクトのすべての `event listener` をオフにする方法を、次のサンプルスクリプトに示します。完全なスクリプトについては、EventListenersOff を参照してください。

```
-tell application "Adobe InDesign CS6"
    tell event listeners to delete
end tell
```

「afterNew」イベントリスナーの例

afterNew イベントは、ユーザー名、ドキュメントの作成日、著作権情報、その他の作業に関する情報などをドキュメントに追加したい場合に便利です。ドキュメントの最初のマスタースプレッドの印刷可能領域にあるテキストフレームにこのような情報を追加する方法を、次のチュートリアルスクリプトに示します（完全なスクリプトについては、AfterNew というチュートリアルを参照してください）。このスクリプトでは、ドキュメントのメタデータ（ファイル情報または XMP 情報とも呼ばれます）も追加しています。

```
--Registers an event listener on the afterNew event.
tell application "Adobe InDesign CS6"
    make event listener with properties {event type:"afterNew",
handler:"yukino:IDEventHandlers:AfterNewHandler.applescript"}
end tell
```

このスクリプトで参照されているスクリプトを次に示します。このファイル参照で示されているディスク上の場所に、該当するスクリプトファイルが存在している必要があります。完全なスクリプトについては、AfterNewHandler.applescript を参照してください。

```
--AfterNewHandler.applescript
--An InDesign CS6 AppleScript
--
--This script is called by the AfterNew.applescript. It
--Sets up a basic document layout and adds XMP information
--to the document.
main(evt)
on main(myEvent)
    tell application "Adobe InDesign CS6"
        if user name = "" then
            set user name to "Adobe"
        end if
        set myUserName to user name
        --set myDocument to parent of myEvent
        set myDocument to document 1
        tell view preferences of myDocument
            set horizontal measurement units to points
            set vertical measurement units to points
            set ruler origin to page origin
        end tell
        --MySlugOffset is the distance from the bottom of the page
        --to the top of the slug.
        set mySlugOffset to 12
        --MySlugHeight is the height of the text frame
        --containing the job information.
        set mySlugHeight to 72
        tell document preferences of myDocument
            set documentSlugUniformSize to false
            set slug bottom offset to mySlugOffset + mySlugHeight
            set slug top offset to 0
            set slug inside or left offset to 0
            set slug right or outside offset to 0
        end tell
        repeat with myCounter from 1 to (count master spreads of myDocument)
            set myMasterSpread to master spread myCounter of myDocument
            repeat with myMasterPageCounter from 1 to (count pages of
            myMasterSpread)
                set myPage to page myMasterPageCounter of myMasterSpread
                set mySlugBounds to my myGetSlugBounds(myDocument, myPage,
                mySlugOffset, mySlugHeight)
                tell myPage
                    set mySlugFrame to make text frame with properties {geometric
                    bounds:mySlugBounds, contents:"Created: " &
                    time stamp of myEvent &
                    return & "by: " & myUserName}
                end tell
```

```
            end repeat
        end repeat
        tell metadata preferences of myDocument
            set author to "Adobe Systems"
            set description to "This is an example document
            containing XMP metadata. Created: " & time stamp of myEvent
        end tell
    end tell
end main
on myGetSlugBounds(myDocument, myPage, mySlugOffset, mySlugHeight)
    tell application "Adobe InDesign CS6"
        tell myDocument
            set myPageWidth to page width of document preferences
            set myPageHeight to page height of document preferences
        end tell
        set myLeft to left of margin preferences of myPage
        set myRight to right of margin preferences of myPage
        set myX1 to myLeft
        set myY1 to myPageHeight + mySlugOffset
        set myX2 to myPageWidth - myRight
        set myY2 to myY1 + mySlugHeight
        return {myY1, myX1, myY2, myX2}
    end tell
end myGetSlugBounds
```

「beforePrint」イベントリスナーの例

beforePrint イベントは、様々なプリフライトチェックを行うスクリプトを実行したい場合に便利です。プリントを行う前にドキュメントの特定の属性を確認するイベントリスナーの追加方法を、次のスクリプトに示します（完全なスクリプトについては、BeforePrint を参照してください）。

```
--Adds an event listener that performs a preflight check on
--a document before printing. If the preflight check fails,
--the script gives the user the opportunity to cancel the print job.
tell application "Adobe InDesign CS6"
    make event listener with properties {event type:"beforePrint",
handler:"yukino:IDEventHandlers:BeforePrintHandler.applescript"}
end tell
```

このスクリプトで参照されているスクリプトを次に示します。このファイル参照で示されているディスク上の場所に、該当するスクリプトファイルが存在している必要があります。完全なスクリプトについては、BeforePrintHandler.applescript を参照してください。

```
--BeforePrintHandler.applescript
--An InDesign CS6 AppleScript
--
--Peforms a preflight check on a document. Called by the
--BeforePrint.applescript event listener example.
--"evt" is the event passed to this script by the event listener.
main(evt)
on main(myEvent)
    tell application "Adobe InDesign CS6"
        --The parent of the event is the document.
        set myDocument to parent of myEvent
        if my myPreflight(myDocument) is false then
            tell myEvent
                stop propagation
                prevent default
            end tell
            display dialog ("Document did not pass preflight check.
            Please fix the problems and try again.")
        else
            display dialog ("Document passed preflight check. Ready to print.")
        end if
```

```
    end tell
end main
on myPreflight(myDocument)
    set myPreflightCheck to true
    set myFontCheck to my myCheckFonts(myDocument)
    set myGraphicsCheck to my myCheckGraphics(myDocument)
    display dialog ("Fonts: " & myFontCheck & return & "Links:" & myGraphicsCheck)
    if myFontCheck = false or myGraphicsCheck = false then
        set myPreflightCheck to false
        return myPreflightCheck
    end if
end myPreflight
on myCheckFonts(myDocument)
    tell application "Adobe InDesign CS6"
        set myFontCheck to true
        repeat with myCounter from 1 to (count fonts of myDocument)
            set myFont to font myCounter of myDocument
            if font status of myFont is not installed then
                set myFontCheck to false
                exit repeat
            end if
        end repeat
        return myFontCheck
    end tell
end myCheckFonts
on myCheckGraphics(myDocument)
    tell application "Adobe InDesign CS6"
        set myGraphicsCheck to true
        repeat with myCounter from 1 to (count graphics of myDocument)
            set myGraphic to graphic myCounter of myDocument
            set myLink to item link of myGraphic
            if link status of myLink is not normal then
                set myGraphicsCheck to false
                exit repeat
            end if
        end repeat
        return myGraphicsCheck
    end tell
end myCheckGraphics
```

選択に関連するイベントリスナーの例

InDesign では、選択に関連するイベントに応答することができます。InDesign ドキュメントのオブジェクトを選択したり選択解除したりしたときには、`afterSelectionChanged` イベントが発生します。選択したオブジェクトの属性（フォーマットや位置）を変更したときには、`afterSelectionAttributeChanged` イベントが発生します。この 2 つのイベントは、ユーザーアクションに応答するスクリプトを作成する場合に便利です。

選択が変更されたときにオブジェクトの種類を取得して表示する方法を、次のスクリプトに示します。完全なスクリプトについては、AfterSelectionChanged を参照してください。

```
set myDocument to make document
tell myDocument
    set myEventListener to make event listener with properties {event
type:"afterSelectionChanged",
    handler:my myDisplaySelectionType}
end tell
```

このスクリプトで参照されているイベントハンドラーを次に示します。

```
on myDisplaySelectionType(en)
    tell application "Adobe InDesign CS6"
        if (count documents) is greater than 0 then
            tell document 1
                set mySelection to selection
                if (count mySelection) is greater than 0 then
                    set myString to "Selection Contents:" & return
                    repeat with myCounter from 1 to (count mySelection)
                        set myString to myString & class of item myCounter of
                        mySelection & return
                    end repeat
                    display dialog myString
                end if
            end tell
        end if
    end tell
end myDisplaySelectionType
```

このスクリプトで追加したイベントリスナーを削除するには、RemoveAfterSelectionChanged スクリプトを実行します。

選択の属性の変更に応答する方法を、次のスクリプトに示します。この例では、選択で Registration スウォッチが適用されたかどうかをイベントハンドラーで確認します （Registration スウォッチを誤って適用すると、業務用のプリンターで問題が発生する可能性があります）。Registration スウォッチが適用されていた場合、変更が意図的に行われたものであるかどうかをユーザーに確認します。完全なスクリプトについては、AfterSelectionAttributeChanged を参照してください。

```
set myDocument to make document
tell myDocument
    set myEventListener to make event listener with properties {event
type:"afterSelectionAttributeChanged", handler:my myCheckForRegistration}¥
end tell
```

このスクリプトで参照されているイベントハンドラーを次に示します。

```
on myCheckForRegistration()
    tell application "Adobe InDesign CS6"
        if (count documents) is greater than 0 then
            tell document 1
                set mySelection to selection
                if (count mySelection) is greater than 0 then
                    set myRegistrationSwatchUsed to false
                    repeat with myCounter from 1 to (count mySelection)
                        set myFillColor to fill color of item myCounter of mySelection
                        set myStrokeColor to stroke color of item myCounter of mySelection
                        if name of myFillColor is "Registration" or
                        name of myStrokeColor is "Registration" then
                            set myRegistrationSwatchUsed to true
                        end if
                    end repeat
                    if myRegistrationSwatchUsed is true then
                        display dialog "The Registration swatch is applied to some of the"
& return &
                        "objects in the selection. Did you really intend to apply this swatch?"
                    end if
                end if
            end tell
        end if
    end tell
end myCheckForRegistration
```

このスクリプトで追加したイベントリスナーを削除するには、RemoveAfterSelectionAttributeChanged スクリプトを実行します。

「onIdle」イベントリスナーの例

InDesign のアイドルタスクは、アプリケーションのイベントキューに処理対象のイベントが存在しないときに実行されます。アイドルタスクは、スクリプトで簡単に実行できます。`onIdle` イベントにより、スクリプトベースのアイドルタスクを実行する方法が提供されます。このイベントを使用して、InDesign および InCopy がアイドル状態のときに自動的にスクリプトを実行できます。イベントターゲットは `IdleTask` に、イベントオブジェクトは `IdleEvent` になります。

アイドルタスクの `sleep` プロパティは、タスクが再度呼び出されるまでの時間です。当然ですが、この待機時間には、ユーザーの作業に支障が出ないだけの大きな値を設定する必要があります。ただし、この値は、スクリプトで実行するタスクによって異なります。

待機時間をゼロに設定すると、タスクは削除されます（ただし、イベントリスナーは削除されません）。この方法は、アイドルタスクを中止する場合に便利です。

イベントリスナーを追加し、アイドルタスクからメッセージボックスを表示する方法を、次のスクリプトに示します（完全なスクリプトについては、Reminder を参照してください）。

```
set myIdleTaskName to "my_idle_task"
set myIdleTask to make idle task with properties {name:myIdleTaskName, sleep:10000}
tell myIdleTask
    --You need to fill in your own file path.
    set fileName to "Macintosh HD:scripting:OnIdleEventHandler.applescript"
    set onIdleEventListener to make event listener with properties {event type:"onIdle",
handler:fileName}
    display alert "Created idle task " & name & "; added event listener on " & event type of
onIdleEventListener
end tell
```

イベントハンドラーは、OnIdleEventHandler.applescript というスクリプトファイルです。

```
--"evt" is the event passed to this script by the event listener.
onIdleEventHandler(evt)
on onIdleEventHandler(myIdleEvent)
    tell application "Adobe InDesign CS6"
        if (count of documents) = 0 then
            set myDoc to make document
            display alert "Created document " & name of myDoc & " in idle task."
            return
        end if

        tell item 1 of pages of active document
            if (count of text frames) = 0 then
                make text frame with properties {geometric bounds:["72pt", "72pt", "288pt",
"288pt"], contents:"Text frame created in idle task"}
                display alert "Created a text frame in idle task."
                return
            end if
        end tell

        --Delete idle task by setting its sleep time to zero.
        set sleep of parent of myIdleEvent to 0
        display alert "Nothing to do. Delete idle task."
    end tell
end onIdleEventHandler
```

このスクリプトで作成したアイドルタスクを削除するには、次のスクリプトを実行します（完全なスクリプトについては、RemoveIdleTask を参照してください）。

```
set taskCount to count of idle tasks
if taskCount is 0 then
    display alert "There is no idle task."
else
    set myIdleTaskName to "my_idle_task"
    repeat with i from taskCount to 1 by -1
        set myIdleTask to item i of idle tasks
        if name of myIdleTask is myIdleTaskName then
            delete myIdleTask
        end if
    end repeat
    display alert "Idle task " & myIdleTaskName & " removed."
end if
```

すべてのアイドルタスクを削除するには、次のスクリプトを実行します（完全なスクリプトについては、RemoveAllIdleTasks を参照してください）。

```
set taskCount to count of idle tasks
if taskCount is 0 then
    display alert "There is no idle task."
else
    repeat with i from taskCount to 1 by -1
        set myIdleTask to item i of idle tasks
        delete myIdleTask
    end repeat
    display alert "" & taskCount & " idle task(s) removed."
end if
```

既存のアイドルタスクを表示するには、次のスクリプトを実行します（完全なスクリプトについては、ListIdleTasks を参照してください）。

```
set taskCount to count of idle tasks
if taskCount is 0 then
    display alert "There is no idle task."
else
    set str to ""
    repeat with i from 1 to taskCount
        set myIdleTask to item i of idle tasks
        tell myIdleTask
            set str to str & "idle task " & id & ": " & name & return
        end tell
    end repeat
    display alert str
end if
```

9 メニュー

章の更新ステータス	
CS6	変更なし

InDesign スクリプティングでは、メニュー項目の追加と削除、任意のメニューコマンドの実行、メニュー項目へのスクリプトの割り当てを行うことができます。

この章では、InDesign のメニューをスクリプトで操作する方法について説明します。最初は簡単なサンプルスクリプトから始めて、徐々に複雑なスクリプトへと進んでいきます。

ここでは、読者が『Adobe InDesign スクリプティングチュートリアル』を既に読んでおり、スクリプトを作成、インストール、実行する方法を理解しているものとして説明を行います。

メニューモデルについて

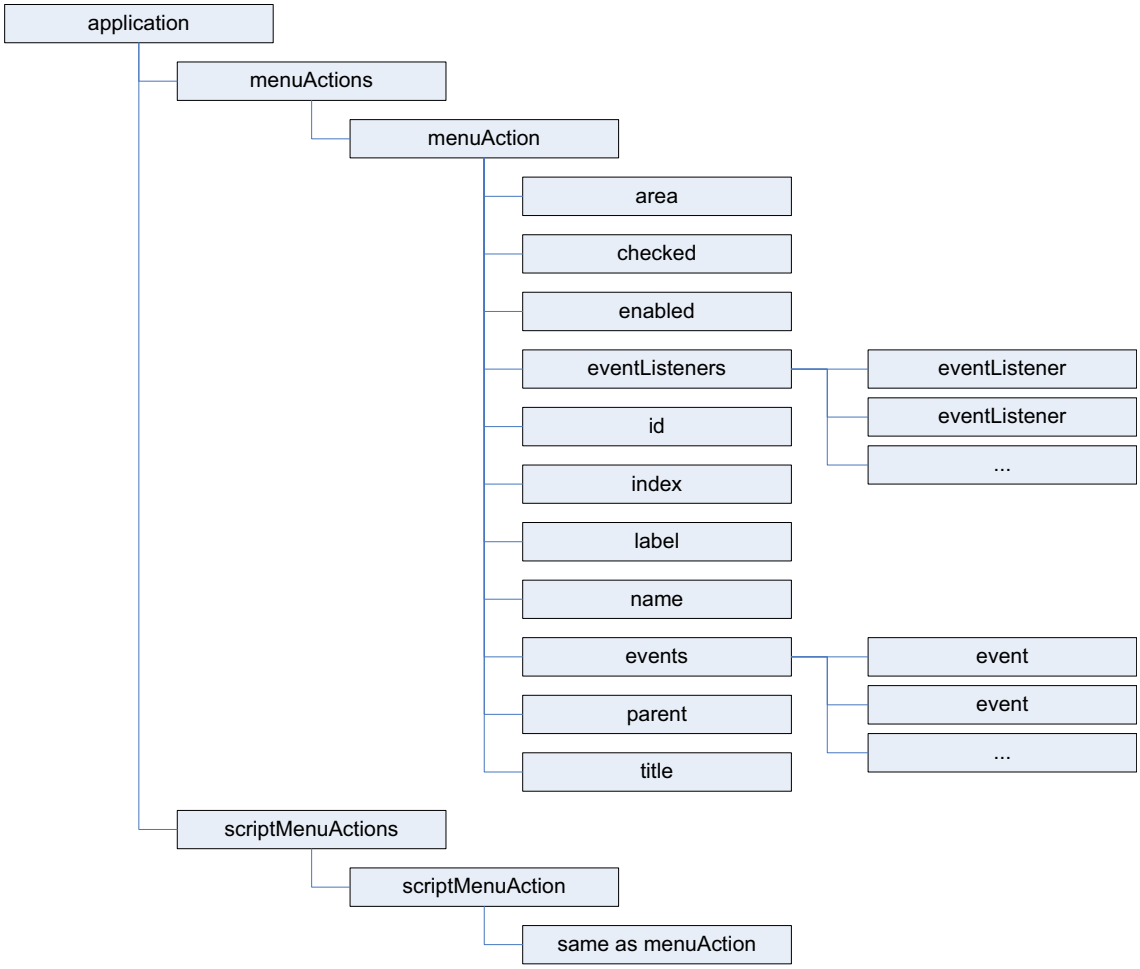
InDesign のメニュースクリプトモデルは、アプリケーションのユーザーインターフェイスに表示されるメニューに対応する一連のオブジェクトによって構成されています（メインのメニューバーに表示されるメニューだけでなく、パネルに用意されているメニューも含みます）。menu オブジェクトには、次のオブジェクトが含まれています。

- menu items — メニューに表示されるメニューオプション （サブメニューは含まれません）
- menu separators — メニュー内のメニューオプションを区切るために使用される線
- submenus — 更なるメニュー選択が含まれているメニューオプション
- menu elements — メニューに表示されるすべての menu items、menu separators、submenus
- event listeners — メニューに対するユーザーアクションやスクリプトアクションに応答するためのリスナー
- events — メニューによって発生した events

どの menu item にも、associated menu action というプロパティによって menu action が割り当てられています。そのメニュー項目が選択されたときの処理は、menu action のプロパティによって定義されています。ユーザーインターフェイスで定義されている menu action に加えて、独自の script menu action を作成して、メニュー項目にスクリプトを割り当てることができます。

1 つの menu action や script menu action は、0 個以上の menu item に関連付けることができます。

次の図に、様々なメニューオブジェクトの関係を示します。



すべてのメニューアクションのリスト（テキストファイル）を作成するには、次のスクリプトを実行します（チュートリアルスクリプトの GetMenuActions より）。

```
set myTextFile to choose file name("Save menu action names as:")
if myTextFile is not equal to "" then
    tell application "Adobe InDesign CS6"
        set myString to ""
        set myMenuActionNames to name of every menu action
        repeat with myMenuActionName in myMenuActionNames
            set myString to myString & myMenuActionName & return
        end repeat
        my myWriteToFile(myString, myTextFile, false)
    end tell
end if
on myWriteToFile(myString, myFileName, myAppendData)
    set myTextFile to open for access myFileName with write permission
    if myAppendData is false then
        set eof of myTextFile to 0
    end if
    write myString to myTextFile starting at eof
    close access myTextFile
end myWriteToFile
```

利用可能なメニューのリスト（テキストファイル）を作成するには、次のスクリプトを実行します（完全なスクリプトについては、GetMenuNames を参照してください）。InDesign には多数のメニューがあるので、このスクリプトの実行には時間がかかる場合があります。

```
--Open a new text file.
set myTextFile to choose file name ("Save Menu Action Names As")
--If the user clicked the Cancel button, the result is null.
if (myTextFile is not equal to "") then
    tell application "Adobe InDesign CS6"
        --Open the file with write access.
        my myWriteToFile("Adobe InDesign CS6 Menu Names" & return,
        myTextFile, false)
        repeat with myCounter from 1 to (count menus)
            set myMenu to item myCounter of menus
            set myString to "----------" & return & name of myMenu & return &
            "----------" & return
            set myString to my myProcessMenu(myMenu, myString)
            my myWriteToFile(myString, myTextFile, true)
        end repeat
        display dialog ("done!")
    end tell
end if
on myProcessMenu(myMenu, myString)
    tell application "Adobe InDesign CS6"
        set myIndent to my myGetIndent(myMenu)
        repeat with myCounter from 1 to (count menu elements of myMenu)
            set myMenuElement to menu element myCounter of myMenu
            set myClass to class of myMenuElement
            if myClass is not equal to menu separator then
                set myMenuElementName to name of myMenuElement
                set myString to myString & myIndent & myMenuElementName & return
                if class of myMenuElement is submenu then
                    if myMenuElementName is not "Font" then
                        set myString to my myProcessMenu(myMenuElement, myString)
                    end if
                end if
            end if
        end repeat
        return myString
    end tell
end myProcessMenu
on myGetIndent(myObject)
    tell application "Adobe InDesign CS6"
        set myString to ""
        repeat until class of myObject is menu
            set myString to myString & tab
            set myObject to parent of myObject
        end repeat
        return myString
    end tell
end myGetIndent
on myWriteToFile(myString, myFileName, myAppendData)
    set myTextFile to open for access myFileName with write permission
    if myAppendData is false then
        set eof of myTextFile to 0
    end if
    write myString to myTextFile starting at eof
    close access myTextFile
end myWriteToFile
```

ローカライゼーションとメニュー名

InDesign スクリプティングでは、menu item、menu、menu action、submenu はすべて名前で参照されます。したがって、アプリケーションがインストールされているロケールに依存せずにオブジェクトを指定する方法が必要になります。これを行うには、ロケールに依存せずに特定の項目を参照するための文字列が格納されている内部データベースを使用します。例えば、次のスクリプトを使用すれば、ロケールに依存しないメニューアクション名を取得することができます（完全なスクリプトについては、GetKeyStrings を参照してください）。

```
tell application "Adobe InDesign CS6"
    set myMenuAction to menu action "Convert to Note"
    set myKeyStrings to find key strings for title of myMenuAction
    if class of myKeyStrings is list then
        repeat with myKeyString in myKeyStrings
            set myString to myKeyString & return
        end repeat
    else
        set myString to myKeyStrings
    end if
    display dialog(myString)
end tell
```

注意 : ロケールに依存しない名前を取得する場合は、menu、menu item、submenu の名前を取得するよりも、menu action の名前を取得するほうが便利です。menu action のタイトルは、単一の文字列である場合が多いからです。その他のメニューオブジェクトの多くからは、find key strings メソッドで複数の文字列が返されます。

ロケールに依存しない名前を取得したら、スクリプトで使用することができます。この文字列を使用したスクリプトは、現在使用している InDesign と異なるロケールにインストールされている InDesign でも正常に機能します。

ロケールに依存しない文字列を現在のロケールに変換するには、次のスクリプトを使用します（チュートリアルスクリプトの TranslateKeyString より）。

```
tell application "Adobe InDesign CS6"
    set myString to translate key string for "$ID/NotesMenu_ConvertToNote"
    display dialog(myString)
end tell
```

スクリプトからのメニューアクションの実行

InDesign に組み込まれている menu action は、いずれもスクリプトから実行することができます。すべての menu action が menu item に割り当てられているとは限りませんが、その点を除けば、スクリプトから menu action を実行することは、ユーザーインターフェイスでメニューオプションを選択することとまったく同じです。例えば、あるメニューオプションを選択するとダイアログボックスが表示される場合、それに対応する menu action をスクリプトから実行すると同じダイアログボックスが表示されます。

スクリプトから menu action を実行する方法を、次のスクリプトに示します（完全なスクリプトについては、InvokeMenuAction を参照してください）。

```
tell application "Adobe InDesign CS6"
    --Get a reference to a menu action.
    set myMenuAction to menu action "$ID/NotesMenu_ConvertToNote"
    --Run the menu action. The example action will fail if you do not
    --have text selected.
    invoke myMenuAction
end tell
```

注意 : 一般的に、InDesign のプロセスをスクリプトで自動化する場合、メニューアクションやユーザーインターフェイスを選択することで必要な処理を実行するのは、あまりよい方法とは言えません。InDesign のスクリプトオブジェクトモデルを操作するほうが、はるかに確実で強力な処理を実現できます。メニューアクションは、選択やウィンドウの状態など、ユーザーインターフェイスの様々な条件に依存しています。オブジェクトモデルを使用すれば、InDesign ドキュメント内のオブジェクトを直接操作できるので、ユーザーインターフェイスに依存することなく、一貫性のある高速な処理を実現できます。

メニューやメニュー項目の追加

InDesign のユーザーインターフェイスと同じように、スクリプトでもメニューやメニュー項目の作成や削除が行えます。新しい場所に新しいメニューを作成して、あるサブメニューの内容をそのメニューに複製する方法を、次のサンプルスクリプトに示します（完全なスクリプトについては、CustomizeMenu を参照してください）。

```
tell application "Adobe InDesign CS6"
    set myMainMenu to menu "Main"
    set myTypeMenu to submenu "Type" of myMainMenu
    set myFontMenu to submenu "Font" of myTypeMenu
    set myKozukaMenu to submenu "Kozuka Mincho Pro " of myFontMenu
    tell myMainMenu
        set mySpecialFontMenu to make submenu with properties
        {title:"Kozuka Mincho Pro"}
    end tell
    repeat with myMenuItem in menu items of myKozukaMenu
        set myAssociatedMenuAction to associated menu action of myMenuItem
        tell mySpecialFontMenu
            make menu item with properties
            {associated menu action:myAssociatedMenuAction}
        end tell
    end repeat
end tell
```

このスクリプトで作成されたカスタムメニュー項目を削除するには、RemoveCustomMenu を使用します。

```
tell application "Adobe InDesign CS3"
    set myMainMenu to menu "Main"
    try
        set myKozukaMenu to submenu "Kozuka Mincho Pro" of myMainMenu
        tell myKozukaMenu to delete
    end try
end tell
```

メニューとイベント

ユーザーインターフェイスでメニューやサブメニューを選択すると、イベントが生成されます。これと同じように、menu action や script menu action を実行した場合にも、イベントが生成されます。スクリプトを使用すれば、これらのイベントに応答するために event listener を登録することができます。様々なメニュースクリプティングコンポーネントのイベントを次の表に示します。

オブジェクト	イベント	説明
menu	beforeDisplay	メニューの内容が表示される前に、登録したスクリプトが実行されます。
menu action	afterInvoke	関連付けられている menu item が選択されて、onInvoke イベントが発生した後に、登録したスクリプトが実行されます。
	beforeInvoke	関連付けられている menu item が選択されて、onInvoke イベントが発生する前に、登録したスクリプトが実行されます。

オブジェクト	イベント	説明
script menu action	afterInvoke	関連付けられている menu item が選択されて、onInvoke イベントが発生した後に、登録したスクリプトが実行されます。
	beforeInvoke	関連付けられている menu item が選択されて、onInvoke イベントが発生する前に、登録したスクリプトが実行されます。
	beforeDisplay	script menu action の有効／チェックのステータスに対する内部要求が行われる前に、登録したスクリプトが実行されます。
	onInvoke	script menu action が呼び出されたときに、登録したスクリプトが実行されます。
submenu	beforeDisplay	submenu の内容が表示される前に、登録したスクリプトが実行されます。

event や event listener について詳しくは、第 8 章「イベント」を参照してください。

メニューに表示する項目を変更するには、beforeDisplay event に対する event listener を追加します。メニューが選択されると、登録したスクリプト（メニュー項目の有効化／無効化や名前の変更などの、メニューに関連するタスクを実行するスクリプト）が event listener によって実行されます。この仕組みは、利用可能なフォント、最近使用したドキュメント、開いているウィンドウなどをメニューに表示するために、内部的に使用されています。

scriptMenuAction の操作

script menu action を使用すると、menu action を新しく作成して、onInvoke event の発生時に指定のスクリプトを実行することができます。

script menu action を作成してメニュー項目に割り当てる方法を、次のスクリプトに示します（完全なスクリプトについては、MakeScriptMenuAction を参照してください）。このスクリプトでは、メニュー項目が選択されると、単に警告が表示されます。

```
tell application "Adobe InDesign CS6"
    --Create the script menu action "Display Message"
    --if it does not already exist.
    try
        set myScriptMenuAction to script menu action "Display Message"
    on error
        set myScriptMenuAction to make script menu action
        with properties {title:"Display Message"}
    end try
    tell myScriptMenuAction
        --If the script menu action already existed,
        --remove the existing event listeners.
        if (count event listeners) > 0 then
            tell every event listener to delete
        end if
        set myEventListener to make event listener with properties
        {event type:"onInvoke", handler:"yukino:message.applescript"}
    end tell
    tell menu "$ID/Main"
        set mySampleScriptMenu to make submenu with properties
        {title:"Script Menu Action"}
        tell mySampleScriptMenu
            set mySampleScriptMenuItem to make menu item with properties
            {associated menu action:myScriptMenuAction}
        end tell
    end tell
end tell
```

message.applescript スクリプトファイルには、次のコードが含まれます。

```
    tell application "Adobe InDesign CS6"
        display dialog ("You selected an example script menu action.")
    end tell
```

このスクリプトで作成した menu、submenu、menu item、script menu action を削除するには、次のスクリプトを実行します（チュートリアルスクリプトの RemoveScriptMenuAction より）。

```
tell application "Adobe InDesign CS6"
    try
        set myScriptMenuAction to script menu action "Display Message"
        tell myScriptMenuAction
            delete
        end tell
        tell submenu "Script Menu Action" of menu "$ID/Main" to delete
    end try
end tell
```

すべての script menu action を削除することもできます。次のスクリプトにその例を示します（チュートリアルスクリプトの RemoveAllScriptMenuActions より）。このスクリプトでは、script menu action のメニューリストも削除されますが、作成したメニューやサブメニューは削除されません。

```
tell application "Adobe InDesign CS6"
    delete every script menu action
end tell
```

現在のすべての script menu action のリストを作成することができます。次のスクリプトにその例を示します（チュートリアルスクリプトの ListScriptMenuActions より）。

```
set myTextFile to choose file name {"Save Script Menu Action Names As"}
--If the user clicked the Cancel button, the result is null.
if myTextFile is not equal to "" then
    tell application "Adobe InDesign CS6"
        set myString to ""
        set myScriptMenuActionNames to name of every script menu action
        repeat with myScriptMenuActionName in myScriptMenuActionNames
            set myString to myString & myScriptMenuActionName & return
        end repeat
        my myWriteToFile(myString, myTextFile, false)
    end tell
end if
on myWriteToFile(myString, myFileName, myAppendData)
    set myTextFile to open for access myFileName with write permission
    if myAppendData is false then
        set eof of myTextFile to 0
    end if
    write myString to myTextFile starting at eof
    close access myTextFile
end myWriteToFile
```

script menu action を使用して、beforeDisplay event の発生時にスクリプトを実行することもできます。この場合は、script menu action の状態に対する内部要求が行われる前に（メニュー項目が表示される直前に）スクリプトが実行されます。これは、スクリプトでメニュー名を変更したり、有効／チェックのステータスを設定したりする場合に特に役立ちます。

次のサンプルスクリプトでは、beforeDisplay event に対する event listener を追加して、現在の選択を確認するようにしています。何も選択されていない場合は、event listener のスクリプトによってメニュー項目が無効化されます。アイテムが選択されている場合は、メニュー項目が有効化されます。メニュー項目を選択すると、選択されている最初のアイテムの種類が表示されます （完全なスクリプトについては、BeforeDisplay を参照してください）。

```
tell application "Adobe InDesign CS6"
    --Create the script menu action "Display Message"
    --if it does not already exist.
    try
        set myScriptMenuAction to script menu action "Display Message"
    on error
        set myScriptMenuAction to make script menu action with properties
        {title:"Display Message"}
    end try
    tell myScriptMenuAction
        --If the script menu action already existed,
        --remove the existing event listeners.
        if (count event listeners) > 0 then
            tell every event listener to delete
        end if
        --Fill in a valid file path for your system.
        make event listener with properties {event type:"onInvoke",
        handler:"yukino:WhatIsSelected.applescript"}
    end tell
    tell menu "$ID/Main"
        set mySampleScriptMenu to make submenu with properties
        {title:"Script Menu Action"}
        tell mySampleScriptMenu
            set mySampleScriptMenuItem to make menu item with properties
            {associated menu action:myScriptMenuAction}
            --Fill in a valid file path for your system.
            make event listener with properties {event type:"beforeDisplay",
            handler:"yukino:BeforeDisplayHandler.applescript"}
        end tell
    end tell
end tell
```

BeforeDisplayHander チュートリアルスクリプトファイルには、次のスクリプトが含まれます。

```
tell application "Adobe InDesign CS6"
    try
        set mySampleScriptAction to script menu action "Display Message"
        set mySelection to selection
        if (count mySelection) > 0 then
            set enabled of mySampleScriptAction to true
        else
            set enabled of mySampleScriptAction to false
        end if
    on error
        alert("Script menu action did not exist.")
    end try
end tell
```

WhatIsSelected チュートリアルスクリプトファイルには、次のスクリプトが含まれます。

```
tell application "Adobe InDesign CS6"
    set mySelection to selection
    if (count mySelection) > 0 then
        set myString to class of item 1 of mySelection as string
        display dialog ("The first item in the selection is a " & myString & ".")
    end if
end tell
```

さらに複雑なメニュースクリプティングの例

InDesign のユーザーインターフェイスで別の項目を選択すると、コンテキストメニューの内容が変わります。選択したオブジェクトのプロパティに基づいてコンテキストメニューを変更する方法を、次のサンプルスクリプトに示します。このサンプルスクリプトは断片的なスクリプトです。完全なスクリプトについては、LayoutContextMenu を参照してください。

次のスクリプトでは、レイアウトコンテキストメニュー（ページアイテムを選択すると表示されるコンテキストメニュー）に新規のメニュー項目を追加する方法を示します。このスクリプトでは、beforeDisplay event listener が追加されています。このイベントリスナーは、menu item が既に存在するかどうかを確認し、存在する場合は削除します。これにより、グラフィックが選択されていない場合はコンテキストメニューに menu item が表示されず、コンテキストメニューに複数のメニュー項目が追加されるのを防ぎます。event listener では、次に、グラフィックが選択されているかどうかを確認し、選択されている場合は新規の script menu item が作成されます。

```
tell application "Adobe InDesign CS6"
    --The locale-independent name (aka "key string") for the
    --Layout context menu is "$ID/RtMouseLayout".
    set myLayoutMenu to menu "$ID/RtMouseLayout"
    --Note that the following script actions only create the script menu action
    --and set up event listeners--they do not actually add the menu item to the
    --Layout context menu. That job is taken care of by the event handler scripts
    --themselves. The Layout context menu will not display the menu item unless
    --a graphic is selected.
    tell myLayoutContextMenu
        set myBeforeDisplayEventListener to make event listener with
        properties{event type:"beforeDisplay",
        handler:"yukino:IDEventHandlers:LayoutMenuBeforeDisplay.applescript",
        captures:false}
    end tell
end tell
```

この例で参照されている LayoutMenuBeforeDisplay.applescript ファイルには、次のコードが含まれています。

```
myBeforeDisplayHandler(evt)
on myBeforeDisplayHandler(myEvent)
    tell application "Adobe InDesign CS6"
        set myGraphicList to {PDF, EPS, image}
        set myParentList to {rectangle, oval, polygon}
        set myObjectList to {}
        set myLayoutContextMenu to menu "$ID/RtMouseLayout"
        if (count documents) > 0 then
            set mySelection to selection
            if (count mySelection) > 0 then
                repeat with myCounter from 1 to (count mySelection)
                    set myObject to item myCounter of mySelection
                    if class of myObject is in myGraphicList then
                        copy myObject to end of myObjectList
                    else if class of myObject is in myParentList then
                        if (count graphics of myObject) > 0 then
                            copy graphic 1 of myObject to end of myObjectList
                        end if
                    end if
                end repeat
                if (count myObjectList) is not equal to 0 then
                    --The selection contains a qualifying item or items.
                    --Add the menu item if it does not already exist.
                    if my myMenuItemExists(myLayoutContextMenu,
                    "Create Graphic Label") is false then
                        my myMakeLabelGraphicMenuItem(myLayoutContextMenu)
                    end if
                else
```

```
                    --Remove the menu item if it exists.
                    if my myMenuItemExists(myLayoutContextMenu,
                    "Create Graphic Label") is true then
                        tell myLayoutContextMenu
                            delete menu item "Create Graphic Label"
                        end tell
                    end if
                end if
            end if
        end if
    end tell
end myBeforeDisplayHandler
on myMakeLabelGraphicMenuItem(myLayoutContextMenu)
    tell application "Adobe InDesign CS6"
        if my myScriptMenuActionExists("Create Graphic Label") is false then
            set myLabelGraphicMenuAction to make script menu action with
            properties {name:"Create Graphic Label"}
            tell myLabelGraphicMenuAction


                set myLabelGraphicEventListener to make event listener
                with properties {event type:"onInvoke",handler:
                "yukino:IDEventHandlers:LayoutMenuOnInvoke.applescript",
                captures:false}
            end tell
        else
            set myLabelGraphicMenuAction to script menu action
            "Create Graphic Label"
        end if
        tell myLayoutContextMenu
            set myLabelGraphicMenuItem to make menu item with properties
            {associated menu action:myLabelGraphicMenuAction}
        end tell
    end tell
end myMakeLabelGraphicMenuItem
```

この例で参照されている `LayoutMenuOnInvoke.applescript` では、`menu item` が選択されるとアクティブになる（`onInvoke` イベント）`script menu action` が定義されます。

```
LabelGraphicEventHandler(evt)
on LabelGraphicEventHandler(myEvent)
    tell application "Adobe InDesign CS6"
        set myGraphicList to {PDF, EPS, image}
        set myParentList to {rectangle, oval, polygon}
        set myObjectList to {}
        set myLayoutContextMenu to menu "$ID/RtMouseLayout"
        if (count documents) > 0 then
            set mySelection to selection
            if (count mySelection) > 0 then
                repeat with myCounter from 1 to (count mySelection)
                    set myObject to item myCounter of mySelection
                    if class of myObject is in myGraphicList then
                        copy myObject to end of myObjectList
                    else if class of myObject is in myParentList then
                        if (count graphics of myObject) > 0 then
                            copy graphic 1 of myObject to end of myObjectList
                        end if
                    end if
                end repeat
                if (count myObjectList) is not equal to 0 then
                    --The selection contains a qualifying item or items.
                    my myDisplayDialog(myObjectList)
                end if
            end if
        end if
    end tell
```

```
end LabelGraphicEventHandler
--Displays a dialog box containing label options.
on myDisplayDialog(myObjectList)
    tell application "Adobe InDesign CS6"
        set myLabelWidth to 100
        set myStyleNames to my myGetParagraphStyleNames(document 1)
        set myLayerNames to my myGetLayerNames(document 1)
        set myDialog to make dialog with properties {name:"LabelGraphics"}
        tell myDialog
            tell (make dialog column)
                --Label Type
                tell (make dialog row)
                    tell (make dialog column)
                        make static text with properties
                        {static label:"Label Type:", min width:myLabelWidth}
                    end tell
                    tell (make dialog column)
                        set myLabelTypeDropdown to make dropdown with properties
                        {string list:{"File Name", "File Path", "XMP Author",
                        "XMP Description"}, selected index:0}
                    end tell
                end tell
                --Text frame height
                tell (make dialog row)
                    tell (make dialog column)
                        make static text with properties
                        {static label:"Label Height:", min width:myLabelWidth}
                    end tell
                    tell (make dialog column)
                        set myLabelHeightField to make measurement editbox
                        with properties {edit value:24, edit units:points}
                    end tell
                end tell
                --Text frame offset
                tell (make dialog row)
                    tell (make dialog column)
                        make static text with properties
                        {static label:"Label Offset:", min width:myLabelWidth}
                    end tell
                    tell (make dialog column)
                        set myLabelOffsetField to make measurement editbox with
                        properties {edit value:0, edit units:points}
                    end tell
                end tell
                --Paragraph style to apply
                tell (make dialog row)
                    tell (make dialog column)
                        make static text with properties
                        {static label:"Label Style:", min width:myLabelWidth}
                    end tell
                    tell (make dialog column)
                        set myLabelStyleDropdown to make dropdown with properties
                        {string list:myStyleNames, selected index:0}
                    end tell
                end tell
                --Layer
                tell (make dialog row)
                    tell (make dialog column)
                        make static text with properties
                        {static label:"Layer:", min width:myLabelWidth}
```

```
                    end tell
                    tell (make dialog column)
                        set myLayerDropdown to make dropdown with properties
                        {string list:myLayerNames, selected index:0}
                    end tell
                end tell
            end tell
        end tell
        set myResult to show myDialog
        if myResult is true then
            set myLabelType to (selected index of myLabelTypeDropdown)
            set myLabelHeight to edit value of myLabelHeightField
            set myLabelOffset to edit value of myLabelOffsetField
            set myLabelStyleName to item ((selected index of


            myLabelStyleDropdown)+ 1) of myStyleNames
            set myLayerName to item ((selected index of myLayerDropdown) + 1)
            of myLayerNames
            destroy myDialog
            tell view preferences of document 1
                set myOldXUnits to horizontal measurement units
                set myOldYUnits to vertical measurement units
                set horizontal measurement units to points
                set vertical measurement units to points
            end tell
            repeat with myCounter from 1 to (count myObjectList)
                set myGraphic to item myCounter of myObjectList
                my myAddLabel(myGraphic, myLabelType, myLabelHeight,
                myLabelOffset, myLabelStyleName, myLayerName)
            end repeat
            tell view preferences of document 1
                set horizontal measurement units to myOldXUnits
                set vertical measurement units to myOldYUnits
            end tell
        else
            destroy myDialog
        end if
    end tell
end myDisplayDialog
on myAddLabel(myGraphic, myLabelType, myLabelHeight, myLabelOffset, myLabelStyleName,
myLayerName)
    tell application "Adobe InDesign CS6"
        set myDocument to document 1
        set myLabelStyle to paragraph style myLabelStyleName of myDocument
        try
            set myLabelLayer to layer myLayerName of myDocument
        on error
            tell myDocument
                set myLabelLayer to make layer with properties {name:myLayerName}
            end tell
        end try
        set myLink to item link of myGraphic
        if myLabelType is 0 then
            set myLabel to name of myLink
        else if myLabelType is 1 then
            set myLabel to file path of myLink
        else if myLabelType is 2 then
            try
```

```
                set myLabel to author of link xmp of myLink
            on error
                set myLabel to "No author available."
            end try
        else if myLabelType is 3 then
            try
                set myLabel to description of link xmp of myLink
            on error
                set myLabel to "No description available."
            end try
        end if
        set myFrame to parent of myGraphic
        set myBounds to geometric bounds of myFrame
        set myX1 to item 2 of myBounds


        set myY1 to (item 3 of myBounds) + myLabelOffset
        set myX2 to item 4 of myBounds
        set myY2 to myY1 + myLabelHeight
        tell parent of myFrame
            set myTextFrame to make text frame with properties
            {item layer:myLabelLayer, contents:myLabel,
            geometric bounds:{myY1, myX1, myY2, myX2}}
            set first baseline offset of text frame preferences of
            myTextFrame to leading offset
            tell myTextFrame
                set applied paragraph style of paragraph 1 to myLabelStyle
            end tell
        end tell
    end tell
end myAddLabel
```

10 プリフライトの操作

章の更新ステータス	
CS6	変更なし

プリフライトを実行すると、パブリケーションを出力デバイスに送る前に、ファイルやフォント、アセット（配置画像や PDF ファイルなど）、プリンター設定、トラップスタイルなど、必要なものがすべてそろっているかどうかを検証できます。例えば、画像を低解像度プロキシで配置しているが、使用中のハードディスク（またはワークグループのサーバー）で高解像度の元の画像にアクセスできない場合、プリント処理時にエラーが発生することがあります。プリフライトは、このような問題を検証します。プリフライトは、ユーザーの作業中にバックグラウンドで実行できます。

この章では、スクリプトを使用して、インタラクティブにプリフライトシステムを操作する方法について説明します。説明のため、ページサイズがレターサイズ（8.5" x 11"）以外のときにエラーを発生させるように、プリフライトを設定する方法を紹介します。ユーザーインターフェイスでの設定方法を簡単に紹介し、その後でスクリプトを使用して同じ設定を行う方法について説明します。

ここでは、読者が『Adobe InDesign スクリプティングチュートリアル』を既に読んでおり、スクリプトを作成、インストール、実行する方法を理解しているものとして説明を行います。

プリフライトプロファイルの調査

InDesign のプリフライト機能は、プロファイルベースで、ルールに従って実行されます。また、パラメーターを使用します。プリフライトプロファイルは、1 つ以上存在します。最初は、[基本] という名前のプロファイルが 1 つ用意されています。このプロファイルは読み取り専用で、変更したり削除したりできません。

プリフライトプロファイルには、様々なプリフライトルールが含まれています。各ルールは、名前と複数のデータオブジェクトを保持しています。各データオブジェクトには、名前、データ型、データ値があります。データ値は変更することができます。各ルールは、次のように設定できます。

- 無効 — プリフライトルールは無効化されます。
- エラーを返す — プリフライトルールは、エラーレベルのフィードバックを返します。
- 警告を返す — プリフライトルールは、警告レベルのフィードバックを返します。
- 通知を返す — プリフライトルールは、通知レベルのフィードバックを返します。

InDesign でプロファイルを確認するには、プリフライトパネル／プロファイルを定義を選択します。また、スクリプトでもプロファイル情報を取得できます。

プリフライトプロファイルの一覧表示

すべてのプリフライトプロファイルを一覧表示する方法を、次のスクリプトに示します。完全なスクリプトについては、ListPreflightProfiles を参照してください。

```
tell application "Adobe InDesign CS6"
    set myProfiles to preflight profiles
    set myProfileCount to count of myProfiles
    set myStr to "Preflight profiles: "
    repeat with i from 1 to myProfileCount
        if i > 1 then
            set myStr to myStr & ", "
        end if
        set myStr to myStr & name of item i of myProfiles
    end repeat
    display alert myStr
end tell
```

プリフライトルールの一覧表示

プロファイル内のすべてのプリフライトルールを一覧表示する方法を、次のスクリプトに示します。完全なスクリプトについては、ListPreflightRules を参照してください。

```
tell application "Adobe InDesign CS6"
    -- Assume the [Basic] profile exists.
    set myProfile to item 1 of preflight profiles
    set myRules to preflight profile rules of myProfile
    set ruleCount to count of myRules
    set str to "Preflight rules of " & name of myProfile & ": "
    repeat with i from 1 to ruleCount
        if i > 1 then
            set str to str & ", "
        end if
        set str to str & name of item i of myRules
    end repeat
    display alert str
end tell
```

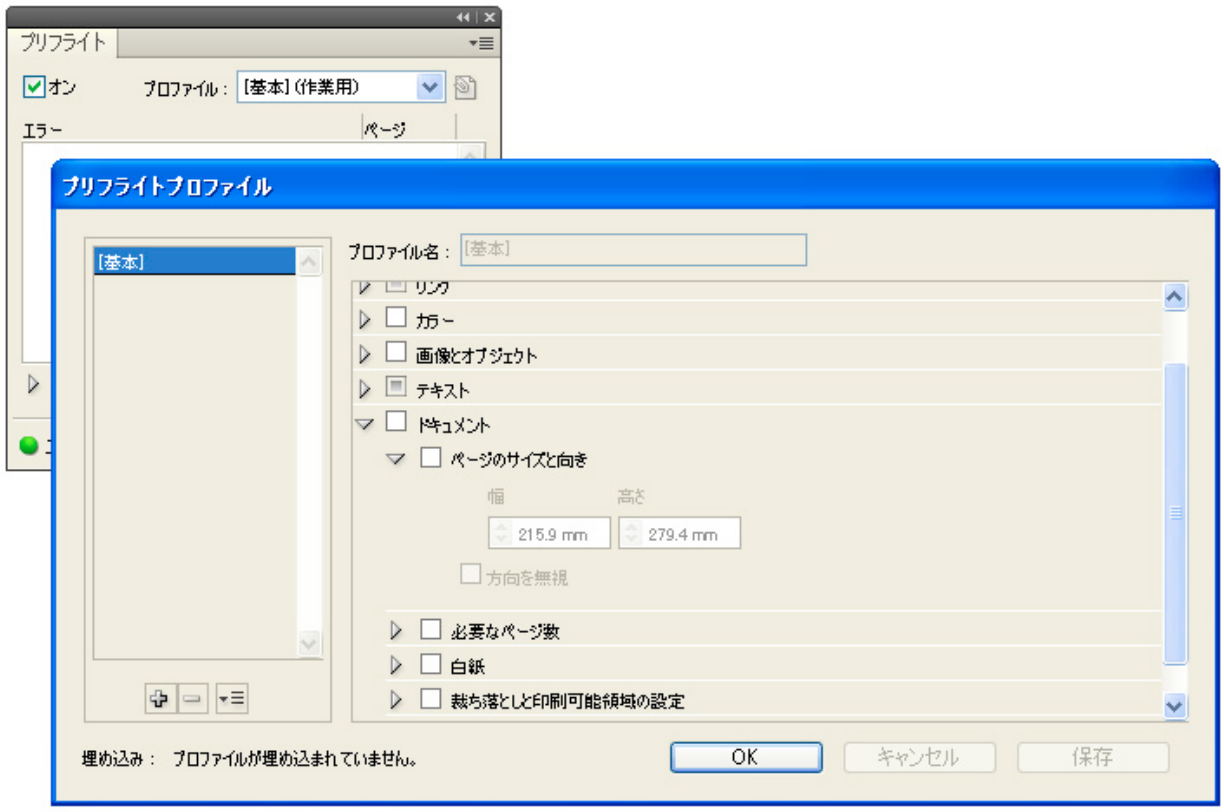
プリフライトデータオブジェクトの一覧表示

プロファイルのルール内のすべてのプリフライトデータオブジェクトを一覧表示する方法を、次のスクリプトに示します。完全なスクリプトについては、ListPreflightDataObjects を参照してください。

```
tell application "Adobe InDesign CS6"
    set myProfile to item 1 of preflight profiles
    set myRule to item 1 of preflight profile rules of myProfile
    set dataObjects to rule data objects of myRule
    set dataObjectCount to count of dataObjects
    set str to"Preflight rule data objects of " & name of myProfile &"." &name of myRule &": "
    repeat with i from 1 to dataObjectCount
        if i > 1 then
            set str to str & "; "
        end if
        set myObject to item i of dataObjects
        set str to str & name of myObject & ", "
        set str to str & data type of myObject & ", "
        set str to str & data value of myObject
    end repeat
    display alert str
end tell
```

プリフライトプロファイルの読み込み

プリフライトパネルからプリフライトプロファイルを読み込むには、プリフライトパネル／プロファイルを定義を選択します。続いて、プリフライトプロファイルウィンドウのドロップダウンメニューで「プロファイルを読み込み」を選択します。



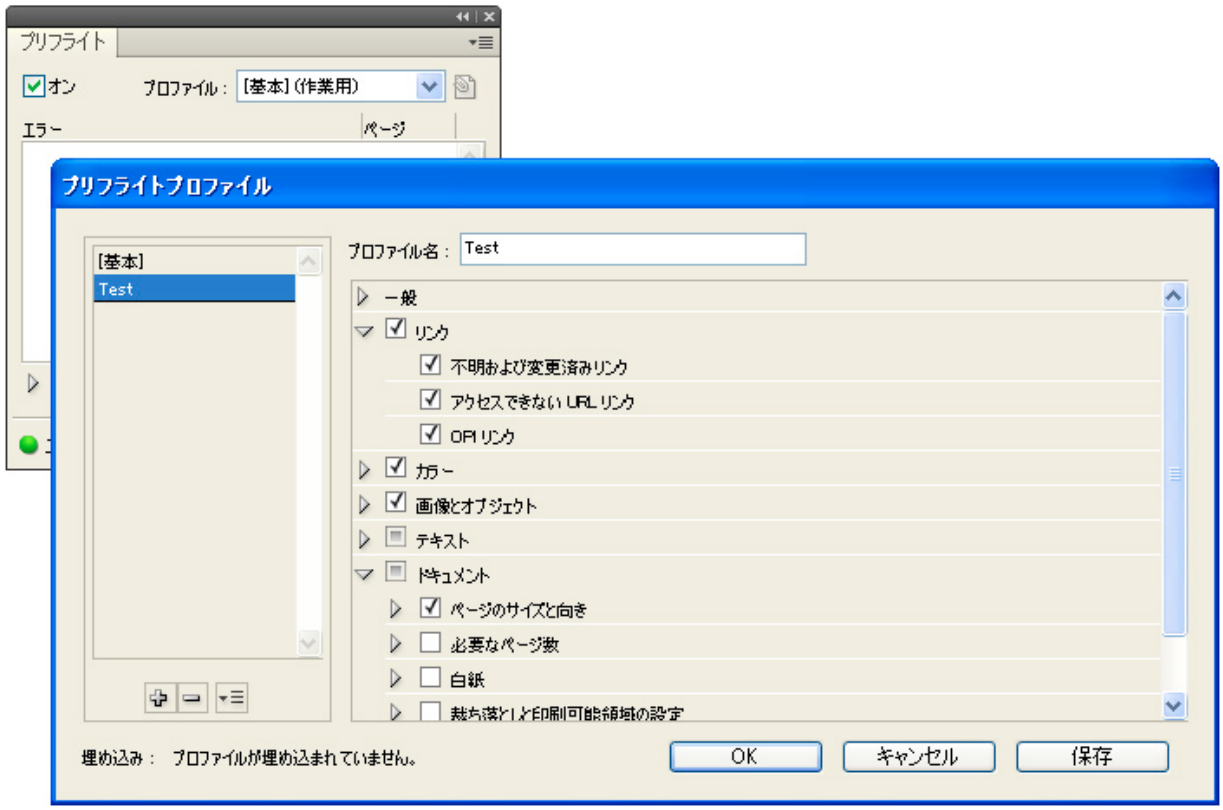
また、スクリプトでプロファイルを読み込むこともできます。次のスクリプトでは、Test という名前のプロファイルを読み込みます。完全なスクリプトについては、ImportPreflightProfile を参照してください。

```
set myProfile to load preflight profile from "Macintosh HD:tmp:Test.idpp"
if myProfile is nothing then
    display alert "The profile did not load successfully"
else
    display alert "Preflight profile " & (name of myProfile) & " is loaded."
end if
```

プロファイルの作成には、スクリプトよりもプリフライトパネルを使用したほうが簡単です。1 つのワークフローとしては、ユーザーインターフェイスですべてのプロファイルを作成し、プロファイルをファイルに書き出し、書き出したファイルをスクリプトで読み込むといった流れが考えられます。この方法を使用すると、スクリプトを使用して手動でルールを追加する手間を省くことできます。

プリフライトプロファイルの作成

プリフライトパネルからプリフライトプロファイルを作成するには、プリフライトパネル／プロファイルを定義を選択し、プラス記号（+）を選択して新しいプリフライトプロファイルを追加します。プロファイルに名前を付け、使用可能なルールのデータ値をすべて入力します。



また、スクリプトでプロファイルを作成することもできます。次のスクリプトでは、Test という名前のプロファイルを 1 つ追加します。完全なスクリプトについては、CreatePreflightProfile を参照してください。

```
--Add a new preflight profile.
set myProfile to make preflight profile
display alert "Preflight profile " & (name of myProfile) & " is created."
--Rename the profile
tell myProfile
    set oldName to name
    set name to "Test"
    set description to "Test description"
end tell
display alert "Profile " & oldName & " is renamed to " & (name of myProfile) & "."
```

プリフライトプロファイルの名前は、一意である必要があります。このスクリプトを同じインスタンスの InDesign で 2 回以上実行すると、エラーが発生し、該当する名前のプロファイルは既に存在していると通知されます。この問題を回避するには、`app.preflightProfiles.itemByName()` を使用して既存のプロファイルにアクセスするか、プロファイルが存在するかどうかを確認して削除します。次のスクリプトにその例を示します。完全なスクリプトについては、DeletePreflightProfile を参照してください。

```
on removeProfile(profileName)
    tell application "Adobe InDesign CS6"
        set myProfiles to preflight profiles
        set profileCount to count of myProfiles
        repeat with i from 1 to profileCount
            if name of item i of myProfiles is equal to profileName then
                delete item i of myProfiles
            end if
        end repeat
    end tell
end removeProfile
```

ルールの追加

プリフライトプロファイルには、相互排他的なルール一式が含まれます。プロファイルにルールを追加するには、次の手順に従います。

1. プロファイルに、名前による指定方式でルールを追加します。

   名前による指定方式でルールが追加されます。ルール名について詳しくは、159 ページの「使用可能なルール」を参照してください。次のスクリプトでは、プロファイルに ADBE_PageSizeOrientation のルールを追加しています。

   ```
   --Add a rule that requires a specific page size and orientation
   --(portrait or landscape).
   set RULE_NAME to "ADBE_PageSizeOrientation"
   set myRule to make preflight profile rule in myProfile with properties {name:RULE_NAME,
   id:RULE_NAME}
   ```

2. ルールのデータ値を設定します。

   すべてではありませんが、多くのルールにはデータプロパティが含まれています。InDesign で使用可能なルールの完全な仕様について詳しくは、159 ページの「使用可能なルール」を参照してください。ADBE_PageSizeOrientation のルールには、固有のデータプロパティが含まれており、このプロパティではページサイズを指定できます。次のスクリプトでは、許可するページの高さと幅、許容範囲（誤差）、ページ方向の処理オプションを設定しています。

   ```
   tell myRule
       --Requires the page size to be 8.5in x 11in (Letter Size)
       --enters a value for tolerance
       make rule data object with properties {name:"tolerance", data type:real data
   type, data value:0.01}
       --Sets the width  to the point equivalent of 8.5 inches
       make rule data object with properties {name:"width", data type:real data type, data
   value:612}
       --Sets the width  to the point equivalent of 11 inches
       make rule data object with properties {name:"height", data type:real data type, data
   value:792}
       --true = ignore orientation is checked
       make rule data object with properties {name:"ignore_orientation", data type:boolean data
   type, data value:true}
   end tell
   ```

3. ルールのレポート状態を設定します。

   この設定を行うには、ルールの `flag` プロパティを使用します。いくつかの選択肢（無効、通知、警告、エラー）があり、この選択肢は `PreflightRuleFlag` 列挙法で制御されます。

   ```
   --set the rule to return an error
   set flag of myRule to return as error
   ```

プロファイルの処理

デスクトップ版の InDesign では、プリフライトエラーはユーザーインターフェイスでレポートされます。（特に InDesign Server の）スクリプトでは、エラーは必要に応じて生成されます。次のスクリプトでは、InDesign ドキュメントに対する処理を行います （完全なスクリプトについては、ProcessPreflightProfile を参照してください）。エラーが発生した場合は、新しい PDF ファイルにエラー結果が書き込まれます。出力結果の例については、スクリプトの後のスクリーンショットを参照してください。

```
--Assume there is an document.
set myDoc to document 1
--Use the second preflight profile
set myProfile to preflight profile 2

--Process the doc with the profile
set myProcess to make preflight process with properties {target object:myDoc, applied
profile:myProfile}
wait for process myProcess
set results to process results of myProcess

-- If errors were found
if results is not "None" then
    -- Export the file to PDF, and to open the file after export.
    save report myProcess to "Macintosh HD:tmp:PreflightResults.pdf" with auto open
end if

--Cleanup
delete myProcess
```

テキストファイルでの作成が望ましい場合は、単純に、出力ファイルの名前に `.txt` の拡張子を付けます。

また、エラーに対する繰り返し処理をユーザー自身が行いたい場合も考えられます。エラー配列にアクセスする方法を、次のスクリプトに示します。完全なスクリプトについては、ProcessPreflightProfileShowErrors を参照してください。

```
-- If errors were found
if results is not "None" then
    --array containing detailed results
    tell aggregated results of myProcess
        set str to "Document: " & item 1 & ", Profile: " & item 2 & ", Results: ["
        set errorResults to item 3
    end tell
    --Show the errors in a message box.
    repeat with i from 1 to count of errorResults
        if i > 1 then
            str = str & ", "
        end if
        set str to str & item 2 of item i of errorResults
    end repeat
    set str to str & "]"
    display alert str
end if
```

カスタムルール

プリフライトパネルやスクリプトからはカスタムルールを作成できません。ただし、C++ プラグインを使用すると、カスタムルールを作成することができます。InDesign Products SDK にはサンプルの PreflightRule が含まれおり、この中に、プラグインを使用してカスタムルールを追加する方法が示されています。

使用可能なルール

スクリプトでルールを使用する際に苦労する点は、ルールの名前とプロパティを把握しておくことです。ルールの動的な性質上（ルールは実際には単なる文字列であるため）、ルールの具体的な名前とプロパティは ExtendScript Toolkit のオブジェクトモデルビューアに表示されません。これらの情報を確認する方法について詳しくは、153 ページの「プリフライトプロファイルの調査」を参照してください。ご参考までに、SDK には `DumpPreflightRules.jsx` スクリプトが用意されています。このスクリプトは、次のような出力の HTML ファイルを作成します（*SDK*¥docs¥references¥PreflightRules.html）。カスタムルールを追加するプラグインを使用している場合、スクリプトを実行して新しい名前とプロパティを取得できます。

ルール名	ルールのプロパティ
`ADBE_BlankPages`	160 ページの「ADBE_BlankPages」
`ADBE_BleedSlug`	160 ページの「ADBE_BleedSlug」
`ADBE_BleedTrimHazard`	161 ページの「ADBE_BleedTrimHazard」
`ADBE_CMYPlates`	なし
`ADBE_Colorspace`	162 ページの「ADBE_Colorspace」
`ADBE_ConditionIndicators`	なし
`ADBE_CrossReferences`	162 ページの「ADBE_CrossReferences」
`ADBE_FontUsage`	162 ページの「ADBE_FontUsage」
`ADBE_ImageColorManagement`	162 ページの「ADBE_ImageColorManagement」
`ADBE_ImageResolution`	163 ページの「ADBE_ImageResolution」
`ADBE_InteractiveContent`	なし
`ADBE_LayerVisibility`	なし

ルール名	ルールのプロパティ
ADBE_MissingFonts	なし
ADBE_MissingGlyph	なし
ADBE_MissingModifiedGraphics	なし
ADBE_OPI	なし
ADBE_Overprint	なし
ADBE_OversetText	なし
ADBE_PageCount	163 ページの「ADBE_PageCount」
ADBE_PageSizeOrientation	163 ページの「ADBE_PageSizeOrientation」
ADBE_Registration	なし
ADBE_ScaledGraphics	164 ページの「ADBE_ScaledGraphics」
ADBE_ScaledType	164 ページの「ADBE_ScaledType」
ADBE_SmallText	164 ページの「ADBE_SmallText」
ADBE_SpellCheck	なし
ADBE_SpotColorSetup	164 ページの「ADBE_SpotColorSetup」
ADBE_StrokeRequirements	164 ページの「ADBE_StrokeRequirements」
ADBE_TextOverrides	165 ページの「ADBE_TextOverrides」
ADBE_TransparencyBlending	165 ページの「ADBE_TransparencyBlending」
ADBE_TransparencyUsage	なし
ADBE_WhiteOverprint	なし

ADBE_BlankPages

データ型	名前	デフォルト値
Boolean	ignore_master	true
Boolean	ignore_nonprinting	true

ADBE_BleedSlug

データ型	名前	デフォルト値
Real	bleed_b	9
Real	bleed_b_aux	9
Integer	bleed_comparison_type	3
Boolean	bleed_enabled	true

データ型	名前	デフォルト値
Real	bleed_l	9
Real	bleed_l_aux	9
Real	bleed_r	9
Real	bleed_r_aux	9
Real	bleed_t	9
Real	bleed_t_aux	9
Real	slug_b	18
Real	slug_b_aux	18
Integer	slug_comparison_type	3
Boolean	slug_enabled	false
Real	slug_l	18
Real	slug_l_aux	18
Real	slug_r	18
Real	slug_r_aux	18
Real	slug_t	18
Real	slug_t_aux	18
Real	tolerance	0.01

ADBE_BleedTrimHazard

データ型	名前	デフォルト値
Boolean	binding_enabled	false
Real	binding_width	1
Real	live_b	18
Real	live_l	18
Real	live_r	18
Real	live_t	18
Real	tolerance	0.01

ADBE_Colorspace

データ型	名前	デフォルト値
Boolean	no_cmyk	false
Boolean	no_gray	false
Boolean	no_lab	false
Boolean	no_rgb	false
Boolean	no_spot	false

ADBE_CrossReferences

データ型	名前	デフォルト値
Boolean	xrefs_out_of_date	true
Boolean	xrefs_unresolved	true

ADBE_FontUsage

データ型	名前	デフォルト値
Boolean	no_ATC	false
Boolean	no_Bitmap	false
Boolean	no_CID	false
Boolean	no_MultipleMaster	false
Boolean	no_OpenTypeCFF	false
Boolean	no_OpenTypeCID	false
Boolean	no_OpenTypeTT	false
Boolean	no_TrueType	false
Boolean	no_Type1	false
Boolean	no_protected	true

ADBE_ImageColorManagement

データ型	名前	デフォルト値
Boolean	no_cmyk_profiles	true
Boolean	no_image_overrides	true
Boolean	overrides_exclude_uncal	true

ADBE_ImageResolution

データ型	名前	デフォルト値
Boolean	bw_max_enabled	false
Real	bw_max_res	2400
Boolean	bw_min_enabled	true
Real	bw_min_res	800
Boolean	color_max_enabled	false
Real	color_max_res	1200
Boolean	color_min_enabled	true
Real	color_min_res	250
Boolean	gray_max_enabled	false
Real	gray_max_res	1200
Boolean	gray_min_enabled	true
Real	gray_min_res	250
Real	tolerance	0.5

ADBE_PageCount

データ型	名前	デフォルト値
Integer	comparison_type	2
Integer	comparison_value	1
Integer	comparison_value_aux	1

ADBE_PageSizeOrientation

データ型	名前	デフォルト値
Real	height	792
Boolean	ignore_orientation	false
Real	tolerance	0.01
Real	width	612

ADBE_ScaledGraphics

データ型	名前	デフォルト値
Real	max_scale	100.5

ADBE_ScaledType

データ型	名前	デフォルト値
Boolean	ignore_justification	true
Real	max_scale	100.5

ADBE_SmallText

データ型	名前	デフォルト値
Real	minSize	4
Boolean	minSize_trap_safe_only	false

ADBE_SpotColorSetup

データ型	名前	デフォルト値
Boolean	lab_spots	true
Boolean	lab_spots_enabled	false
Integer	max_spots	1
Boolean	max_spots_enabled	true

ADBE_StrokeRequirements

データ型	名前	デフォルト値
Real	min_width	0.125
Boolean	min_width_trap_safe_only	false

ADBE_TextOverrides

データ型	名前	デフォルト値
Boolean	ignore_color_overrides	false
Boolean	ignore_font_overrides	false
Boolean	ignore_kerning_tracking_overrides	false
Boolean	ignore_language_overrides	false

ADBE_TransparencyBlending

データ型	名前	デフォルト値
Integer	space	3

11 ダイナミックドキュメントの作成

章の更新ステータス	
CS6	変更なし

InDesign では、Web やオンラインで使用するドキュメントを作成できます。これは、Rich Interactive Document（RID）とも呼びます。ダイナミックドキュメントには、サウンド、アニメーション、ハイパーリンクなどインタラクティブなコンテンツが含まれます。InDesign ドキュメントは、SWF や XFL、PDF に書き出すことができます。SWF および XFL ファイルの場合は、ドキュメントにアニメーション、ボタン、マルチステートオブジェクト、ムービー、サウンドクリップを含めることができます。InDesign のプレビューパネルを使用すると、書き出す前に、いくつかの種類のダイナミックコンテンツをテストすることができます。

この章では、スクリプトを使用してダイナミックドキュメントを作成する方法について説明します。PDF、SWF および XFL 形式で書き出す方法について詳しくは、ユーザーガイドの「ドキュメントを使用した作業」の章を参照してください。

ムービーとサウンドの読み込み

InDesign では、ムービーファイルやサウンドファイルを読み込み、書き出し後の PDF、SWF、XFL ドキュメントで表示したり再生したりすることができます。InDesign ドキュメント内のムービーとサウンドは、InDesign ページ上のコンテナオブジェクトに含まれるという点でグラフィックによく似ています。ただし、グラフィックとは異なり、読み込んだマルチメディアファイルのコンテンツは InDesign ページ上で表示（または再生）できません。ファイルを表示するには、プレビューパネルでページを表示するか、ファイルを書き出してコンテンツを表示可能なビューア（Adobe Reader や Web ブラウザーなど）でそのファイルを開く必要があります。

スクリプトを使用して、書き出し後のダイナミックドキュメント内のサウンドやムービーの再生プロパティを制御できます。また、サウンドやムービーが含まれているページアイテムに、プレビュー画像（「ポスター」画像）を追加することもできます。

ムービーを読み込み、書き出し後のドキュメントのムービーの表示と再生仕様を制御する方法を、次のスクリプトに示します（完全なスクリプトについては、PlaceMovie を参照してください）。

```
--Given a page "myPage"...
tell myPage
    set myFrame to make rectangle with properties {geometric bounds:{72, 72, 288, 288}}
end tell
--Import a movie file (you'll have to provide a valid file path on your system)
tell myFrame to place file "hazuki:movie.flv"
set myMovie to movie 1 of myFrame
--Set movie properties.
set embed in PDF of myMovie to true
set show controls of myMovie to true
get properties of myMovie
--Add a preview image. You'll have to provide a valid path on your system.
set poster file of myMovie to "hazuki:movie poster.jpg"
```

サウンドファイルを読み込み、書き出し後のドキュメントのサウンドの再生と表示を制御する方法を、次のスクリプトに示します（完全なスクリプトについては、PlaceSound を参照してください）。

```
--Given a page "myPage" in a document "myDocument..."
--Import a sound file (you'll have to provide a valid file path on your system)
tell myPage
    set mySound to place file "hazuki:sound.mp3" place point {72, 72}
end tell
set mySound to item 1 of mySound
tell mySound
    set embed in PDF to true
    set do not print poster to true
    set sound loop to true
    set stop on page turn to true
end tell
--Add a preview image. You'll have to provide a valid path on your system.
set poster file of mySound to "hazuki:sound poster.jpg"
```

ボタンを使用して、サウンドやムービーの再生を制御することもできます。スクリプトでボタンを操作する方法について詳しくは、次の節を参照してください。

ボタンの作成

ボタンは多くの場合、ダイナミックドキュメントでのナビゲーションに使用します。ボタンには、「通常」、「ロールオーバー」、「クリック」の 3 つのステートが含まれていますが、さらに、ページアイテム（長方形、楕円形、テキストフレーム、画像など）を含めることもできます。ボタンは一度に 1 つのステートしか表示できません。その他のステートは、マウスアクションによって呼び出されたときに表示されます。

動作は、ユーザーが特定のマウスアクションを実行したときに、ボタンがどのように反応するかを制御します。動作は、InDesign のユーザーインターフェイスのボタンパネルに表示されるアクションに対応します。ボタンには、複数の動作を含めることができます。

書き出し後の PDF または SWF で次のページを表示する簡単なボタンを作成する方法を、次のスクリプトに示します（完全なスクリプトについては、SimpleButton を参照してください）。このボタンでは、「通常」ステートのみを使用しています。

```
--Given a page "myPage" and a document containing the color "Red"...
--Make a button by converting a page item.
tell myPage
    set myRightArrow to make polygon with properties {fill color:"Red",
    name:"GoToNextPageButton"}
    set entire path of path 1 of myRightArrow to {{72, 72}, {144, 108}, {72, 144}}
    set myButton to make button with properties {geometric bounds:{72, 72, 144, 144}}
end tell
tell state 1 of myButton
    add items to state pageitems {myRightArrow}
end tell
tell myButton
    set myGoToNextPageBehavior to make goto next page behavior
    with properties {behavior event:mouse up}
end tell
```

3 種類の各ボタンステートの外観を変更するページアイテムが含まれた、もう少し複雑なボタンを作成する方法を、次のスクリプトに示します （完全なスクリプトについては、ButtonStates を参照してください）。

```
--Given a page "myPage" in a document "myDocument," containing the colors
--"Blue" and "Red"...
--Make a button "from scratch."
tell page 1 of myDocument
    set myButton to make button with properties {geometric bounds:{72, 72, 144, 144},
    name:"GoToNextPageButton"}
end tell
tell state 1 of myButton
    set myRightArrow to make polygon with properties {fill color:color "Red" of
    myDocument, stroke color:"None"}
    set entire path of path 1 of myRightArrow to {{72, 72}, {144, 108}, {72, 144}}
end tell
--Add the Rollover state.
tell myButton
    set myRolloverState to make state
end tell
tell myRolloverState
    set myRolloverArrow to make polygon with properties {fill color:color "Red"
    of myDocument, stroke color:"None"}
    set entire path of path 1 of myRolloverArrow to {{72, 72}, {144, 108}, {72, 144}}
    --Add a shadow to the polygon in the Rollover state.
end tell
tell drop shadow settings of fill transparency settings of myRolloverArrow
    set mode to drop
    set angle to 90
    set x offset to 0
    set y offset to 0
    set size to 6
end tell
tell myButton
    set myClickState to make state
end tell
tell myClickState
    set myClickArrow to make polygon with properties {fill color:color "Blue"
    of myDocument, stroke color:"None"}
    set entire path of path 1 of myClickArrow to {{72, 72}, {144, 108}, {72, 144}}
end tell
--Set the behavior for the button.
tell myButton
    set myGoToNextPageBehavior to make goto next page behavior
    with properties {behavior event:mouse up}
end tell
```

ボタンを使用して、ムービーファイルやサウンドファイルの再生を制御することもできます。ムービーファイルの再生を制御するボタン一式を使用した例を、次のスクリプトに示します（完全なスクリプトについては、MovieControl を参照してください）。

```
--Given a page "myPage" in a document "myDocument,"
--containing the colors "Gray" and "Red"...
tell myPage
    set myFrame to make rectangle with properties {geometric bounds:{72, 72, 288, 288}}
    --Import a movie file (you'll have to provide a valid file path on your system)
    tell myFrame to place file "hazuki:movie.flv"
    --Create the movie "Start" button.
    set myPlayButton to make button with properties {geometric bounds:
    {294, 186, 354, 282}, name:"PlayMovieButton"}
    tell myPlayButton
        set myRightArrow to make polygon with properties {fill color:color "Gray"
        of myDocument, stroke color:"None"}
    end tell
    set entire path of path 1 of myRightArrow to {{186, 294}, {186, 354}, {282, 324}}
    --Add the Rollover state.
    tell myPlayButton
        set myRolloverState to make state
    end tell
    --Add a shadow to the polygon in the Rollover state.
```

```
    tell myRolloverState
        set myRolloverArrow to make polygon with properties {fill color:color "Gray"
        of myDocument, stroke color:"None"}
    end tell
    set entire path of path 1 of myRolloverArrow to {{186, 294}, {186, 354}, {282, 324}}
    tell drop shadow settings of fill transparency settings of myRolloverArrow
        set mode to drop
        set angle to 90
        set x offset to 0
        set y offset to 0
        set size to 6
    end tell
    tell myPlayButton
        set myClickState to make state
    end tell
    tell myClickState
        set myClickArrow to make polygon with properties {fill color:color "Red"
        of myDocument, stroke color:"None"}
    end tell
    set entire path of path 1 of myClickArrow to {{186, 294}, {186, 354}, {282, 324}}
    --Set the behavior for the button.
    tell myPlayButton
        set myMovieStartBehavior to make movie behavior with properties {movie item:
        movie 1 of myFrame, behavior event:mouse up, operation:play}
    end tell
    --Create the movie "Stop" button.
    set myStopButton to make button with properties {geometric bounds:
    {294, 78, 354, 174}, name:"StopMovieButton"}
    tell state 1 of myStopButton
        set myNormalRectangle to make rectangle with properties {geometric bounds:
        {294, 78, 354, 174}, fill color:color "Gray" of myDocument}
    end tell
    tell myStopButton
        set myRolloverState to make state
    end tell
    tell myRolloverState
        set myRolloverRectangle to make rectangle with properties {geometric bounds:
        {294, 78, 354, 174}, fill color:color "Gray" of myDocument}
    end tell
    tell drop shadow settings of fill transparency settings of myRolloverRectangle
        set mode to drop
        set angle to 90
        set x offset to 0
        set y offset to 0
        set size to 6
    end tell
    tell myStopButton
        set myClickState to make state
    end tell
    tell myClickState
        set myClickRectangle to make rectangle with properties {geometric bounds:
        {294, 78, 354, 174}, fill color:color "Red" of myDocument}
    end tell
    tell myStopButton
        set myMovieStopBehavior to make movie behavior with properties {movie item:
        movie 1 of myFrame, behavior event:mouse up, operation:stop}
    end tell
end tell
```

また、ボタンはマルチステートオブジェクトの外観を制御する場合に重要です。この点については、次の節で説明します。

マルチステートオブジェクトの作成

マルチステートオブジェクト（MSO）は、複数のステートを含んでいる点と、一度に 1 つのステートしか表示できない点がボタンに似ています。ボタンと異なる点は、マルチステートオブジェクトにはいくつでもステートを含めることができることです。ボタンには最大 3 つまでしかステートを含めることができません。マルチステートオブジェクトは、ステートの表示方法を変更する場合はボタンに依存します。

簡単なマルチステートオブジェクトを作成し、このオブジェクトのステート表示を制御するボタンを追加する方法を、次のスクリプトに示します（完全なスクリプトについては、MakeMultiStateObject を参照してください）。

```
--Given a document "myDocument" and a page "myPage" and
--four colors "myColorA," "myColorB," "myColorC," and "myColorD"...
tell myPage
    set myMSO to make multi state object with properties {name:"Spinner",
    geometric bounds:{72, 72, 144, 144}}
    --New multistate objects contain two states when they're created. Add two more.
    tell myMSO
        set name of state 1 to "Up"
        set name of state 2 to "Right"
        make state with properties {name:"Down"}
        make state with properties {name:"Left"}
    end tell
    --Add page items to the states.
    tell state 1 of myMSO
        set myPolygon to make polygon with properties {fill color:myColorA,
        stroke color:"None"}
        set entire path of path 1 of myPolygon to {{72, 144}, {144, 144}, {108, 72}}
    end tell
    tell state 2 of myMSO
        set myPolygon to make polygon with properties {fill color:myColorB,
        stroke color:"None"}
        set entire path of path 1 of myPolygon to {{72, 72}, {72, 144}, {144, 108}}
    end tell
    tell state 3 of myMSO
        set myPolygon to make polygon with properties {fill color:myColorC,
        stroke color:"None"}
        set entire path of path 1 of myPolygon to {{72, 72}, {108, 144}, {144, 72}}
    end tell
    tell state 4 of myMSO
        set myPolygon to make polygon with properties {fill color:myColorD,
        stroke color:"None"}
        set entire path of path 1 of myPolygon to {{144, 72}, {72, 108}, {144, 144}}
    end tell
end tell
```

一般的に、マルチステートオブジェクトのステート表示を制御するには、ボタンを使用します。次のスクリプトにその例を示します（完全なスクリプトについては、MultiStateObjectControl を参照してください）。

```
--Given a document "myDocument" and a page "myPage" and
--four colors "myColorA," "myColorB," "myColorC," and "myColorD"...
tell myPage
    set myMSO to make multi state object with properties {name:"Spinner",
    geometric bounds:{72, 72, 144, 144}}
    --New multistate objects contain two states when they're created. Add two more.
    tell myMSO
        set name of state 1 to "Up"
        set name of state 2 to "Right"
        make state with properties {name:"Down"}
        make state with properties {name:"Left"}
    end tell
    --Add page items to the states.
    tell state 1 of myMSO
        set myPolygon to make polygon with properties {fill color:myColorA,
```

```
        stroke color:"None"}
        set entire path of path 1 of myPolygon to {{72, 144}, {144, 144}, {108, 72}}
    end tell
    tell state 2 of myMSO
        set myPolygon to make polygon with properties {fill color:myColorB,
        stroke color:"None"}
        set entire path of path 1 of myPolygon to {{72, 72}, {72, 144}, {144, 108}}
    end tell
    tell state 3 of myMSO
        set myPolygon to make polygon with properties {fill color:myColorC,
        stroke color:"None"}
        set entire path of path 1 of myPolygon to {{72, 72}, {108, 144}, {144, 72}}
    end tell
    tell state 4 of myMSO
        set myPolygon to make polygon with properties {fill color:myColorD,
        stroke color:"None"}
        set entire path of path 1 of myPolygon to {{144, 72}, {72, 108}, {144, 144}}
    end tell
    set myButton to make button with properties {geometric bounds:{72, 72, 144, 144}}
    tell myButton
        set myRolloverState to make state
        set myClickState to make state
        set myNextStateBehavior to make goto next state behavior with
        properties {associated multi state object:myMSO, behavior event:mouse down,
        enable behavior:true, loops to next or previous:true}
    end tell
    tell myRolloverState
        set myRolloverRectangle to rectangle 1 of group 1
    end tell
    set stroke color of myRolloverRectangle to myColorD
    set stroke weight of myRolloverRectangle to 1
    set myStrokeTransparencySettings to stroke transparency settings of
    myRolloverRectangle
    set myDropShadowSettings to drop shadow settings of myStrokeTransparencySettings
    tell myDropShadowSettings
        set mode to drop
        set angle to 90
        set x offset to 0
        set y offset to 0
        set size to 6
    end tell
end tell
```

アニメーションの操作

ページアイテムはアニメーション化することができます。アニメーション化するには、InDesign で作成したダイナミックドキュメントにモーションを追加します。オブジェクトにアニメーションを適用するにはモーションプリセットを、アニメーションオブジェクトの動作を定義するにはモーションパスを、アニメーションの長さを制御するにはタイミング設定、タイミングリスト、タイミンググループを使用します。

オブジェクトの `animation settings` は、オブジェクトに適用されるアニメーションを制御します。オブジェクトにアニメーション設定が適用されている場合、オブジェクトの `has custom settings` プロパティには true が設定されます。オブジェクトがアニメーション化されない場合は、false が設定されます。

アニメーションの再生が開始される位置は、アニメーションを呼び出すイベントを基準に、ページアイテムまたは親コンテナの 1 つ（通常はスプレッド）に割り当てられている `timing settings` オブジェクトのオブジェクトとプロパティによって制御されます。

基本的なアニメーション

簡単なアニメーションの作成方法を、次のスクリプトに示します（完全なスクリプトについては、SimpleAnimation を参照してください）。最も基本的な形式のアニメーションは、タイミング設定を使用せずに適用することができます。

```
--Given a document "myDocument" and a page "myPage" and a color "myColorA"...
--Add a page items to animate.
tell myPage
    set myPolygon to make polygon with properties {fill color:myColorA,
    stroke color:"None"}
end tell
set entire path of path 1 of myPolygon to {{72, 72}, {72, 144}, {144, 108}}
--Create a motion path.
set myMotionPathPoints to {{{{108, 108}, {108, 108}, {108, 108}}, {{516, 108},
{516, 108}, {516, 108}}}, true}
--Set animation preferences for the polygon. We havent' set a dynamic trigger
--for the animation, so the polygon's animation will be triggered by
--on page load (the default).
tell animation settings of myPolygon
    set duration to 2
    set motion path points to myMotionPathPoints
end tell
```

タイミング設定

スプレッド、ページおよびページアイテムの `timing settings` オブジェクトは、そのオブジェクトや、そのオブジェクトに含まれているすべてのオブジェクトに適用されるアニメーションのタイミングを制御します。`timing settings` には、次のプロパティが含まれます。

- `timing lists`。アニメーションを開始するトリガーイベント（ページの読み込みやページのクリックなど）を定義します。
- `timing groups`。1 つまたは複数のページアイテムを特定のタイミングに関連付けます。また、アニメーションが表示される順番を定義します。

`timing groups` には `timing targets` が含まれています。このプロパティは、特定の `timing group` に関連付けられているオブジェクトを定義します。また、`timing targets` は、ページアイテムに適用されるアニメーションの遅延値も指定します。この値は、`timing group` のアニメーションの開始に対する相対値で指定するか（`timing group` 内の最初のアイテムの場合）、`timing group` 内の前のアイテムの開始からの値で指定します（`timing group` 内のその他のアイテムの場合）。

様々なタイミングオブジェクトを使用して、オブジェクトのアニメーションのタイミングを制御する方法を、次のスクリプトに示します（完全なスクリプトについては、TimingSettings を参照してください）。`timing group` の作成に使用するパラメーターにより、`timing group` 内の最初の `timing target` のプロパティが指定されます。続いて、（存在する場合は）`timing targets` を個別に追加できます。

```
--Given a document "myDocument" and a page "myPage" and the color "myColorA",
--"myColorB", and "myColorC"...
--Add a page items to animate.
tell myPage
    set myPolygonA to make polygon with properties {fill color:myColorA,
    strokeColor:"None"}
    set entire path of path 1 of myPolygonA to {{72, 72}, {72, 144}, {144, 108}}
    set myPolygonB to make polygon with properties {fill color:myColorB,
    strokeColor:"None"}
    set entire path of path 1 of myPolygonB to {{72, 72}, {72, 144}, {144, 108}}
    set myPolygonC to make polygon with properties {fill color:myColorC,
    strokeColor:"None"}
    set entire path of path 1 of myPolygonC to {{72, 72}, {72, 144}, {144, 108}}
end tell
--Create a motion path.
```

```
set myMotionPathPoints to {{{{108, 108}, {108, 108}, {108, 108}}, {{516, 108},
{516, 108}, {516, 108}}}, true}
--Set animation preferences for the polygons.
tell animation settings of myPolygonA
    set duration to 2
    set motion path points to myMotionPathPoints
end tell
tell animation settings of myPolygonB
    set duration to 2
    set motion path points to myMotionPathPoints
end tell
tell animation settings of myPolygonC
    set duration to 2
    set motion path points to myMotionPathPoints
end tell
set myTimingSettings to timing settings of parent of myPage
--Remove the default timing list.
tell myTimingSettings
    delete timing list 1
    --Add a new timing list that triggers when the page is clicked.
    set myTimingList to make timing list with properties {trigger event:on page click}
end tell
--Add the polygons to a single timing group.
tell myTimingList
    set myTimingGroup to make timing group with properties {dynamic target:myPolygonA,
    delay seconds:0}
end tell
tell myTimingGroup
    make timing target with properties {dynamic target:myPolygonB, delay seconds:2}
    make timing target with properties {dynamic target:myPolygonC, delay seconds:2}
end tell
```

（ページアイテムの `animation settings` オブジェクトの）`has custom settings` プロパティが false のページアイテムを `timing target` に追加しようとすると、エラーが発生することに注意してください。

ページ上のオブジェクトに適用されるアニメーションの順番を制御する方法を、次のスクリプトに示します（完全なスクリプトについては、MultipleTimingGroups を参照してください）。`timing list` に `timing groups` が追加された順番により、`timing list` で指定されたトリガーイベントが発生したときのアニメーションの再生順が決まります。一部のトリガーイベント（`on page load` など）は、`timing list` のアニメーションを（順番に）呼び出しますが、それ以外のトリガーイベント（`on page click` など）は、イベントの各インスタンスで 1 つずつ（順番に）アニメーションを呼び出します。例えば、`on page click` のトリガーイベントが設定された `timing list` に 5 つの `timing groups` が含まれ、それぞれに 1 つの `timing target` が含まれている場合、すべてのアニメーションを処理するにはマウスを 5 回クリックする必要があります。

```
--Given a document "myDocument" and a page "myPage" and the color "myColorA",
--"myColorB", and "myColorC"...
--Add a page items to animate.
tell myPage
    set myPolygonA to make polygon with properties {fill color:myColorA,
    strokeColor:"None"}
    set entire path of path 1 of myPolygonA to {{72, 72}, {72, 144}, {144, 108}}
    set myPolygonB to make polygon with properties {fill color:myColorB,
    strokeColor:"None"}
    set entire path of path 1 of myPolygonB to {{72, 72}, {72, 144}, {144, 108}}
    set myPolygonC to make polygon with properties {fill color:myColorC,
    strokeColor:"None"}
    set entire path of path 1 of myPolygonC to {{72, 72}, {72, 144}, {144, 108}}
    set myPolygonD to make polygon with properties {fill color:myColorA,
    strokeColor:"None"}
    set entire path of path 1 of myPolygonD to {{72, 144}, {72, 216}, {144, 180}}
    set myPolygonE to make polygon with properties {fill color:myColorB,
    strokeColor:"None"}
    set entire path of path 1 of myPolygonE to {{72, 144}, {72, 216}, {144, 180}}
    set myPolygonF to make polygon with properties {fill color:myColorC,
    strokeColor:"None"}
```

```
    set entire path of path 1 of myPolygonF to {{72, 144}, {72, 216}, {144, 180}}
end tell
--Create a motion path.
set myMotionPathPointsA to {{{{108, 108}, {108, 108}, {108, 108}}, {{516, 108},
{516, 108}, {516, 108}}}, true}
set myMotionPathPointsB to {{{{108, 180}, {108, 180}, {108, 180}}, {{516, 180},
{516, 180}, {516, 180}}}, true}
--Set animation preferences for the polygons.
--DynamicTriggerEvents.onPageLoad (the default).
set duration of animation settings of myPolygonA to 2
set motion path points of animation settings of myPolygonA to myMotionPathPointsA
set duration of animation settings of myPolygonB to 2
set motion path points of animation settings of myPolygonB to myMotionPathPointsA
set duration of animation settings of myPolygonC to 2
set motion path points of animation settings of myPolygonC to myMotionPathPointsA
set duration of animation settings of myPolygonD to 2
set motion path points of animation settings of myPolygonD to myMotionPathPointsB
set duration of animation settings of myPolygonE to 2
set motion path points of animation settings of myPolygonE to myMotionPathPointsB
set duration of animation settings of myPolygonF to 2
set motion path points of animation settings of myPolygonF to myMotionPathPointsB
set myTimingSettings to timing settings of parent of myPage
--Remove the default timing list.
tell myTimingSettings
    delete timing list 1
    --Add a new timing list that triggers when the page is clicked.
    set myTimingList to make timing list with properties {trigger event:on page click}
    --Add the polygons to a single timing group.
    tell myTimingList
        set myTimingGroupA to make timing group with properties
        {dynamic target:myPolygonA, delay seconds:0}
        tell myTimingGroupA
            make timing target with properties {dynamic target:myPolygonB,
            delay seconds:2}
            make timing target with properties {dynamic target:myPolygonC,
            delay seconds:2}
        end tell
        --myTimingGroupB will play on the second page click.
        set myTimingGroupB to make timing group with properties {dynamic
        target:myPolygonD, delay seconds:0}
        tell myTimingGroupB
            make timing target with properties {dynamic target:myPolygonE,
            delay seconds:2}
            make timing target with properties {dynamic target:myPolygonF,
            delay seconds:2}
        end tell
    end tell
end tell
```

任意の `timing settings` オブジェクトに複数の `timing list` オブジェクトを含めて、それぞれのオブジェクトで別々のトリガーイベントに応答することができます。次のスクリプトでは、`on page load` および `on page click` によって呼び出された一連のアニメーションを表示します（完全なスクリプトについては、MultipleTimingLists を参照してください）。

```
set myTimingSettings to timing settings of parent of myPage
tell myTimingSettings
    --At this point, all of the polygons have already been added as timing targets
    --of the default timing list. Change the delay of myPolygonB and myPolygonC,
    --which are the targets of the second and third timing groups.
    set myTimingListA to timing list 1
    tell myTimingListA
        tell timing group 2
            set delay seconds of timing target 1 to 2
        end tell
        tell timing group 3
            set delay seconds of timing target 1 to 2
        end tell
        --Remove the last three timing groups in the timing list.
        --We have to do this, because we don't want these polygons to be
        --animated when the page loads.
        delete timing group -1
        delete timing group -1
        delete timing group -1
    end tell
    --Add a new timing list that triggers when the page is clicked.
    set myTimingListB to make timing list with properties {trigger event:on page click}
    tell myTimingListB
        set myTimingGroupB to make timing group with properties {dynamic
        target:myPolygonD, delay seconds:0}
        tell myTimingGroupB
            make timing target with properties {dynamic target:myPolygonE,
            delay seconds:2}
            make timing target with properties {dynamic target:myPolygonF,
            delay seconds:2}
        end tell
    end tell
end tell
```

前述の例では、アニメーション化したいページアイテムが含まれているスプレッドの `timing settings` を操作しました。ユーザーがクリックしたときにページアイテムをアニメーション化したい場合は、ページアイテム自体の `timing settings` を使用する必要があります。次のスクリプトにその例を示します（完全なスクリプトについては、PageItemTimingSettings を参照してください）。

```
--Given a page "myPage"...
tell myPage
    set myPolygonA to make polygon with properties {fill color:myColorA,
    stroke color:"None"}
    set entire path of path 1 of myPolygonA to {{72, 72}, {72, 144}, {144, 108}}
    --Create a motion path.
    set myMotionPathPointsA to {{{{108, 108}, {108, 108}, {108, 108}}, {{516, 108},
    {516, 108}, {516, 108}}}, true}
    --Set animation preferences for the polygon.
    tell animation settings of myPolygonA
        set duration to 2
        set motion path points to myMotionPathPointsA
    end tell
    --Remove the default timing list in the timing settings for the spread.
    set myTimingSettings to timing settings of parent of myPage
    tell myTimingSettings
        delete timing list 1
    end tell
    set myTimingSettings to timing settings of myPolygonA
    tell myTimingSettings
        set myTimingList to make timing list with properties {trigger event:on click}
    end tell
    tell myTimingList
        set myTimingGroup to make timing group with properties {dynamic
        target:myPolygonA, delay seconds:0}
    end tell
end tell
```

変形のアニメーション化

ページアイテムのアニメーションを再生するときに、ページアイテムのサイズ、回転または角度の歪み、不透明度および表示／非表示を変更できます。ページアイテムの `animation settings` にはプロパティ（`rotation array`、`hidden after` など）が含まれていて、このプロパティでアニメーション中に適用される変形を定義できます。モーションパスに従ってページアイテムを回転する方法を、次のスクリプトに示します（完全なスクリプトについては、AnimateRotation を参照してください）。

```
--Given a document "myDocument" and a page "myPage" and the color "myColorA"...
--Add a page items to animate.
tell page 1
    set myPolygon to make polygon with properties {fill color:myColorA,
    stroke color:"None"}
end tell
--Create a motion path.
set entire path of path 1 of myPolygon to {{72, 72}, {72, 144}, {144, 108}}
set myMotionPathPoints to {{{{108, 108}, {108, 108}, {108, 108}}, {{516, 108},
{516, 108}, {516, 108}}}, true}
--Set animation preferences for the polygon.
tell animation settings of myPolygon
    set duration to 2
    set motion path points to myMotionPathPoints
    --Assuming 24 Frames Per Second (FPS)
    --23 = 1 second, 47 = 2 seconds, 71 = 3 seconds, 95 = 4 seconds,
    --119 = 5 seconds, 143 = 6 seconds
    --Since the duration of our animation is 2 seconds, the following line will
    --make the polygon rotate 360 degrees from the start to the end
    --of the animation.
    set rotation array to {{0, 0}, {47, 360}}
end tell
set myTimingSettings to timing settings of parent of myPage
tell myTimingSettings
    --Remove the default timing list.
    delete timing list 1
    --Add a new timing list that triggers when the page is clicked.
    set myTimingList to make timing list with properties {trigger event:on page click}
    tell myTimingList
        --Add the polygon to a single timing group.
        make timing group with properties {dynamic target:myPolygon, delay seconds:0}
    end tell
end tell
```

スクリプトを使用すると、InDesign のユーザーインターフェイスを使用した場合よりも、アニメーションを細かく制御できます。例えば、スクリプトを使用したアニメーションの場合、任意のモーションパスの各キーフレームで変形を適用できます。詳しくは、この章で後述する「キーフレーム」を参照してください。

モーションプリセット

前述の例では、InDesign のユーザーインターフェイスで基本レベルからアニメーションを作成するように、モーションパスを構成してアニメーション設定を行いました。しかし、InDesign では、モーションプリセットを使用して、レイアウト内のページアイテムのアニメーションを定義することもできます。モーションプリセットは、一度に複数のアニメーションのプロパティに適用できます。次のスクリプトにその例を示します（完全なスクリプトについては、MotionPreset を参照してください）。InDesign には、多くのモーションプリセットが付属しています。また、ユーザーインターフェイスまたはスクリプトを使用して、新しいプリセットを追加することもできます。

```
--Given a page containing the oval "myOvalA"...
set myMotionPreset to Motion Preset "move-right-grow"
tell animation settings of myOvalA
    set duration to 2
    set plays loop to true
    set preset to myMotionPreset
end tell
```

デザインオプション

デザインオプションは、オブジェクトのアニメーション設定で指定したモーションに関連して、アニメーションオブジェクトの表示仕様に影響を与えます。次のスクリプトでは、アニメーションシェイプのデザインオプションが、アニメーションの再生にどのような影響を与えるかを示します（完全なスクリプトについては、DesignOptions を参照してください）。

```
--Given a page containing the ovals "myOvalA" and "myOvalB"...
set myMotionPreset to Motion Preset "move-right-grow"
tell animation settings of myOvalA
    set duration to 2
    set plays loop to true
    set preset to myMotionPreset
    set design option to from current appearance
end tell
tell animation settings of myOvalB
    set duration to 2
    set plays loop to true
    set preset to myMotionPreset
    set design option to to current appearance
end tell
```

キーフレーム

キーフレームは、アニメーションのタイムライン内における時点です。InDesign のスクリプティングでは、キーフレームをアニメーションの任意の時点に追加することで、オブジェクトがアニメーションするときに変更を適用できます。キーフレームは、アニメーションするページアイテムに適用されるモーションパスの一部で、アニメーションの長さと速度を基準にして指定します。例えば、長さ 2 秒のアニメーションの場合、1 秒間に 24 フレームで再生すると、アニメーションの最後のフレームはフレーム 48 です。

モーションパスにキーフレームを追加する方法と、アニメーション化するページアイテムに適用される変形を各キーフレームで変更する方法を、次のスクリプトに示します （完全なスクリプトについては、TransformAnimation を参照してください）。

```
--Given a page containing ovals "myOvalA," "myOvalB," and "myOvalC"...
--The motion path is constructed relative to the center of the object, and key frames
--are based on the duration of the animation divided by the number of frames per second
--(usually 24). The following array sets key frames at the start, midpoint, and end
--of a path.
set myMotionPath to {{0, {{0, 0}, {0, 0}, {0, 0}}}, {23, {{234, 0}, {234, 0},
{234, 0}}}, {47, {{468, 0}, {468, 0}, {468, 0}}}}
tell animation settings of myOvalA
    set duration to 2
    set motion path to myMotionPath
    --The transformation changes at each key frame.
    --scale x array in the form {{keyframe, scale_percentage}, {keyframe, scalePercentage}, ...}
    set scale x array to {{0, 100}, {23, 200}, {47, 100}}
    --scale y array in the form {{keyframe, scale_percentage}, {keyframe, scalePercentage}, ...}
    set scale y array to {{0, 100}, {23, 200}, {47, 100}}
    --opacity array in the form {{keyframe, opacity}, {keyframe, opacity},...}
    set opacity array to {{0, 100}, {23, 20}, {47, 100}}
    set plays loop to true
end tell
tell animation settings of myOvalB
```

```
    set duration to 2
    set motion path to myMotionPath
    set scale x array to {{0, 200}, {23, 300}, {47, 50}}
    set scale y array to {{0, 200}, {23, 300}, {47, 50}}
    set opacity array to {{0, 10}, {23, 80}, {47, 60}}
    set plays loop to true
end tell
tell animation settings of myOvalC
    set duration to 2
    set motion path to myMotionPath
    set scale x array to {{0, 50}, {23, 200}, {47, 400}}
    set scale y array to {{0, 50}, {23, 200}, {47, 400}}
    set opacity array to {{0, 100}, {23, 40}, {47, 80}}
    set plays loop to true
end tell
end tell
```

ページ効果の追加

ページ効果は、書き出し後のダイナミックドキュメントのページを変更した場合に現れる特殊効果です。ページ効果の追加は、スクリプトを使用して簡単に行えます。次のスクリプトにその例を示します（完全なスクリプトについては、PageTransitions を参照してください）。

```
--Given a document "myDocument" containing at least two spreads...
repeat with myCounter from 1 to (count spreads of myDocument)
    set page transition type of spread myCounter of myDocument to page turn transition
    --This page transition option does not support the page transition direction options
    --or page transition duration properties. If you chose wipe transition
    --(for example), you would be able to set those options, as shown in the next
    --two lines:
    --set page transition direction options of spread myCounter of myDocument
    --to left to right
    --set page transition duration of spread myCounter of myDocument to medium
end repeat
--Export the document to SWF, and you'll see the page transitions.
```

12 XML

章の更新ステータス	
CS6	変更なし

XML（Extensible Markup Language）は、World Wide Web Consortium（www.w3.org）で作成と管理が行われている、テキストベースのマークアップシステムです。XML では、HTML（Hypertext Markup Language）と同じように、不等号記号で囲まれたマークアップタグ（<article> や <para> など）が使用されます。HTML では、使用できるタグが事前に定義されていますが、XML では独自のタグを作成して、ドキュメントの構造をより詳細に記述することができます。

高い柔軟性を備えている XML は、データの保存形式として広く利用されています。InDesign には、XML データをページレイアウトに読み込むための様々な機能が用意されていますが、それらはスクリプトを使用して制御できます。

ここでは、読者が『Adobe InDesign スクリプティングチュートリアル』を既に読んでおり、スクリプトを作成して実行する方法を理解しているものとして説明を行います。また、XML、DTD、XSLT に関する基礎知識を持っていることを前提としています。

概要

XML はフォーマットを行うためのものではなく、ドキュメントの構造を記述するためのものです。したがって、ページレイアウト作業で XML を活用するには、やや複雑な手順が必要になります。InDesign は、XML に完全かつ柔軟に対応できますが、次のような制約があります。

- InDesign ドキュメントに読み込んだ XML 要素は、XML 構造に対応する InDesign 要素に変換されます。InDesign によって表現された XML 要素は、元の XML 要素とは異なります。
- 1 つの XML 要素は、レイアウト上で 1 つのみ使用できます。レイアウト上の XML 要素の情報を複製したい場合は、XML 要素そのものを複製する必要があります。
- XML 要素のレイアウト上での出現順序は、XML 構造での出現順序とほぼ同じになります。
- XML 要素に関連付けられているストーリー内のテキストは、その要素のデータになります。

InDesign での XML スクリプティングの推奨事項

XML ファイルの処理は、InDesign の外部で（InDesign のレイアウトに XML ファイルを読み込む前に）そのほとんどを行うのが一般的です。InDesign の外部では、XML エディターや XML パーサーなどの様々な便利なツールが利用できます。

大規模な XML データ構造に含まれている要素を再配置したり複製したりする場合は、XSLT を使用して XML を変換することを推奨します。これは、XML ファイルを読み込むときに実行できます。

InDesign ドキュメントで既にフォーマットされている XML データを操作する場合は、XML ルールを使用すると便利です。XML ルールを使用すると、効率よくドキュメント内の XML 構造を検索して、目的の XML 要素を処理できます。

XML ルールについて詳しくは、第 13 章「XML ルール」を参照してください。

XML 要素のスクリプティング

ここでは、XML 環境設定や XML 読み込み環境設定の指定、XML の読み込み、XML 要素の作成、XML 属性の追加を行う方法について説明します。ここでは、XML コンテンツ自体を操作するスクリプトについて説明します。XML 要素にフォーマットを適用するスクリプトについては、187 ページの「レイアウトへの XML 要素の追加」を参照してください。

XML 環境設定の指定

XML ビュー環境設定オブジェクトを使用して、InDesign の構造パネルの外観を制御できます。次のスクリプトにその例を示します（チュートリアルスクリプトの XMLViewPreferences より）。

```
tell application "Adobe InDesign CS6"
    set myDocument to make document
    tell myDocument
        tell XML view preferences
            set show attributes to true
            set show structure to true
            set show tagged frames to true
            set show tag markers to true
            set show text snippets to true
        end tell
    end tell
end tell
```

また、XML 環境設定オブジェクトを使用して、XML タグ付きプリセット環境設定（表やストーリーのデフォルトのタグ名やユーザーインターフェイスカラー）を指定することもできます。次のスクリプトにその例を示します（チュートリアルスクリプトの XMLPreferences より）。

```
tell application "Adobe InDesign CS6"
    set myDocument to make document
    tell myDocument
        tell XML preferences
            set default cell tag color to blue
            set default cell tag name to "cell"
            set default image tag color to brick red
            set default image tag name to "image"
            set default story tag color to charcoal
            set default story tag name to "text"
            set default table tag color to cute teal
            set default table tag name to "table"
        end tell
    end tell
end tell
```

XML 読み込み環境設定の指定

XML ファイルを読み込む前に、XML 読み込み環境設定を指定することができます。この環境設定を使用すれば、XSLT 変換を適用したり、XML ファイル内の空白文字の処理方法を制御したり、繰り返すテキスト要素を作成したりすることができます。XML 読み込み環境設定オブジェクトの設定方法を、次のスクリプトに示します（チュートリアルスクリプトの XMLImportPreferences より）。

```
tell application "Adobe InDesign CS6"
    set myDocument to make document
    tell myDocument
        tell XML import preferences
            set allow transform to false
            set create link to XML to false
            set ignore unmatched incoming to true
            set ignore whitespace to true
            set import CALS tables to true
            set import style to merge import
            set import text into tables to false
            set import to selected to false
            set remove unmatched existing to false
            set repeat text elements to true
            --The following properties are only used when the
            --allow transform property is set to true.
            --set transform filename to "yukino:myTransform.xsl"
            --If you have defined parameters in your XSL file,
            --you can pass them to the file during the XML import
            --process. For each parameter, enter a list containign two
            --strings. The first string is the name of the parameter,
            --the second is the value of the parameter.
            --set transform parameters to {{"format", "1"}}
        end tell
    end tell
end tell
```

XML の読み込み

XML 読み込み環境設定を指定したら、XML ファイルを読み込みます。次のスクリプトにその例を示します（チュートリアルスクリプトの ImportXML より）。

```
tell myDocument
    import XML from "yukino:xml_test.xml"
end tell
```

XML ファイルの内容を特定の XML 要素に読み込む必要がある場合は、ドキュメントではなく XML 要素の `importXML` メソッドを使用します。次のスクリプトにその例を示します（チュートリアルスクリプトの ImportXMLIntoElement より）。

```
set myRootElement to XML element 1
tell myRootElement
    import XML from "yukino:xml_test.xml"
end tell
--Place the root XML element so that you can see the result.
tell story 1
    place XML using myRootElement
end tell
```

または、`xml import preferences` オブジェクトの `import to selected` プロパティを true に設定し、XML 要素を選択した状態で XML ファイルを読み込む方法もあります。次のスクリプトにその例を示します（チュートリアルスクリプトの ImportXMLIntoSelectedElement より）。

```
tell XML import preferences
    set import to selected to true
end tell
select myXMLElement
import XML from "yukino:xml_test.xml"
```

XML タグの作成

XML タグとは、ドキュメント内で使用する XML 要素の名前のことです。XML を読み込むと、XML ファイル内の要素名が、ドキュメントの XML タグのリストに追加されます。また、XML タグを直接作成することもできます。次のスクリプトにその例を示します（チュートリアルスクリプトの MakeXMLTags より）。

```
tell application "Adobe InDesign CS6"
    set myDocument to make document
    tell myDocument
        --You can create an XML tag without specifying a color for the tag.
        set myXMLTagA to make XML tag with properties {name:"XML_tag_A"}
        --You can define the highlight oclor o fthe XML tag.
        set myXMLTagB to make XML tag with properties
        {name:"XML_tag_B", color:gray}
        --...or you can proved an RGB array to set the color of the tag.
        set myXMLTagC to make XML tag with properties
        {name:"XML_tag_C", color:{0, 92, 128}}
    end tell
end tell
```

XML タグのロード

XML ファイルの XML コンテンツを読み込まずに、XML タグだけを読み込むことができます。この機能は、XML データを読み込む前にタグとスタイルのマッピングを行いたい場合に便利です。次のスクリプトにその例を示します（チュートリアルスクリプトの LoadXMLTags より）。

```
tell myDocument
    load xml tags "yukino:test.xml"
end tell
```

XML タグの保存

ファイルから XML タグをロードするのとは逆に、XML タグをファイルに保存することもできます。これを行うと、タグだけが XML ファイルに保存されます。ドキュメントデータは保存されません。この処理は、XML を書き出すより高速で、生成されるファイルも小さくなります。XML タグを保存する方法を、次のサンプルスクリプトに示します（完全なスクリプトについては、SaveXMLTags を参照してください）。

```
tell myDocument
    save xml tags "yukino:xml_tags.xml"
    version comments "Tag set created October 5, 2006")
end tell
```

XML 要素の作成

通常、XML 要素は XML ファイルを読み込んで作成しますが、InDesign スクリプティングを使用して XML 要素を作成することもできます。次のスクリプトにその例を示します（チュートリアルスクリプトの CreateXMLElement より）。

```
tell application "Adobe InDesign CS6"
    set myDocument to make document
    tell myDocument
        set myXMLTag to make XML tag with properties {name:"myXMLTag"}
        set myRootElement to XML element 1
        tell myRootElement
            set myXMLElement to make XML element with properties
            {markup tag:myXMLTag}
        end tell
        set contents of myXMLElement to "This is an XML element containing text."
    end tell
end tell
```

XML 要素の移動

move メソッドを使用して、XML 構造内の XML 要素を移動できます。次のスクリプトにその例を示します（チュートリアルスクリプトの MoveXMLElement より）。

```
tell application "Adobe InDesign CS6"
    set myDocument to make document
    tell myDocument
        set myXMLTag to make XML tag with properties {name:"myXMLTag"}
        set myRootElement to XML element 1
        tell myRootElement
            set myXMLElementA to make XML element with properties
            {markup tag:myXMLTag}
            set contents of myXMLElementA to "This is XML element A."
            set myXMLElementB to make XML element with properties
            {markup tag:myXMLTag}
            set contents of myXMLElementB to "This is XML element B."
        end tell
        move myXMLElementA to after myXMLElementB
        end tell
    end tell
end tell
```

XML 要素の削除

XML 要素を削除すると、レイアウトと XML 構造の両方から XML 要素が削除されます。次のスクリプトにその例を示します（チュートリアルスクリプトの DeleteXMLElement より）。

```
tell xml element 1 of XML element 1 of myDocument to delete
```

XML 要素の複製

XML 要素を複製すると、新しい XML 要素が、XML 構造内の元の XML 要素の直後に追加されます。次のスクリプトにその例を示します（チュートリアルスクリプトの DuplicateXMLElement より）。

```
tell application "Adobe InDesign CS6"
    set myDocument to make document
    tell myDocument
        set myXMLTag to make XML tag with properties {name:"myXMLTag"}
        set myRootElement to XML element 1
        tell myRootElement
            set myXMLElementA to make XML element with properties
            {markup tag:myXMLTag}
            set contents of myXMLElementA to "This is XML element A."
            set myXMLElementB to make XML element with properties
            {markup tag:myXMLTag}
            set contents of myXMLElementB to "This is XML element B."
        end tell
        duplicate myXMLElementA
    end tell
end tell
```

XML 構造からのアイテムの削除

XML 要素とページアイテムやテキストとの関連付けを削除するには、untag メソッドを使用します。次のスクリプトにその例を示します。このメソッドを使用すると、XML 要素との関連付けは削除されますが、オブジェクトそのものは削除されません。XML 要素が削除されたコンテンツは、親の XML 要素に関連付けられます。XML 要素がルート XML 要素の場合、その XML 要素に関連付けられているレイアウトオブジェクト（テキストまたはページアイテム）は、ドキュメントに保持されます （完全なスクリプトについては、UntagElement を参照してください）。

```
set myXMLElement to XML element 1 of XML element 1 of myDocument
tell myXMLElement to untag
```

XML コメントの作成

XML コメントは、XML データ構造に注釈を作成する場合に使用します。XML コメントを追加する方法を、次のスクリプトに示します（チュートリアルスクリプトの MakeXMLComment より）。

```
tell application "Adobe InDesign CS6"
    set myDocument to make document
    tell myDocument
        set myXMLTag to make XML tag with properties {name:"myXMLElement"}
        set myRootXMLElement to XML element 1
        tell myRootXMLElement
            set myXMLElement to make XML element with properties
            {markup tag:myXMLTag}
            tell myXMLElement
                make XML comment with properties {value:"This is an XML comment."}
            end tell
        end tell
    end tell
end tell
```

XML 処理命令の作成

処理命令（PI：Processing Instruction）とは、XML ドキュメントを読み込むアプリケーションに対する命令が含まれている XML 要素のことです。XML 処理命令は、InDesign では無視されますが、他のアプリケーションで使用するために InDesign の XML 構造に挿入することができます。XML ドキュメントには複数の処理命令を含めることができます。

XML 処理命令は、ターゲットと値の 2 つの要素によって構成されます。次に例を示します。

```
<?xml-stylesheet type="text/css" href="generic.css"?>
```

XML 処理命令を追加する方法を、次のスクリプトに示します（完全なスクリプトについては、MakeProcessingInstruction を参照してください）。

```
tell application "Adobe InDesign CS6"
    set myDocument to make document
    tell myDocument
        set myRootXMLElement to XML element 1
        tell myRootXMLElement
            set myXMLInstruction to make XML instruction with properties
            {target:"xml-stylesheet type=¥"text/css¥"",
            data:"href=¥"generic.css¥""}
        end tell
    end tell
end tell
```

XML 属性の操作

XML 属性とは、XML 要素に関連付けられている「メタデータ」のことです。XML 要素に XML 属性を追加する方法を、次のスクリプトに示します（チュートリアルスクリプトの MakeXMLAttribute より）。XML 要素には任意の数の XML 属性を追加できますが、各属性の名前はその要素内で固有のものであることが必要です（例えば、「id」という名前の属性を 2 つ使用することはできません）。

```
tell application "Adobe InDesign CS6"
    set myDocument to make document
    tell myDocument
        set myXMLTag to make XML tag with properties {name:"myXMLElement"}
        set myRootXMLElement to XML element 1
        tell myRootXMLElement
            set myXMLElement to make XML element with properties
            {markup tag:myXMLTag}
            tell myXMLElement
                make XML attribute with properties
                {name:"example_attribute", value:"This is an XML attribute."}
            end tell
        end tell
    end tell
end tell
```

スクリプトを使用して属性を直接作成する以外に、XML 要素を変換して属性を作成することもできます。これを行うと、XML 要素のテキスト内容が XML 属性の値として使用され、XML 要素の親に追加されます。XML 要素の名前が属性の名前になるので、同じ名前を持つ属性が XML 要素の親に既に存在している場合は、作成に失敗します。XML 要素にページアイテムが含まれている場合、それらのページアイテムはレイアウトから削除されます。

XML 属性を XML 要素に変換する際には、新しい XML 要素を追加する場所を指定できます。新しい XML 要素は、XML 属性の親の先頭または末尾に追加できます。デフォルトでは、親要素の先頭に追加されます。

新しい XML 要素に XML マークアップタグを指定することもできます。このパラメーターを省略すると、その XML 属性が含まれている XML 要素と同じ XML タグを使用して新しい XML 要素が作成されます。

XML 要素を XML 属性に変換する方法を、次のスクリプトに示します（完全なスクリプトについては、ConvertElementToAttribute を参照してください）。

```
tell application "Adobe InDesign CS6"
    set myDocument to make document
    tell myDocument
        set myXMLTag to make XML tag with properties {name:"myXMLElement"}
        set myRootXMLElement to XML element 1
    end tell
    tell myRootXMLElement
        set myXMLElement to make XML element with properties
        {markup tag:myXMLTag, contents:"This is content in an XML element."}
    end tell
    tell myXMLElement
        convert to attribute
    end tell
end tell
```

XML 属性を XML 要素に変換することもできます。次のスクリプトにその例を示します（チュートリアルスクリプトの ConvertAttributeToElement より）。

```
tell application "Adobe InDesign CS6"
    set myDocument to make document
    tell myDocument
        set myXMLTag to make XML tag with properties {name:"myXMLElement"}
        set myRootXMLElement to XML element 1
    end tell
    tell myRootXMLElement
        set myXMLElement to make XML element with properties
        {markup tag:myXMLTag, contents:"This is content in an XML element."}
    end tell
    tell myXMLElement
        convert to attribute
    end tell
end tell
```

XML ストーリーの操作

レイアウト要素（ストーリーまたはページアイテム）に関連付けられていない XML 要素を読み込むと、その要素は XML ストーリーに格納されます。テキストフレーム内のテキストと同じように、配置されていない XML 要素内のテキストを操作することができます。次のスクリプトにその例を示します（完全なスクリプトについては、XMLStory を参照してください）。

```
tell application "Adobe InDesign CS6"
    set myDocument to make document
    tell myDocument
        set myXMLTag to make XML tag with properties {name:"myXMLElement"}
        set myRootXMLElement to XML element 1
    end tell
    tell myRootXMLElement
        set myXMLElementA to make XML element with properties
        {markup tag:myXMLTag, contents:"This is a paragraph in an XML story."}
        set myXMLElementB to make XML element with properties
        {markup tag:myXMLTag, contents:"This is another paragraph in an XML
        story."}
        set myXMLElementC to make XML element with properties
        {markup tag:myXMLTag, contents:"This is the third paragraph in an
        example XML story."}
        set myXMLElementD to make XML element with properties {markup
        tag:myXMLTag, contents:"This is the last paragraph in the XML story."}
    end tell
    set myXMLStory to xml story 1 of myDocument
    set the point size of text 1 of myXMLStory to 72
end tell
```

XML の書き出し

InDesign ドキュメントから XML を書き出す方法としては、ドキュメント内の XML 構造全体を書き出す方法と、1 つの XML 要素（とその要素に含まれているすべての子 XML 要素）を書き出す方法があります。次のスクリプトにその例を示します（完全なスクリプトについては、ExportXML を参照してください）。

```
tell myDocument
    export to "yukino:test.xml" format "XML"
end tell
```

また、xml export preferences オブジェクトの export from selected プロパティを使用して、ユーザーインターフェイスで選択されている XML 要素を書き出すこともできます。次のスクリプトにその例を示します（完全なスクリプトについては、ExportSelectedXMLElement を参照してください）。

```
tell myDocument
    set export from selected of xml export preferences to true
    select xml element 2 of xml element 1
    export to "yukino:selectedXMLElement.xml" format "XML"
    set export from selected of xml export preferences to false
end tell
```

レイアウトへの XML 要素の追加

これまでの節では、XML データを InDesign ドキュメントに読み込んで、ドキュメント内の XML 構造を操作する方法について説明しました。この節では、XML 情報をページレイアウトに読み込んでフォーマットを行う方法について説明します。

XML 要素とページアイテムやテキストとの関連付け

ページアイテムやテキストに既存の XML 要素を関連付けるには、place xml メソッドを使用します。このメソッドを使用すると、ページアイテムの内容が XML 要素の内容に置き換わります。次のスクリプトにその例を示します（チュートリアルスクリプトの PlaceXML より）。

```
tell text frame 1 of page 1 of myDocument
    place XML using xml element 1 of myDocument
end tell
```

既存のページアイテムやテキストオブジェクトに既存の XML 要素を関連付けるには、markup メソッドを使用します。このメソッドを使用すると、ページアイテムやテキストの内容と XML 要素の内容（存在する場合）が結合されます。markup メソッドを使用する方法を、次のスクリプトに示します（完全なスクリプトについては、Markup を参照してください）。

```
tell XML element 1 of XML element 1 of myDocument
    markup text frame 1 of page 1 of myDocument
end tell
```

ページアイテムへの XML の配置

XML 要素をページアイテムに関連付ける別の方法として、place into frame メソッドを使用する方法があります。このメソッドでは、XML を配置するときにフレームを作成できます。次のスクリプトにその例を示します（完全なスクリプトについては、PlaceIntoFrame を参照してください）。

```
tell application "Adobe InDesign CS6"
    set myDocument to document 1
    tell view preferences of myDocument
        set horizontal measurement units to points
        set vertical measurement units to points
    end tell
    --place into frame has two parameters:
    --on: The page, spread, or master spread on which to create the frame
    --geometric bounds: The bounds of the new frame (in page coordinates).
    tell XML element 1 of myDocument
        place into frame on page 1 geometric bounds {72, 72, 288, 288}
    end tell
end tell
```

XML 要素をインラインページアイテム（アンカーオブジェクト）に関連付けるには、`place into copy` メソッドを使用します。次のスクリプトにその例を示します（チュートリアルスクリプトの PlaceIntoCopy より）。

```
tell application "Adobe InDesign CS6"
    set myDocument to document 1
    set myPage to page 1 of myDocument
    set myTextFrame to text frame 1 of myPage
    tell XML element 1 of myDocument
        set myFrame to place into copy on myPage place point {288, 72} copy item
        myTextFrame
    end tell
end tell
```

既存のページアイテム（または既存のページアイテムのコピー）を XML 要素に関連付けて、XML 構造内のその要素がある場所にページアイテムを挿入するには、`place into inline copy` メソッドを使用します。次のスクリプトにその例を示します（チュートリアルスクリプトの PlaceIntoInlineCopy より）。

```
tell application "Adobe InDesign CS6"
    set myDocument to document 1
    set myPage to page 1 of myDocument
    tell myPage
        set myNewTextFrame to make text frame with properties
        {geometric bounds:{72, 72, 96, 144}}
    end tell
    set myXMLElement to XML element 3 of XML element 1 of myDocument
    set myFrame to place into inline copy myXMLElement copy item myNewTextFrame
    without retain existing frame
end tell
```

XML 要素を新規インラインフレームに関連付けるには、`placeIntoInlineFrame` メソッドを使用します。次のスクリプトにその例を示します（チュートリアルスクリプトの PlaceIntoInlineFrame より）。

```
set myDocument to document 1
set myXMLElement to xml element 2 of xml element 1 of myDocument
set myFrame to place into inline frame myXMLElement place point {72, 24}
```

XML テキスト要素の内部や周囲へのテキストの挿入

XML データを InDesign レイアウトに配置する際に、空白文字（改行やタブ文字など）や静的テキスト（「名前」や「住所」などのラベル）を XML 要素のテキストに追加したいことがあります。XML 要素の内部や周囲にテキストを追加する方法を、次のサンプルスクリプトに示します（完全なスクリプトについては、InsertTextAsContent を参照してください）。

```
tell application "Adobe InDesign CS6"
    set myDocument to make document
    tell myDocument
        set myXMLTag to make XML tag with properties {name:"myXMLElement"}
        set myRootXMLElement to XML element 1
        tell myRootXMLElement
            set myXMLElementA to make XML element with properties
            {markup tag:myXMLTag}
            set contents of myXMLElementA to "This is a paragraph in an XML
            story."
            set myXMLElementB to make XML element with properties
            {markup tag:myXMLTag}
            set contents of myXMLElementB to "This is a another paragraph in an
            XML story."
            set myXMLElementC to make XML element with properties {markup
            tag:myXMLTag}
            set contents of myXMLElementC to "This is the third paragraph in an
            XML story."
            set myXMLElementD to make XML element with properties {markup
            tag:myXMLTag}
            set contents of myXMLElementD to "This is the last paragraph in an XML
            story."
            tell myXMLElementA
                --By inserting the return character after the XML element, the
                --character becomes part of the content of the parent XML element,
                --and not part of the content of the XML element itself.
                insert text as content using return position after element
            end tell
            tell myXMLElementB
                insert text as content using "Static text: " position before
                element
                insert text as content using return position after element
            end tell
            tell myXMLElementC
                insert text as content using "Text at the start of an element: "
                position element start
                insert text as content using " Text at the end of an element. "
                position element end
            end tell
            tell myXMLElementD
                insert text as content using "Text before the element: " position
                before element
                insert text as content using " Text after the element. " position
                after element
            end tell
        end tell
    end tell
end tell
```

既存レイアウトのマークアップ

XML 以外のファイルに基づいて XML を作成したいことがあります（例えば、既存のページレイアウトを XML に変換したい場合）。そのような場合は、既存のページレイアウトの内容をマークアップして、XML 構造に追加します。その後、この構造を書き出して、InDesign 外部の XML ツールで処理します。

スタイルへのタグのマッピング

XML テキスト要素にフォーマットを適用する最も簡単な方法の 1 つが、`XMLImportMaps` を使用してタグをスタイルにマッピングする方法です。この方法では、特定の XML タグを段落スタイルや文字スタイルに関連付けることができます。ドキュメントの `map XML tags to styles` メソッドを使用すると、テキストにスタイルが適用されます。次のスクリプトにその例を示します（チュートリアルスクリプトの MapTagsToStyles より）。

```
tell application "Adobe InDesign CS6"
    set myDocument to document 1
    tell myDocument
        --Create a tag to style mapping.
        make XML import map with properties {markup tag:"heading_1",
        mapped style:"heading 1"}
        make XML import map with properties {markup tag:"heading_2",
        mapped style:"heading 2"}
        make XML import map with properties {markup tag:"para_1",
        mapped style:"para 1"}
        make XML import map with properties {markup tag:"body_text",
        mapped style:"body text"}
        --Map the XML tags to the defined styles.
        map XML tags to styles
        --Place the story so that you can see the result of the change.
        set myPage to page 1
        tell page 1
            set myTextFrame to make text frame with properties
            {geometric bounds:my myGetBounds(myDocument, myPage)}
            set myStory to parent story of myTextFrame
        end tell
        tell myStory
            place XML using XML element 1 of myDocument
        end tell
    end tell
end tell
```

タグへのスタイルのマッピング

XML 要素が関連付けられていないフォーマットされたテキストを XML 構造に移動したい場合は、スタイルをタグにマッピングします。これによって、段落スタイルや文字スタイルを XML タグに関連付けることができます。これを行うには、`XML export map` オブジェクトを使用して、XML タグとスタイルの間にリンクを作成し、`map styles to XML tags` メソッドを使用して該当する XML 要素を作成します。次のスクリプトにその例を示します（チュートリアルスクリプトの MapStylesToTags より）。

```
tell application "Adobe InDesign CS6"
    set myDocument to document 1
    tell myDocument
        --Create a tag to style mapping.
        make XML export map with properties {markup tag:"heading_1",
        mapped style:"heading 1"}
        make XML export map with properties {markup tag:"heading_2",
        mapped style:"heading 2"}
        make XML export map with properties {markup tag:"para_1",
        mapped style:"para 1"}
        make XML export map with properties {markup tag:"body_text",
        mapped style:"body text"}
        --Apply the style to tag mapping.
        map styles to XML tags
    end tell
end tell
```

別の方法としては、単純に、ドキュメント内のすべての段落スタイルや文字スタイルに対して新しい XML タグを作成して、スタイルをタグにマッピングする方法があります。次のスクリプトにその例を示します（チュートリアルスクリプトの MapAllStylesToTags より）。

```
tell application "Adobe InDesign CS6"
    set myDocument to document 1
    tell myDocument
        --Create a tag to style mapping.
        repeat with myParagraphStyle in paragraph styles
            set myParagraphStyleName to name of myParagraphStyle
            set myXMLTagName to my myReplace(myParagraphStyleName, " ", "_")
            set myXMLTagName to my myReplace(myXMLTagName, "[", "")
            set myXMLTagName to my myReplace(myXMLTagName, "]", "")
            set myMarkupTag to make XML tag with properties {name:myXMLTagName}
            make XML export map with properties {markup tag:myMarkupTag,
            mapped style:myParagraphStyle}
        end repeat
        map styles to XML tags
    end tell
end tell
```

グラフィックのマークアップ

グラフィックに XML 要素を関連付ける方法を、次のスクリプトに示します（完全なスクリプトについては、MarkingUpGraphics を参照してください）。

```
tell application "Adobe InDesign CS6"
    set myDocument to document 1
    tell myDocument
        set myXMLTag to make xml tag with properties{name:"graphic"}
        tell page 1
            set myGraphic to place "yukino:test.tif"
            --Associate the graphic with a new XML element as you create the
            --element
            tell XML element 1
                set myXMLElement to make XML element with properties
                {markup tag:myXMLTag, content:myGraphic}
            end tell
        end tell
end tell
```

XML 要素へのスタイルの適用

タグとスタイルをマッピングしたり、XML 要素が関連付けられているテキストやページアイテムにスタイルを適用したりする以外に、XML 要素にスタイルを直接適用することもできます。`apply paragraph style`、`apply character style`、`apply object style` の 3 つのメソッドを使用する方法を、次のスクリプトに示します（完全なスクリプトについては、ApplyStylesToXMLElements を参照してください）。

```
main()
on main()
    mySnippet()
end main
on mySnippet()
    tell application "Adobe InDesign CS6"
        set myDocument to make document
        tell myDocument
            set horizontal measurement units of view preferences to points
            set vertical measurement units of view preferences to points
            --Create a series of XML tags.
            set myHeading1XMLTag to make XML tag with properties
            {name:"heading_1"}
            set myHeading2XMLTag to make XML tag with properties
            {name:"heading_2"}
            set myPara1XMLTag to make XML tag with properties {name:"para_1"}
            set myBodyTextXMLTag to make XML tag with properties
            {name:"body_text"}
```

```
--Create a series of paragraph styles.
set myHeading1Style to make paragraph style with properties
{name:"heading 1", point size:24}
set myHeading2Style to make paragraph style with properties
{name:"heading 2", point size:14, space before:12}
set myPara1Style to make paragraph style with properties
{name:"para 1", point size:12, first line indent:0}
set myBodyTextStyle to make paragraph style with properties
{name:"body text", point size:12, first line indent:24}
--Create a character style.
set myCharacterStyle to make character style with properties
{name:"Emphasis", font style:"Italic"}
--Add XML elements.
set myRootXMLElement to XML element 1
tell myRootXMLElement
    set myXMLElementA to make XML element with properties {markup
    tag:myHeading1XMLTag, contents:"Heading 1"}
    tell myXMLElementA
        insert text as content using return position after element
        apply paragraph style using myHeading1Style clearing
        overrides yes
    end tell
    set myXMLElementB to make XML element with properties {markup
    tag:myPara1XMLTag, contents:"This is the first paragraph in the
    article."}
    tell myXMLElementB
        insert text as content using return position after element
        apply paragraph style using myPara1Style clearing overrides yes
    end tell
    set myXMLElementC to make XML element with properties {markup
    tag:myBodyTextXMLTag, contents:"This is the second paragraph in
    the article."}
    tell myXMLElementC
        insert text as content using return position after element
        apply paragraph style using myBodyTextStyle clearing
        overrides yes
    end tell
    set myXMLElementD to make XML element with properties
    {markup tag:myHeading2XMLTag, contents:"Heading 2"}
    tell myXMLElementD
        insert text as content using return position after element
        apply paragraph style using myHeading2Style clearing
        overrides yes
    end tell
    set myXMLElementE to make XML element with properties {markup
    tag:myPara1XMLTag, contents:"This is the first paragraph following
    the subhead."}
    tell myXMLElementE
        insert text as content using return position after element
        apply paragraph style using myPara1Style clearing overrides yes
    end tell
    set myXMLElementF to make XML element with properties {markup
    tag:myBodyTextXMLTag, contents:"Note:"}
    tell myXMLElementF
        insert text as content using " " position after element
        apply character style using myCharacterStyle
    end tell
    set myXMLElementG to make XML element with properties {markup
    tag:myBodyTextXMLTag, contents:"This is the second paragraph
    following the subhead."}
    tell myXMLElementG
        insert text as content using return position after element
        apply paragraph style using myBodyTextStyle clearing
```

```
                overrides no
            end tell
        end tell
    end tell
    set myPage to page 1
    tell page 1
        set myTextFrame to make text frame with properties{geometric bounds:
        my myGetBounds(myDocument, myPage)}
    end tell
    set myStory to parent story of myTextFrame
    tell myStory
        place XML using myRootXMLElement
    end tell
end tell
end mySnippet
```

XML 表の操作

HTML 標準の表タグでデータがマークアップされている場合、InDesign はその XML データを自動的に表セルに読み込みます。デフォルトの表マークアップを使用できないか、使用を避けたい場合は、`convert element to table` メソッドを使用して XML 要素を表に変換できます。

このメソッドを使用するには、表に変換する XML 要素が特定の構造になっている必要があります。表の各行が特定の XML 要素に対応しており、その要素の中に、行内の各セルに対応する一連の XML 要素が含まれている必要があります。このメソッドを使用する方法を、次のスクリプトに示します（完全なスクリプトについては、ConvertXMLElementToTable を参照してください）。テーブル行を表すために使用されている XML 要素は、このプロセスによって変換されます。

```
tell application "Adobe InDesign CS6"
    set myDocument to make document
    tell myDocument
        --Create a series of XML tags.
        set myRowTag to make XML tag with properties {name:"Row"}
        set myCellTag to make XML tag with properties {name:"Cell"}
        set myTableTag to make XML tag with properties {name:"Table"}
        --Add XML elements.
        set myRootXMLElement to XML element 1
        tell myRootXMLElement
            set myTableXMLElement to make XML element with properties {markup
            tag:myTableTag}
            tell myTableXMLElement
                repeat with myRowCounter from 1 to 6
                    set myXMLRow to make XML element with properties {markup
                    tag:myRowTag}
                    tell myXMLRow
                        set myString to "Row " & myRowCounter
                        repeat with myCellCounter from 1 to 4
                            make XML element with properties {markup tag:myCellTag,
                            contents:myString & ":Cell " & myCellCounter}
                        end repeat
                    end tell
```

```
                end repeat
                convert element to table row tag myRowTag cell tag myCellTag
            end tell
        end tell
        set myPage to page 1
        tell page 1
            set myTextFrame to make text frame with properties{geometric bounds:
            my myGetBounds(myDocument, myPage)}
        end tell
        set myStory to parent story of myTextFrame
        tell myStory
            place XML using XML element 1 of myDocument
        end tell
    end tell
end tell
```

XML 要素で構成されている表では、XML 要素に関連付けられている表やセルにスタイルを適用するのではなく、XML 要素に表スタイルやセルスタイルを直接適用することができます。これを行うには、`apply table style` メソッドや `apply cell style` メソッドを使用します。次のスクリプトにその例を示します（チュートリアルスクリプトの ApplyTableStyles より）。

```
tell application "Adobe InDesign CS6"
    set myDocument to make document
    tell myDocument
        --Create a series of XML tags.
        set myRowTag to make XML tag with properties {name:"Row"}
        set myCellTag to make XML tag with properties {name:"Cell"}
        set myTableTag to make XML tag with properties {name:"Table"}
        --Add XML elements.
        set myRootXMLElement to XML element 1
        tell myRootXMLElement
            set myTableXMLElement to make XML element with properties {markup
            tag:myTableTag}
            tell myTableXMLElement
                repeat with myRowCounter from 1 to 6
                    set myXMLRow to make XML element with properties {markup
                    tag:myRowTag}
                    tell myXMLRow
                        set myString to "Row " & myRowCounter
                        repeat with myCellCounter from 1 to 4
                            make XML element with properties {markup tag:myCellTag,
                            contents:myString & ":Cell " & myCellCounter}
                        end repeat
                    end tell
                end repeat
                convert element to table row tag myRowTag cell tag myCellTag
            end tell
        end tell
        set myPage to page 1
        tell page 1
            set myTextFrame to make text frame with properties{geometric bounds:
            my myGetBounds(myDocument, myPage)}
        end tell
        set myStory to parent story of myTextFrame
        tell myStory
            place XML using XML element 1 of myDocument
        end tell
    end tell
end tell
```

13 XML ルール

章の更新ステータス	
CS6	変更なし

InDesign の XML ルール機能は、ドキュメントに含まれている XML コンテンツをスクリプトで処理するための強力なツールです。XML ルールを使用すれば、XML 要素を処理するスクリプトを簡単に記述できるようになるとともに、XML 要素の検索、変更、フォーマットを効率よく行えるようになります。

XML ルールは、オープン、配置、クローズなどのアプリケーションイベントによって実行することができます。一般的には、ドキュメントに XML を読み込んだ後に実行します（イベントへのスクリプトの割り当てについて詳しくは、第 8 章「イベント」を参照してください）。

この章では、XML ルールの構造と処理について簡単に説明した後で、次の方法について説明します。

- XML ルールの定義
- XML ルールの適用
- XML ルールを使用した XML 要素の検索
- XML ルールを使用した XML データのフォーマット
- XML ルールに基づいたページアイテムの作成
- XML ルールを使用したデータの再構築
- XML ルールプロセッサーの使用

ここでは、読者が『Adobe InDesign スクリプティングチュートリアル』を既に読んでおり、スクリプトを作成して実行する方法を理解しているものとして説明を行います。また、XML に関する基礎知識を持っており、第 12 章「XML」を読み終えていることを前提としています。

概要

InDesign の XML ルール機能は、次の 3 つの部分から構成されています。

- **XML ルールプロセッサー（単一のスクリプトオブジェクト）** ― XPath を使用して、XML 構造の中から XML 要素を検索し、適切な XML ルールを適用します。次のことは重要です。1 つのスクリプトには、複数の XML ルールプロセッサーオブジェクトを含めることができます。それぞれの XML ルールプロセッサーオブジェクトには、それぞれ 1 つの XML ルールセットが関連付けられます。
- **グルーコード** ― XML ルールを記述するときや、XML ルールプロセッサーとやり取りを行うときに便利な一連の関数（Adobe 提供）。
- **XML ルール** ― スクリプトに追加する XML アクション。XML ルールはスクリプトコードで記述します。1 つのルールは、XPath ベースの 1 つの条件と、その条件が満たされたときに適用する 1 つの関数に結び付いています。「apply」関数では、InDesign スクリプティングで定義可能な処理であれば、どのような処理でも実行できます。例えば、XML 構造の変更、フォーマットの適用、ページ、ページアイテム、ドキュメントの新規作成などが行えます。

1 つのスクリプトでは、いくつでもルールを定義できます。ルールは、InDesign ドキュメントの XML 構造全体に適用することもできますし、XML 構造の中の一部の要素に適用することもできます。XML ルールプロ

セッサーによって XML ルールが実行されると、ドキュメント内の一致する XML 要素やその他のオブジェクトに変更が加えられます。

XML ルールは、XSLT に似た機能と考えることができます。XSLT では、XPath を使用して XML 構造内の XML 要素を検索し、何らかの変換を行いますが、XML ルールでは、XPath を使用して InDesign の内部で XML 要素を検索し、何らかのアクションを行います。XSLT のテンプレートでは、InDesign の外部にある XML パーサーを使用して XML データに変換を適用しますが、InDesign のルールプロセッサーでは、XML ルールを使用して InDesign の内部で XML データに変換を適用します。

XML ルールを使用する利点

以前のリリースの InDesign では、XPath を使用して InDesign ファイルの XML 構造をナビゲートすることはできませんでした。XML 構造を処理するには、再帰スクリプト関数を作成して、各要素を繰り返し調べていく必要がありました。この処理は難しく、時間がかかりました。

XML ルールを使用すれば、構造内の XML 要素を簡単に検索できます。XML 要素の検索には、XPath と InDesign の XML ルールプロセッサーが使用されます。XML ルールプロセッサーは、ドキュメント内の XML 要素の繰り返し作業をスクリプトよりも高速に処理できます。

XML ルールのプログラミングモデル

XML ルールは、次の 3 つの要素で構成されます。

1. 名前（文字列）
2. `XPath` 文（文字列）
3. `apply` 関数

XPath 文では、XML 構造内の場所を定義します。XML ルールプロセッサーによって、これに一致する要素が検索され、ルールで定義されている apply 関数が実行されます。

XML ルールのサンプルを次に示します。

```
to RuleName()
    script RuleName
        property name:"RuleNameAsString"
        property xpath: "ValidXPathSpecifier"
        on apply(element, ruleSet, ruleProcessor)
            --Do something here.
            --Return true to stop further processing of the XML element.
            return false
        end apply
    end script
end RuleName
```

この例では、ルールの名前は `RuleNameAsString` であり、`RuleName` に関連付けられます。XPath 式は `ValidXPathSpecifier` です。実際に使用できる様々な XML ルールの例については、この章の後の節を参照してください。

注意 : XML ルールでは、XPath 1.0 の制限されたサブセットがサポートされています。200 ページの「XPath の制限事項」を参照してください。

XML ルールセット

XML ルールセットは、XML ルールプロセッサーで適用する 1 つ以上の XML ルールの配列です。それぞれのルールは、配列に格納されている順番に適用されます。XML ルールセットのサンプルを次に示します。

```
set myRuleSet to {my SortByName(), my AddStaticText(), my LayoutElements(), my FormatElements()}
```

この例では、myRuleSet という配列に列挙されているルールが、スクリプトの別の場所に定義されています。XML ルールセットを利用した、実際に使用できる様々なスクリプトについては、この章の後述を参照してください。

「グルー」コード

InDesign のスクリプトモデルに組み込まれている XML ルールプロセッサーオブジェクトに加えて、XML ルールの作成に役立つ一連の関数が用意されています。glue code.as というファイルに、次の関数が定義されています。

- **__processRuleSet(root, ruleSet)** — 一連の XML ルールを実行するには、__processRuleSet 関数を呼び出して、XML 要素や XML ルールセットを指定する必要があります。この XML 要素では、ルールの処理を開始する XML 構造内のポイントを定義します。

- **__processChildren(ruleProcessor)** — 一致する XML 要素の子要素に対して一致する XML ルールを適用するように、XML ルールプロセッサーに指示します。これを使用すれば、子 XML 要素が処理された後に、親 XML 要素に適用したルールでコードを実行することができます。デフォルトでは、XML ルールプロセッサーによって XML 要素の子にルールが適用されると、制御はそのルールに戻されません。__processChildren 関数を使用すれば、子 XML 要素が処理された後に、ルールの apply 関数に制御を戻すことができます。

- **__skipChildren(ruleProcessor)** — 現在の XML 要素の下位要素を XML ルールで処理しないようにプロセッサーに指示します。これは、現在の XML 要素を移動または削除する場合や、XML 構造の不要な部分をスキップしてパフォーマンスを向上させたい場合などに役立ちます。

XML 構造内の反復処理

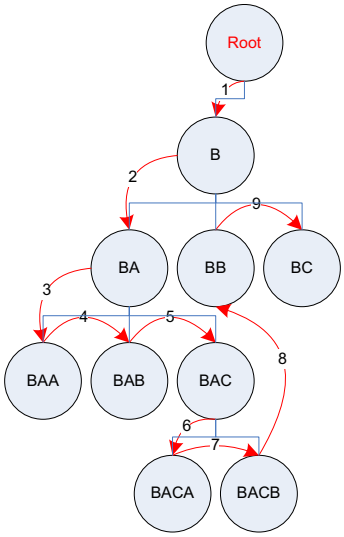
XML ルールプロセッサーは、ドキュメントの XML 階層に現れる順番で XML 要素を反復します。XML 構造を順方向に通過していき、構造内のそれぞれの XML 要素を 2 回訪れます（XML ファイル内でネストしているタグの通常の順番と同じ、親、子、親の順番）。XML ルールプロセッサーは、すべての XML 要素に対して、一致するすべての XML ルールを、現在の XML ルールセットに列挙されている順番で適用しようとします。

__processRuleSet 関数は、「深さ優先」の順番で XML 要素にルールを適用します。つまり、XML 要素とその子要素は、XML 構造に現れる順番で処理されます。XML ルールプロセッサーは、XML 構造の 1 つの「枝」に含まれているすべての XML 要素を訪れてから、次の「枝」に移動します。

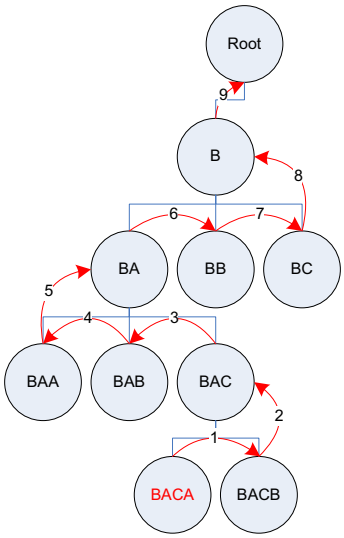
XML ルールプロセッサーは、ある XML 要素に XML ルールを適用すると、その XML 要素の下位要素に適用できるルールがあるかどうかを調べます。__skipChildren または __processChildren 関数を XML ルールで使用したり、他のルールの動作を変更したりすれば、この動作を変えることができます。

これらの関数の動作を確認するには、DepthFirstProcessingOrder.xml ファイルを新規ドキュメントに読み込んで、DepthFirstProcessingOrder.jsx スクリプトを実行します。InDesign でテキストフレームが作成され、サンプル XML ファイルに含まれている各要素の属性名が、各ルールの訪問順にリストされます。このスクリプトと AddAttribute チュートリアルスクリプトを組み合わせれば、作成した XML ドキュメントでの XML 検索の問題点を見つけることができます（XML 構造に合わせて AddAttribute スクリプトを編集する必要があります）。

次の図に、通常の反復処理を示します（構造内のすべての XML 要素に一致するルールが 1 つある場合）。



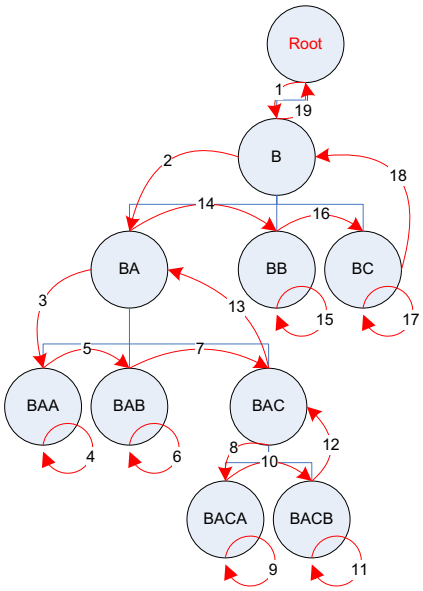
次の図に、__processChildren を使用した場合の反復処理を示します（構造内のすべての XML 要素に一致するルールが 1 つある場合）。



次のルールセットを使用した場合の反復処理を、スクリプトの後の図に示します。ルールセットには、すべての要素に一致する 2 つのルールが含まれており、1 つのルールで __processChildren が使用されています。すべての要素は 2 回処理されます （完全なスクリプトについては、ProcessChildren を参照してください）。

```
to NormalRule()
    script NormalRule
        property name : "NormalRule"
        property xpath : "//XMLElement"
        on apply(myXMLElement, myRuleProcessor)
            global myReturnString
            set myReturnString to "OK"
            tell application "Adobe InDesign CS6"
                try
                    tell insertion point -1 of story 1 of document 1
                        set contents to value of XML attribute 1 of myXMLElement &
                        return
                    end tell
                on error myError
                    set myReturnString to myError
```

```
                end try
            end tell
            return false
        end apply
    end script
end NormalRule
to ProcessChildrenRule()
    script ProcessChildrenRule
        property name : "ProcessChildrenRule"
        property xpath : "//XMLElement"
        on apply(myXMLElement, myRuleProcessor)
            tell myGlueCode
                __processChildren(myRuleProcessor)
            end tell
            global myReturnString
            set myReturnString to "OK"
            tell application "Adobe InDesign CS6"
                try
                    tell insertion point -1 of story 1 of document 1
                        set contents to value of XML attribute 1 of myXMLElement &
                        return
                    end tell
                on error myError
                    set myReturnString to myError
                end try
            end tell
            return false
        end apply
    end script
end ProcessChildrenRule
```



反復処理時の構造の変更

XML ルールプロセッサーが一致する XML 要素に XML ルールを適用したときに、このルールによってドキュメントの XML 構造が変更されることがあります。XML ルールプロセッサーの現在のパスに含まれている XML 要素が変更された場合は、他のルールの適用処理との間で競合が発生する可能性があります。XML ルールで XML 構造を変更する場合は、XML ルールプロセッサーでエラーが発生しないように、次の制限を守ってください。

- **上位にある XML 要素の削除** — 一致した XML 要素の上位にある XML 要素を削除したい場合は、その上位にある XML 要素に一致および処理を行うルールを別途作成します。
- **親 XML 要素の挿入** — 一致した XML 要素に上位の XML 要素を追加したい場合は、現在の XML 要素を処理した後に追加します。上位の XML 要素を追加しても、XML ルールプロセッサーで現在実行されているルールの反復処理では処理されません（追加した要素は現在の要素の「上」にあるからです）。
- **現在の XML 要素の削除** — 一致した XML 要素に含まれている子 XML 要素がすべて処理されるまで、その要素の削除や移動は行えません。このような変更を行いたい場合は、`__skipChildren` 関数を使用した上で変更を行います。
- **処理は再実行されない** — 処理済みのノードを変更しても、XML ルールが再実行されることはありません。

複数のルールが一致する場合の処理

複数のルールが 1 つの XML 要素に一致する場合は、XML ルールプロセッサーによって、一致するルールの一部またはすべてが適用されます。いずれかのルールの `apply` 関数が `true` を返すまで、ルールセットに列挙されている順に XML ルールが適用されます。つまり、戻り値の `true` は、要素の処理が完了したことを表します。あるルールで `true` が返されると、その XML 要素に一致する他の XML ルールがあっても無視されます。XML ルールの `apply` 関数で `false` を返せば、次に一致するルールが適用されるようになります。

`apply` 関数で `false` を返すときは、XML ルールで定義された `XPath` プロパティ以外の条件（スクリプトの他の変数の状態など）に基づいて、XML ルールの一致動作を制御することができます。

XPath の制限事項

InDesign の XML ルールでは、XPath 1.0 仕様の制限されたサブセットがサポートされています。その代表的なものを次に示します。

- 名前（ルートからのパス）による要素の検索（`/doc/title` など）。
- ワイルドカードやノード一致を使用したパスの検索（`/doc/*/subtree/node()` など）。
- 指定の属性が指定の値である要素の検索（`/doc/para[@font='Courier']` など）。
- 指定の属性が指定の値でない要素の検索（`/doc/para[@font !='Courier']` など）。
- 数字のインデックス（ただし `last()` は除く）を使用した子要素の検索（`/doc/para[3]` など）。
- 現在の要素（self）または子孫要素（descendant）の検索（`//para` など）。
- 終端としてのコメント（comment()）の検索（`/doc/comment()` など）。
- ターゲットなどによる PI の検索（`/doc/processing-instruction('foo')` など）。
- 複数の述語（predicate）の検索（`/doc/para[@font='Courier'][@size=5][2]` など）。
- following-sibling 軸に沿った検索（`/doc/note/following-sibling::*` など）。

この実装では 1 パスの性質があるので、特に次の XPath 式は使用できません。

- `..`、`ancestor::`、`preceding-sibling::` などの、上位の要素や preceding-sibling 軸は使用できません。
- 述語でのパス指定は行えません（`foo[bar/c]` など）
- `last()` 関数は使用できません
- `text()` 関数の使用やテキストの比較は行えません。ただし、InDesign スクリプティングを使用して、目的の XML ルールに一致する XML 要素のテキストコンテンツを調べることは可能です。

- 述語でのブール演算の組み合わせは行えません（`foo[@bar=font or @c=size]` など）。
- 述語での比較は行えません（`foo[@bar < font or @c > 3]` など）。
- 相対パスは使用できません（`doc/chapter` など）。

エラー処理

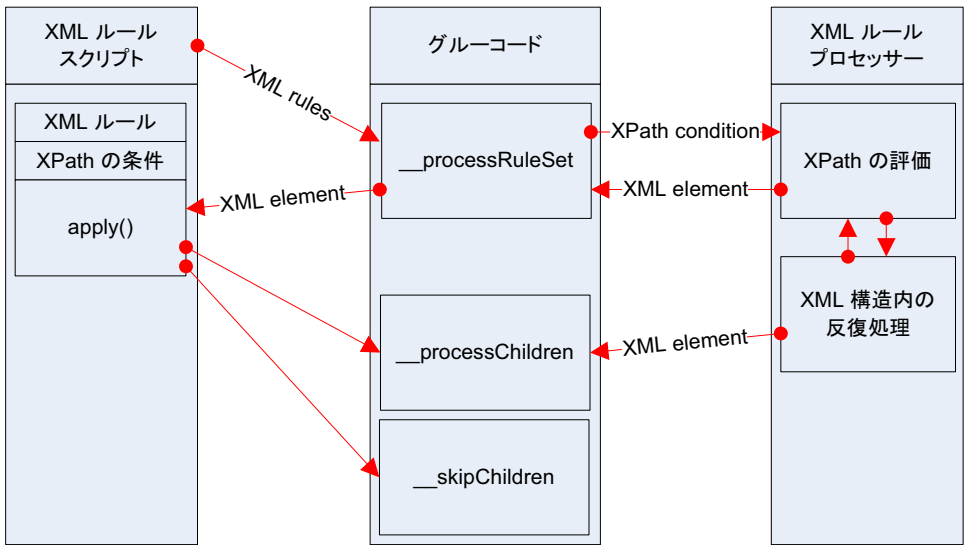
XML ルールは InDesign スクリプティングモデルの一部分なので、ルールを使用したスクリプトの基本的な性質は、通常のスクリプトと同じです。したがって、エラー処理メカニズムも同じものを使用できます。InDesign にエラーが発生した場合の動作は、XML ルールスクリプトも通常のスクリプトも同じです。InDesign のエラーは、スクリプト言語に用意されている任意のエラー捕捉メカニズム（`try...catch` ブロックなど）を使用して捕捉できます。

InDesign には、XML ルールの処理に固有のエラーがいくつか用意されています。サポートされていない XPath 指定がルールに使用されていると、XML ルールプロセッサーの初期化時に InDesign エラーが発生することがあります（200 ページの「XPath の制限事項」を参照）。また、モデルの変更によって XML ルールプロセッサーが無効な状態になった場合にも InDesign エラーが発生することがあります。XML ルールの処理で XML 構造を変更すると、XML ルールプロセッサーが無効になる場合があります。XML 構造に対するこのような変更は、XML ルールプロセッサーを使用したスクリプト、同時に実行されている別のスクリプト、ユーザーインターフェイスで実行されたユーザーアクションなどによって行われる可能性があります。

XML ルールプロセッサーを無効にするような変更が XML 構造に加えられた場合は、XML ルールプロセッサーの反復処理が再開されるとエラーが発生します。エラーメッセージには、エラーの原因となった XML 構造の変更が表示されます。

XML ルールの制御フロー

XML ルールを含むスクリプトを実行すると、XML ルールを含むスクリプト関数から各 XML ルールに制御が渡され、さらに各ルールからグルーコードで定義されている関数に制御が渡されます。これらの関数から XML ルールプロセッサーに制御が渡され、プロセッサーが構造内の XML 要素を反復処理します。その結果やエラーは、関数によって処理されるか、スクリプトエラーが発生するまで、このチェーンを逆方向に渡されていきます。次の図に、XML ルールスクリプトの制御フローの概要を示します。



XML ルールの例

XML ルールでは、XPath 文を使用して目的の XML 要素を指定します。したがって、XML ルールとドキュメントの XML 構造の間には緊密な関係があります。この節では、実際に使用できる XML ルールスクリプトをいくつか示しますが、そのためには、スクリプトの実行対象となる XML 構造も同時に示す必要があります。この章の以降の各節では、サンプル XML データファイルに基づいた XML ルールの例を示します。この例では、架空の IC メーカーの製品リストを使用します。XML データファイルの各レコードの構造は、次のとおりです。

```
<device>
    <name></name>
    <type></type>
    <part_number></part_number>
    <supply_voltage>
        <minimum></minimum>
        <maximum></maximum>
    </supply_voltage>
    <package>
        <type></type>
        <pins></pins>
    </package>
    <price></price>
    <description></description>
</device>
```

最初は簡単なスクリプトから始めて、徐々に複雑なスクリプトへと進んでいきます。

サンプルドキュメントのセットアップ

この章の各スクリプトを実行する前に、ドキュメントに `XMLRulesExampleData.xml` データファイルを読み込みます。XML を読み込む場合は、XML 読み込みオプションダイアログボックスの「空白のみの要素を読み込まない」オプションをオンにします。ファイルを保存し、ファイル／復帰を選択してから、この節の各サンプルスクリプトを実行します。または、次のスクリプトを実行してから、各サンプル XML ルールスクリプトを実行します（`XMLRulesExampleSetup.jsx` というスクリプトファイルを参照してください）。

```
//XMLRuleExampleSetup.jsx
//
main();
function main(){
    var myDocument = app.documents.add();
    myDocument.xmlImportPreferences.allowTransform = false;
    myDocument.xmlImportPreferences.ignoreWhitespace = true;
    var myScriptPath = myGetScriptPath();
    var myFilePath = myScriptPath.path + "/XMLRulesExampleData.xml"
    myDocument.importXML(File(myFilePath));
    var myBounds = myGetBounds(myDocument, myDocument.pages.item(0));
    myDocument.xmlElements.item(0).placeIntoFrame(myDocument.pages.item(0), myBounds);
    function myGetBounds(myDocument, myPage){
        var myWidth = myDocument.documentPreferences.pageWidth;
        var myHeight = myDocument.documentPreferences.pageHeight;
```

```
        var myX1 = myPage.marginPreferences.left;
        var myY1 = myPage.marginPreferences.top;
        var myX2 = myWidth - myPage.marginPreferences.right;
        var myY2 = myHeight - myPage.marginPreferences.bottom;
        return [myY1, myX1, myY2, myX2];
    }
    function myGetScriptPath() {
        try {
            return app.activeScript;
        }
        catch(myError){
            return File(myError.fileName);
        }
    }
}
```

最初の XML ルール

まず、非常に単純な XML ルールを作成します。このルールでは、ドキュメントの各 XML 要素の後に改行文字を追加します。XML ルールセットには 1 つのルールが含まれています。完全なスクリプトについては、AddReturns を参照してください。

```
global myGlueCode
on run
    tell application "Adobe InDesign CS6"
        set myRootXML to first XML element of document 1
        set myApplicationPath to file path
        set myFilePath to file path as string
        set myFilePath to myFilePath & "Scripts:Xml rules:glue code.scpt"
        set myGlueCode to load script file myFilePath
        set myRuleSet to {my AddReturns()}
    end tell
    tell myGlueCode
        --The third parameter of __processRuleSet is a
        --prefix mapping table; we'll leave it empty.
        __processRuleSet(myRootXML, myRuleSet, {})
    end tell
end run
to AddReturns()
    script AddReturns
        property name : "AddReturns"
        property xpath : "//*"
        on apply(myXMLElement, myRuleProcessor)
            global myReturnString
            set myReturnString to "OK"
            tell application "Adobe InDesign CS6"
                try
                    tell myXMLElement
                        insert text as content using return position element end
                    end tell
                on error myError
                    set myReturnString to myError
                end try
            end tell
            return false
        end apply
    end script
end AddReturns
```

空白文字と静的テキストの追加

次の XML ルールスクリプトは、空白文字と静的テキストを追加するという点では前のスクリプトと似ていますが、要素名に基づいて XML 要素の扱いを変えている点で少し複雑なものになっています。完全なスクリプトについては、AddReturnsAndStaticText を参照してください。

```
global myGlueCode
on run
    tell application "Adobe InDesign CS6"
        set myRootXML to first XML element of document 1
        set myApplicationPath to file path
        set myFilePath to file path as string
        set myFilePath to myFilePath & "Scripts:Xml rules:glue code.scpt"
        set myGlueCode to load script file myFilePath
        set myRuleSet to {my ProcessDevice(), my ProcessName(), my ProcessType(), my
ProcessPartNumber(), my ProcessSupplyVoltage(), my ProcessPackageType(), my ProcessPackageOne(),
my ProcessPackages(), my ProcessPrice()}
    end tell
    tell myGlueCode
        --The third parameter of __processRuleSet is a
        --prefix mapping table; we'll leave it empty.
        __processRuleSet(myRootXML, myRuleSet, {})
    end tell
end run
to ProcessDevice()
    script ProcessDevice
        property name : "ProcessDevice"
        property xpath : "/devices/device"
        on apply(myXMLElement, myRuleProcessor)
            global myReturnString
            set myReturnString to "OK"
            tell application "Adobe InDesign CS6"
                try
                    tell myXMLElement
                        insert text as content using return position after element
                    end tell
                on error myError
                    set myReturnString to myError
                end try
            end tell
            return true
        end apply
    end script
end ProcessDevice
to ProcessName()
    script ProcessName
        property name : "ProcessName"
        property xpath : "/devices/device/name"
        on apply(myXMLElement, myRuleProcessor)
            global myReturnString
            set myReturnString to "OK"
            tell application "Adobe InDesign CS6"
                try
                    tell myXMLElement
                        insert text as content using "Device Name: " position
                        before element
                        insert text as content using return position after element
                    end tell
                on error myError
                    set myReturnString to myError
                end try
            end tell
            return true
        end apply
    end script
```

```
end ProcessName
to ProcessType()
    script ProcessType
        property name : "ProcessType"
        property xpath : "/devices/device/type"
        on apply(myXMLElement, myRuleProcessor)
            global myReturnString
            set myReturnString to "OK"
            tell application "Adobe InDesign CS6"
                try
                    tell myXMLElement
                        insert text as content using "Circuit Type: " position
                        before element
                        insert text as content using return position after element
                    end tell
                on error myError
                    set myReturnString to myError
                end try
            end tell
            return true
        end apply
    end script
end ProcessType
to ProcessPartNumber()
    script ProcessPartNumber
        property name : "ProcessPartNumber"
        property xpath : "/devices/device/part_number"
        on apply(myXMLElement, myRuleProcessor)
            global myReturnString
            set myReturnString to "OK"
            tell application "Adobe InDesign CS6"
                try
                    tell myXMLElement
                        insert text as content using "Part Number: " position before
                        element
                        insert text as content using return position after element
                    end tell
                on error myError


                    set myReturnString to myError
                end try
            end tell
            return true
        end apply
    end script
end ProcessPartNumber
to ProcessSupplyVoltage()
    script ProcessSupplyVoltage
        property name : "ProcessSupplyVoltage"
        property xpath : "/devices/device/supply_voltage"
        on apply(myXMLElement, myRuleProcessor)
            global myReturnString
            set myReturnString to "OK"
            tell application "Adobe InDesign CS6"
                try
                    tell myXMLElement
                        insert text as content using "Supply Voltage: From: "
                        position before element
                        insert text as content using return position after element
                        tell XML element 1
                            insert text as content using " to " position after
                            element
                        end tell
                        tell XML element -1
                            insert text as content using " volts" position after
```

```
                        element
                    end tell
                end tell
            on error myError
                set myReturnString to myError
            end try
        end tell
        return true
    end apply
end script
end ProcessSupplyVoltage
to ProcessPackageType()
    script ProcessPackageType
        property name : "ProcessPackageType"
        property xpath : "/devices/device/package/type"
        on apply(myXMLElement, myRuleProcessor)
            global myReturnString
            set myReturnString to "OK"
            tell application "Adobe InDesign CS6"
                try
                    tell myXMLElement
                        insert text as content using "-" position after element
                    end tell
                on error myError
                    set myReturnString to myError
                end try
            end tell
            return true
        end apply
    end script
end ProcessPackageType
--Add the text "Package:" before the list of packages.
to ProcessPackageOne()
    script ProcessPackageOne


        property name : "ProcessPackageOne"
        property xpath : "/devices/device/package[1]"
        on apply(myXMLElement, myRuleProcessor)
            global myReturnString
            set myReturnString to "OK"
            tell application "Adobe InDesign CS6"
                try
                    tell myXMLElement
                        insert text as content using "Package: " position before
                        element
                    end tell
                on error myError
                    set myReturnString to myError
                end try
            end tell
            return true
        end apply
    end script
end ProcessPackageOne
--Add commas between the package types and a return at the end of the packages.
to ProcessPackages()
    script ProcessPackages
        property name : "ProcessPackages"
        property xpath : "/devices/device/package"
        on apply(myXMLElement, myRuleProcessor)
            global myReturnString
            set myReturnString to "OK"
            tell application "Adobe InDesign CS6"
                try
                    tell myXMLElement
```

```
                    set myIndex to index of myXMLElement
                    if myIndex is not 1 then
                        insert text as content using ", " position before element
                    end if
                end tell
            on error myError
                set myReturnString to myError
            end try
        end tell
        return true
    end apply
end script
end ProcessPackages
to ProcessPrice()
    script ProcessPrice
        property name : "ProcessPrice"
        property xpath : "/devices/device/price"
        on apply(myXMLElement, myRuleProcessor)
            global myReturnString
            set myReturnString to "OK"
            tell application "Adobe InDesign CS6"


                try
                    tell myXMLElement
                        insert text as content using return position before element
                        insert text as content using "Price: $" position
                        before element
                        insert text as content using return position after element
                    end tell
                on error myError
                    set myReturnString to myError
                end try
            end tell
            return true
        end apply
    end script
end ProcessPrice
```

注意 : このスクリプト（の `ProcessPackages` という XML ルール）では、繰り返される要素の間にカンマを追加しています。同じレベルに類似の要素が並んでいる場合は、forward-axis 一致を使用して同じことを実行できます。次のような XML 構造があるとします。

```
<xmlElement><item>1</item><item>2</item><item>3</item><item>4</item>
</xmlElement>
```

それぞれの `item` という XML 要素の間にカンマを追加してレイアウトするには、次のような XML ルールを使用します（チュートリアルスクリプトの ListProcessing より）。

```
    global myGlueCode
    on run
        tell application "Adobe InDesign CS6"
            set myRootXML to first XML element of document 1
            set myApplicationPath to file path
            set myFilePath to file path as string
            set myFilePath to myFilePath & "Scripts:Xml rules:glue code.scpt"
            set myGlueCode to load script file myFilePath
            set myRuleSet to {my ListItems()}
        end tell
        tell myGlueCode
            __processRuleSet(myRootXML, myRuleSet, {})
        end tell
    end run
    to ListItems()
        script ListItems
            property name : "ListItems"
            property xpath : "/xmlElement/item[1]/following-sibling::*"
            on apply(myXMLElement, myRuleProcessor)
                tell application "Adobe InDesign CS6"
                    tell myXMLElement
                        insert text as content using ", " position before element
                    end tell
                end tell
                return false
            end apply
        end script
    end ListItems
```

XML ルールを使用した XML 構造の変更

InDesign の XML の実装においては、XML 要素の順番が重要になるので、XML ルールを使用して構造内の要素の順番を変更したい場合が考えられます。一般に、XML ドキュメントの構造を大幅に変更する場合は、InDesign に XML を読み込む前か読み込むときに、XSLT ファイルを使用してドキュメントを変換するのが最もよい方法です。

`move` メソッドを使用してこれを行う方法を、次の XML ルールスクリプトに示します。XML プロセッサーが無効にならないように、グルーコードで `__skipChildren` 関数を使用している点に注意してください。完全なスクリプトについては、MoveXMLElement を参照してください。

```
global myGlueCode
on run
    tell application "Adobe InDesign CS6"
        set myRootXML to first XML element of document 1
        set myApplicationPath to file path
        set myFilePath to file path as string
        set myFilePath to myFilePath & "Scripts:Xml rules:glue code.scpt"
        set myGlueCode to load script file myFilePath
        set myRuleSet to {my MoveElement()}
    end tell
    tell myGlueCode
        --The third parameter of __processRuleSet is a
        --prefix mapping table; we'll leave it empty.
        __processRuleSet(myRootXML, myRuleSet, {})
    end tell
end run
to MoveElement()
    script MoveElement
        property name : "MoveElement"
        property xpath : "/devices/device/part_number"
        on apply(myXMLElement, myRuleProcessor)
            tell myGlueCode
                __skipChildren(myRuleProcessor)
            end tell
```

```
            global myReturnString
            set myReturnString to "OK"
            tell application "Adobe InDesign CS6"
                try
                    tell myXMLElement
                        set myParent to parent
                        set myTargetXMLElement to XML element 1 of myParent
                        move to before myTargetXMLElement
                    end tell
                on error myError
                    set myReturnString to myError
                end try
            end tell
            return true
        end apply
    end script
end MoveElement
```

XML ルールを使用した XML 要素の複製

第 12 章「XML」で説明したように、XML 要素とレイアウト内の対応する表現との間には、1 対 1 の関係があります。ある XML 要素の内容をレイアウト内で複数表示したい場合は、その要素を複製する必要があります。XML ルールを使用して要素を複製する方法を、次のスクリプトに示します。完全なスクリプトについては、DuplicateXMLElement を参照してください。このルールでも、`__skipChildren` を使用して無効な XML オブジェクト参照を回避しています。

```
global myGlueCode
on run
    tell application "Adobe InDesign CS6"
        set myRootXML to first XML element of document 1
        set myApplicationPath to file path
        set myFilePath to file path as string
        set myFilePath to myFilePath & "Scripts:Xml rules:glue code.scpt"
        set myGlueCode to load script file myFilePath
        set myRuleSet to {my DuplicateElement()}
    end tell
    tell myGlueCode
        --The third parameter of __processRuleSet is a
        --prefix mapping table; we'll leave it empty.
        __processRuleSet(myRootXML, myRuleSet, {})
    end tell
end run
to DuplicateElement()
    script DuplicateElement
        property name : "DuplicateElement"
        property xpath : "/devices/device/part_number"
        on apply(myXMLElement, myRuleProcessor)
            --Because this rule makes changes to the XML structure,
            --you must use __skipChildren to avoid invalidating
            --the XML object references.
            tell myGlueCode
                __skipChildren(myRuleProcessor)
            end tell
```

```
            global myReturnString
            set myReturnString to "OK"
            tell application "Adobe InDesign CS6"
                try
                    tell myXMLElement
                        duplicate
                    end tell
                on error myError
                    set myReturnString to myError
                end try
            end tell
            return true
        end apply
    end script
end DuplicateElement
```

XML ルールと XML 属性

次の XML ルールでは、「name」要素の内容に基づいて XML 要素に属性を追加しています。XML ルールで、テキストコンテンツに基づいて要素を検索する必要がある場合は、XML 要素の内容を、親の XML 要素の XML 属性にコピーまたは移動すると便利です。XML ルールでサポートされている XPath のサブセットでは、要素のテキストを検索することはできませんが、属性値に基づいて要素を検索することは可能です。完全なスクリプトについては、AddAttribute を参照してください。

```
global myGlueCode
on run
    tell application "Adobe InDesign CS6"
        set myRootXML to first XML element of document 1
        set myApplicationPath to file path
        set myFilePath to file path as string
        set myFilePath to myFilePath & "Scripts:Xml rules:glue code.scpt"
        set myGlueCode to load script file myFilePath
        set myRuleSet to {my AddAttribute()}
    end tell
    tell myGlueCode
        __processRuleSet(myRootXML, myRuleSet, {})
    end tell
end run
to AddAttribute()
    script AddAttribute
        property name : "AddAttribute"
        property xpath : "/devices/device/part_number"
        --Adds the content of an XML element to an attribute of
        --the parent of the XML element. This can make finding the
        --element by its contnet much easier and faster.
        on apply(myXMLElement, myRuleProcessor)
            global myReturnString
            set myReturnString to "OK"
            tell application "Adobe InDesign CS6"
                try
```

```
                    tell myXMLElement
                        set myString to contents of text 1
                        tell parent
                            make XML attribute with properties {name:"part_number",
                            value:myString}
                        end tell
                    end tell
                on error myError
                    set myReturnString to myError
                end try
            end tell
            return true
        end apply
    end script
end AddAttribute
```

前述の XML ルールでは、ある XML 要素のデータを、親の XML 要素の XML 属性にコピーしました。では、XML 要素のデータを属性に移動して、その XML 要素を削除したい場合はどうすればよいでしょうか。その場合は、`convertToAttribute` メソッドを使用します。次のスクリプトにその例を示します（チュートリアルスクリプトの ConvertToAttribute より）。

```
global myGlueCode
on run
    tell application "Adobe InDesign CS6"
        set myRootXML to first XML element of document 1
        set myApplicationPath to file path
        set myFilePath to file path as string
        set myFilePath to myFilePath & "Scripts:Xml rules:glue code.scpt"
        set myGlueCode to load script file myFilePath
        set myRuleSet to {my ConvertToAttribute()}
    end tell
    tell myGlueCode
        __processRuleSet(myRootXML, myRuleSet, {})
    end tell
end run
to ConvertToAttribute()
    script ConvertToAttribute
        property name : "ConvertToAttribute"
        property xpath : "/devices/device/part_number"
        on apply(myXMLElement, myRuleProcessor)
            tell myGlueCode
                __skipChildren(myRuleProcessor)
            end tell
            global myReturnString
            set myReturnString to "OK"
            tell application "Adobe InDesign CS6"
                try
                    tell myXMLElement
                        convert to attribute ("PartNumber")
                    end tell
                on error myError
                    set myReturnString to myError
                end try
            end tell
            return true
        end apply
    end script
end ConvertToAttribute
```

XML 属性のデータを XML 要素に移動するには、第 12 章「XML」の説明にあるように、`convertToElement` メソッドを使用します。

複数の一致ルールの適用

ある XML ルールの apply 関数で true を返すと、それ以降の XML ルールは XML 要素に適用されなくなります。apply 関数で false を返すと、他のルールが必要に応じて XML 要素に適用されます。次のスクリプトは、false を返す XML ルール apply 関数の例を示しています。このスクリプトには、ドキュメント内のすべての XML 要素に一致する 2 つのルールが含まれています。最初のルールは特定のカラーを適用して false を返しますが、2 番目のルールは（myCounter という変数の状態に基づいて）1 つおきに XML 要素に別のカラーを適用する点が異なります。完全なスクリプトについては、ReturningFalse を参照してください。

```
global myGlueCode, myCounter
on run
    set myCounter to 0
    tell application "Adobe InDesign CS6"
        set myRootXML to first XML element of document 1
        set myApplicationPath to file path
        set myFilePath to file path as string
        set myFilePath to myFilePath & "Scripts:Xml rules:glue code.scpt"
        set myGlueCode to load script file myFilePath
        set myRuleSet to {my ReturnFalse(), my ReturnTrue()}
        set myDocument to document 1
        tell myDocument
            set myColorA to make color with properties {name:"ColorA",
            model:process, color value:{0, 100, 80, 0}}
            set myColorB to make color with properties {name:"ColorB",
            model:process, color value:{100, 0, 80, 0}}
        end tell
    end tell
    tell myGlueCode
        __processRuleSet(myRootXML, myRuleSet, {})
    end tell
end run
to ReturnTrue()
    script ReturnTrue
        property name : "ReturnTrue"
        property xpath : "//*"
        on apply(myXMLElement, myRuleProcessor)
            global myReturnString
            set myReturnString to "OK"
            tell application "Adobe InDesign CS6"
                try
                    tell myXMLElement
                        if myCounter mod 2 = 0 then
                            set fill color of text 1 to "ColorA"
                        end if
                        set myCounter to myCounter + 1
                    end tell
                on error myError
                    set myReturnString to myError
                end try
            end tell
            --Do not process the element with any further matching rules.
            return true
        end apply
    end script
end ReturnTrue
to ReturnFalse()
    script ReturnFalse
        property name : "ReturnFalse"
        property xpath : "//*"
        on apply(myXMLElement, myRuleProcessor)
            global myReturnString
```

```
            set myReturnString to "OK"
            tell application "Adobe InDesign CS6"
                try
                    tell myXMLElement
                        set fill color of text 1 to "ColorB"
                    end tell
                on error myError
                    set myReturnString to myError
                end try
            end tell
            --Leave the XML element available for further matching rules.
            return false
        end apply
    end script
end ReturnFalse
```

XML 要素の検索

既に説明したように、XML ルールでサポートされている XPath のサブセットでは、XML 要素のテキストコンテンツを検索することはできません。この制約を回避するには、属性を使用して目的の XML 要素を検索するか、一致する XML 要素のテキストを検索します。属性を使用して XML 要素を検索する方法を、次のスクリプトに示します。このスクリプトでは、検索した要素のテキストにカラーを適用していますが、実際のスクリプトではより意味のある処理を行うことになります。完全なスクリプトについては、FindXMLElementByAttribute を参照してください。

```
global myGlueCode
on run
    tell application "Adobe InDesign CS6"
        set myFilePath to file path as string
        set myFilePath to myFilePath & "Scripts:Xml rules:glue code.scpt"
        tell document 1
            set myRootXML to first XML element
            set myColorA to make color with properties
            {name:"ColorA", model:process, color value:{0, 100, 80, 0}}
        end tell
    end tell
    set myGlueCode to load script file myFilePath
    set myRuleSet to {my AddAttribute()}
    tell myGlueCode
        __processRuleSet(myRootXML, myRuleSet, {})
    end tell

    set myRuleSet to {my FindAttribute()}
    tell myGlueCode
        __processRuleSet(myRootXML, myRuleSet, {})
    end tell
end run
to AddAttribute()
    script AddAttribute
        property name : "AddAttribute"
        property xpath : "/devices/device/part_number"
        --Adds the content of an XML element to an attribute of
        --the parent of the XML element. This can make finding the
        --element by its contnet much easier and faster.
        on apply(myXMLElement, myRuleProcessor)
            global myReturnString
            set myReturnString to "OK"
            tell application "Adobe InDesign CS6"
                try
                    tell myXMLElement
                        set myString to contents of text 1
                        tell parent
                            make XML attribute with properties
                            {name:"part_number", value:myString}
```

```
                    end tell
                end tell
            on error myError
                set myReturnString to myError
            end try
        end tell
        return true
    end apply
end script
end AddAttribute
to FindAttribute()
    script FindAttribute
        property name : "FindAttribute"
        property xpath : "/devices/device[@part_number = 'DS001']"
        on apply(myXMLElement, myRuleProcessor)
            global myReturnString
            set myReturnString to "OK"
            tell application "Adobe InDesign CS6"
                try
                    tell myXMLElement
                        set fill color of text 1 to "ColorA"
                    end tell
                on error myError
                    set myReturnString to myError
                end try
            end tell
            return false
        end apply
    end script
end FindAttribute
```

findText メソッドを使用して、XML コンテンツを検索およびフォーマットする方法を、次のスクリプトに示します（完全なスクリプトについては、FindXMLElementByFindText を参照してください）。

```
global myGlueCode
on run
    tell application "Adobe InDesign CS6"
        set myFilePath to file path as string
        set myFilePath to myFilePath & "Scripts:Xml rules:glue code.scpt"
        tell document 1
            set myRootXML to first XML element
            set myColorA to make color with properties {name:"ColorA",
            model:process, color value:{0, 100, 80, 0}}
        end tell
    end tell
    set myGlueCode to load script file myFilePath
    set myRuleSet to {my FindByFindTExt()}
    tell myGlueCode
        __processRuleSet(myRootXML, myRuleSet, {})
    end tell
end run
to FindByFindTExt()
    script FindByFindTExt
        property name : "FindByFindText"
        property xpath : "/devices/device/description"
        on apply(myXMLElement, myRuleProcessor)
            global myReturnString
            set myReturnString to "OK"
            tell application "Adobe InDesign CS6"
                set find text preferences to nothing
                set change text preferences to nothing
                set find what of find text preferences to "triangle"
                try
                    tell myXMLElement
```

```
                    set myFoundItems to find text
                    if (count myFoundItems) is greater than 0 then
                        set fill color of text 1 to "ColorA"
                    end if
                end tell
            on error myError
                set myReturnString to myError
            end try
            set find text preferences to nothing
            set change text preferences to nothing
        end tell
        return false
    end apply
end script
end FindByFindTExt
```

`findGrep` メソッドを使用して、XML コンテンツを検索およびフォーマットする方法を、次のスクリプトに示します（完全なスクリプトについては、FindXMLElementByFindGrep を参照してください）。

```
global myGlueCode
on run
    tell application "Adobe InDesign CS6"
        set myFilePath to file path as string
        set myFilePath to myFilePath & "Scripts:Xml rules:glue code.scpt"
        tell document 1
            set myRootXML to first XML element
            set myColorA to make color with properties {name:"ColorA",
            model:process, color value:{0, 100, 80, 0}}
        end tell
    end tell
    set myGlueCode to load script file myFilePath
    set myRuleSet to {my FindByFindGrep()}
    tell myGlueCode
        __processRuleSet(myRootXML, myRuleSet, {})
    end tell
end run
to FindByFindGrep()
    script FindByFindGrep
        property name : "FindByFindGrep"
        property xpath : "/devices/device/description"
        on apply(myXMLElement, myRuleProcessor)
            global myReturnString
            set myReturnString to "OK"
            tell application "Adobe InDesign CS6"
                set find grep preferences to nothing
                set change grep preferences to nothing
                --Find all of the devices the mention both "pulse" and
                --"triangle" in their description.
                set find what of find grep preferences to
                "(?i)pulse.*?triangle|triangle.*?pulse"
                try
                    tell myXMLElement
```

```
                    set myFoundItems to find grep
                    if (count myFoundItems) is greater than 0 then
                        set fill color of text 1 to "ColorA"
                    end if
                end tell
            on error myError
                set myReturnString to myError
            end try
            set find grep preferences to nothing
            set change grep preferences to nothing
        end tell
        return false
    end apply
end script
end FindByFindGrep
```

XML ルールを使用した XML 要素の抽出

XSLT は、XML ファイルからデータのサブセットを抽出する場合によく使用されます。同じことを、XML ルールを使用して行うことができます。一連のサンプル XML 要素を複製して、XML 要素階層の別の場所に移動する方法を、次のサンプルスクリプトに示します。複製した XML 要素は、XML ルールに一致しない XML 構造の場所に追加する必要があることに注意してください。そうしなかった場合は、無限ループに陥る恐れがあります。完全なスクリプトについては、ExtractSubset を参照してください。

```
global myGlueCode
on run
    tell application "Adobe InDesign CS6"
        set myFilePath to file path as string
        set myFilePath to myFilePath & "Scripts:Xml rules:glue code.scpt"
        tell document 1
            set myRootXML to first XML element
            set myXMLTag to make XML tag with properties {name:"VCOs"}
            tell myRootXML
                set myXMLElement to make XML element with properties
                {markup tag:myXMLTag}
            end tell
        end tell
    end tell
    set myGlueCode to load script file myFilePath
    set myRuleSet to {my ExtractVCO()}
    tell myGlueCode
        __processRuleSet(myRootXML, myRuleSet, {})
    end tell
end run
to ExtractVCO()
    script ExtractVCO
        property name : "ExtractVCO"
        property xpath : "/devices/device/type"
        on apply(myXMLElement, myRuleProcessor)
            global myReturnString
            set myReturnString to "OK"
```

```
            tell application "Adobe InDesign CS6"
                try
                    if contents of text 1 of myXMLElement is "VCO" then
                        set myNewXMLElement to duplicate parent of myXMLElement
                    end if
                    move myNewXMLElement to end of XML element -1 of XML element 1
                    of document 1
                on error myError
                    set myReturnString to myError
                end try
            end tell
            return true
        end apply
    end script
end ExtractVCO
```

XML ルールを使用したフォーマットの適用

これまでの XML ルールの例では、XML 要素の検索、XML 要素の並べ替え、XML 要素へのテキストの追加を行う基本的な方法を示しました。XML ルールはスクリプトの一部であるので、テキストフォーマットの適用からページアイテム、ページ、ドキュメントの新規作成まで、ほとんどすべてのアクションを実行できます。次の XML ルールの例では、XML ルールを使用して XML 要素にフォーマットを適用する方法と、XML ルールの一致に基づいて新規ページアイテムを作成する方法を示します。

次のスクリプトでは、XML のサンプルデータに静的テキストを追加し、フォーマットを適用しています（完全なスクリプトについては、XMLRulesApplyFormatting を参照してください）。

```
global myGlueCode
on run
    tell application "Adobe InDesign CS6"
        set myApplicationPath to file path
        set myFilePath to file path as string
        set myFilePath to myFilePath & "Scripts:Xml rules:glue code.scpt"
        tell document 1
            tell view preferences
                set horizontal measurement units to points
                set vertical measurement units to points
                set ruler origin to page origin
            end tell
            set myRootXML to first XML element
            set myColor to make color with properties
            {model:process, color value:{0, 100, 100, 0}, name:"Red"}
            make paragraph style with properties {name:"DeviceName",
            point size:24, leading:24, space before:24, fill color:"Red",
            rule above:true, rule above offset:24}
            make paragraph style with properties {name:"DeviceType",
            point size:12, font style:"Bold", leading:12}
            make paragraph style with properties {name:"PartNumber",
            point size:12, font style:"Bold", leading:12}
            make paragraph style with properties {name:"Voltage",
            point size:10, leading:12}
            make paragraph style with properties {name:"DevicePackage",
            pointSize:10, leading:12}
            make paragraph style with properties {name:"Price", point size:10,
            leading:12, font style:"Bold", space after:12}
        end tell
    end tell
    set myGlueCode to load script file myFilePath
    set myRuleSet to {my ProcessDevice(), my ProcessName(), my ProcessType(), my
ProcessPartNumber(), my ProcessSupplyVoltage(), my ProcessPackageType(), my ProcessPrice(), my
ProcessPackageOne(), my ProcessPackages()}
    tell myGlueCode
        __processRuleSet(myRootXML, myRuleSet, {})
    end tell
```

```
end run
to ProcessDevice()
    script ProcessDevice
        property name : "ProcessDevice"
        property xpath : "/devices/device"
        on apply(myXMLElement, myRuleProcessor)
            global myReturnString
            set myReturnString to "OK"
            tell application "Adobe InDesign CS6"
                try
                    tell myXMLElement
                        insert text as content using return position after element
                    end tell
                on error myError
                    set myReturnString to myError
                end try
            end tell
            return true
        end apply
    end script
end ProcessDevice
to ProcessName()
    script ProcessName


        property name : "ProcessName"
        property xpath : "/devices/device/name"
        on apply(myXMLElement, myRuleProcessor)
            global myReturnString
            set myReturnString to "OK"
            tell application "Adobe InDesign CS6"
                try
                    tell myXMLElement
                        insert text as content using return position after element
                        apply paragraph style using "DeviceName"
                    end tell
                on error myError
                    set myReturnString to myError
                end try
            end tell
            return true
        end apply
    end script
end ProcessName
to ProcessType()
    script ProcessType
        property name : "ProcessType"
        property xpath : "/devices/device/type"
        on apply(myXMLElement, myRuleProcessor)
            global myReturnString
            set myReturnString to "OK"
            tell application "Adobe InDesign CS6"
                try
                    tell myXMLElement
                        insert text as content using "Circuit Type: " position
                        before element
                        insert text as content using return position after element
                        apply paragraph style using "DeviceType"
                    end tell
                on error myError
                    set myReturnString to myError
                end try
            end tell
            return true
        end apply
    end script
```

```
end ProcessType
to ProcessPartNumber()
    script ProcessPartNumber
        property name : "ProcessPartNumber"
        property xpath : "/devices/device/part_number"
        on apply(myXMLElement, myRuleProcessor)
            global myReturnString
            set myReturnString to "OK"
            tell application "Adobe InDesign CS6"
                try
                    tell myXMLElement
                        insert text as content using "Part Number: " position before
                        element
                        insert text as content using return position after element
                        apply paragraph style using "PartNumber"
                    end tell
                on error myError
                    set myReturnString to myError
                end try


            end tell
            return true
        end apply
    end script
end ProcessPartNumber
to ProcessSupplyVoltage()
    script ProcessSupplyVoltage
        property name : "ProcessSupplyVoltage"
        property xpath : "/devices/device/supply_voltage"
        on apply(myXMLElement, myRuleProcessor)
            global myReturnString
            set myReturnString to "OK"
            tell application "Adobe InDesign CS6"
                try
                    tell myXMLElement
                        insert text as content using "Supply Voltage: From: "
                        position before element
                        tell XML element 1
                            insert text as content using " to " position after
                            element
                        end tell
                        tell XML element -1
                            insert text as content using " volts" position after
                            element
                        end tell
                        insert text as content using return position after element
                        apply paragraph style using "Voltage"
                    end tell
                on error myError
                    set myReturnString to myError
                end try
            end tell
            return true
        end apply
    end script
end ProcessSupplyVoltage
to ProcessPackageType()
    script ProcessPackageType
        property name : "ProcessPackageType"
        property xpath : "/devices/device/package/type"
        on apply(myXMLElement, myRuleProcessor)
            global myReturnString
            set myReturnString to "OK"
            tell application "Adobe InDesign CS6"
                try
```

```
                    tell myXMLElement
                        insert text as content using "-" position after element
                    end tell
                on error myError
                    set myReturnString to myError
                end try
            end tell
            return true
        end apply
    end script
end ProcessPackageType
--Add the text "Package:" before the list of packages.
to ProcessPackageOne()
    script ProcessPackageOne
        property name : "ProcessPackageOne"


        property xpath : "/devices/device/package[1]"
        on apply(myXMLElement, myRuleProcessor)
            global myReturnString
            set myReturnString to "OK"
            tell application "Adobe InDesign CS6"
                try
                    tell myXMLElement
                        insert text as content using "Package: " position before
                        element
                        apply paragraph style using "DevicePackage"
                    end tell
                on error myError
                    set myReturnString to myError
                end try
            end tell
            return true
        end apply
    end script
end ProcessPackageOne
--Add commas between the package types and a return at the end of the packages.
to ProcessPackages()
    script ProcessPackages
        property name : "ProcessPackages"
        property xpath : "/devices/device/package"
        on apply(myXMLElement, myRuleProcessor)
            global myReturnString
            set myReturnString to "OK"
            tell application "Adobe InDesign CS6"
                try
                    tell myXMLElement
                        set myIndex to index of myXMLElement
                        if myIndex is not 1 then
                            insert text as content using ", " position before element
                        end if
                    end tell
                on error myError
                    set myReturnString to myError
                end try
```

```
            end tell
            return true
        end apply
    end script
end ProcessPackages
to ProcessPrice()
    script ProcessPrice
        property name : "ProcessPrice"
        property xpath : "/devices/device/price"
        on apply(myXMLElement, myRuleProcessor)
            global myReturnString
            set myReturnString to "OK"
            tell application "Adobe InDesign CS6"
                try


                    tell myXMLElement
                        insert text as content using return position before element
                        insert text as content using "Price: $" position before
                        element
                        insert text as content using return position after element
                        apply paragraph style using "Price"
                    end tell
                on error myError
                    set myReturnString to myError
                end try
            end tell
            return true
        end apply
    end script
end ProcessPrice
```

XML ルールを使用したページアイテムの作成

次のスクリプトでは、新しいページアイテムを作成し、そのページアイテムに XML 要素の内容を挿入し、静的テキストを追加して、フォーマットを適用しています。ここでは、スクリプト内の該当する XML ルールの部分のみを示します。詳しくは、完全なスクリプト（XMLRulesLayout）を参照してください。

最初のルールでは、それぞれの「device」という XML 要素に新規テキストフレームを作成します。

```
to ProcessDevice()
    script ProcessDevice
        property name : "ProcessDevice"
        property xpath : "/devices/device"
        on apply(myXMLElement, myRuleProcessor)
            global myReturnString
            set myReturnString to "OK"
            tell application "Adobe InDesign CS6"
                set myDocument to document 1
                tell myDocument
                    try
                        tell myXMLElement
                            insert text as content using return position after
                            element
                        end tell
                        if (count text frames of page 1) is greater than 0 then
                            set myPage to make page
                        else
                            set myPage to page 1
                        end if
```

```
                    set myBounds to my myGetBounds(myDocument, myPage)
                    set myTextFrame to place into frame myXMLElement on myPage
                    geometric bounds myBounds
                    tell text frame preferences of myTextFrame
                        set first baseline offset to leading offset
                    end tell
                on error myError
                    set myReturnString to myError
                end try
            end tell
        end tell
        return true
    end apply
end script
end ProcessDevice
```

"ProcessType" ルールで、「type」という XML 要素を、ページ上の新規フレームに移動します。

```
to ProcessType()
    script ProcessType
        property name : "ProcessType"
        property xpath : "/devices/device/type"
        on apply(myXMLElement, myRuleProcessor)
            global myReturnString
            set myReturnString to "OK"
            tell application "Adobe InDesign CS6"
                set myDocument to document 1
                tell myDocument
                    try
                        tell myXMLElement
                            insert text as content using "Circuit Type: " position
                            before element
                            insert text as content using return position after
                            element
                            apply paragraph style using "DeviceType"
                        end tell
                        set myPage to page -1 of myDocument
                        set myBounds to my myGetBounds(myDocument, myPage)
                        set myX1 to item 2 of myBounds
                        set myY1 to (item 1 of myBounds) - 24
                        set myX2 to myX1 + 48
                        set myY2 to item 1 of myBounds
                        set myTextFrame to place into frame myXMLElement on myPage
                        geometric bounds {myY1, myX1, myY2, myX2}
                        set fill color of myTextFrame to "Red"
                        tell text frame preferences of myTextFrame
                            set inset spacing to {6, 6, 6, 6}
                        end tell
                    on error myError
                        set myReturnString to myError
                    end try
                end tell
            end tell
            return true
        end apply
    end script
end ProcessType
```

XML ルールを使用した表の作成

`convert element to table` メソッドを使用すれば、XML 要素を表に変換することができます。ただし、このメソッドで表に変換できるのは、すべての XML 要素が特定の XML タグ（行タグと、セルタグまたは列タグ）で構成されている場合のみです。多くの場合、表に変換したい XML データはそのような構造になっていません。列に使用したい XML 要素は、「price」や「part number」などの異質な XML タグである場合が多いです。

この制限に対応するには、表の列として追加したい XML 要素を、コンテナ XML 要素を使用して「ラップ」します。次のスクリプトにその例を示します（XMLRulesTable を参照してください）。この例では、XML ルールを使用して各行の XML 要素を作成しています。

```
to ProcessDevice()
    script ProcessDevice
        property name : "ProcessDevice"
        property xpath : "//device[@type = 'VCO']"
        on apply(myXMLElement, myRuleProcessor)
            global myReturnString
            set myReturnString to "OK"
            tell application "Adobe InDesign CS6"
                set myDocument to document 1
                tell myDocument
                    try
                        set myContainerElement to XML element -1 of XML element -1
                        of XML element 1
                        tell myContainerElement
                            set myNewElement to make XML element with properties
                            {markup tag:"Row"}
                        end tell
                    on error myError
                        set myReturnString to myError
                    end try
                end tell
            end tell
            return false
        end apply
    end script
end ProcessDevice
```

次のルールでは、行 XML 要素内のコンテナ要素にその内容を移動してフォーマットしています。

```
to ProcessPrice()
    script ProcessPrice
        property name : "ProcessPrice"
        property xpath : "//device[@type = 'VCO']/price"
        on apply(myXMLElement, myRuleProcessor)
            tell myGlueCode
                __skipChildren(myRuleProcessor)
            end tell
            global myReturnString
            set myReturnString to "OK"
            tell application "Adobe InDesign CS6"
                set myDocument to document 1
                tell myDocument
                    try
                        set myRootElement to XML element 1
                        set myVCOs to XML element -1 of myRootElement
                        set myTable to XML element -1 of myVCOs
                        set myContainerElement to XML element -1 of myTable
                        tell myContainerElement
                            set myNewElement to make XML element with properties
                            {markup tag:"Column"}
```

```
                    end tell
                    set myXMLElement to move myXMLElement to beginning of
                    myNewElement
                    tell myXMLElement
                        insert text as content using "$" position before element
                    end tell
                on error myError
                    set myReturnString to myError
                end try
            end tell
        end tell
        return true
    end apply
end script
end ProcessPrice
```

指定された XML 要素をすべて「ラップ」したら、コンテナ要素を表に変換します。

```
tell myContainerElement
    set myTable to convert to table row tag myRowTag cell tag myColumnTag
end tell
```

XML ルールプロセッサーオブジェクトのスクリプティング

glue code.scpt で一連のユーティリティ関数が既に提供されていますが、XML ルールプロセッサーオブジェクトをスクリプトで直接操作することもできます。これは、XML ルールのサポート関数を開発したり、XML ルールプロセッサーを別の方法で使用したりする場合に必要になります。

スクリプトを使用して、XML ルールのコンテキストの外部で XML 要素を操作した場合、XPath を使用して要素を探すことはできません。しかし、一致する XML 要素を単純に返す XML ルールを作成して、XML ルールプロセッサーでルールを適用することが可能です。次のスクリプトにその例を示します （このスクリプトでも、以前の節のサンプルスクリプトと同じ XML データファイルを使用します）。完全なスクリプトについては、XMLRulesProcessor を参照してください。

```
set myXPath to {"/devices/device"}
set myXMLMatches to mySimulateXPath(myXPath)
--At this point, myXMLMatches contains all of the XML elements
--that matched the XPath expression provided in myXPath.
--In a real script, you could now process the elements.
--For this example, however, we'll simply display a message.
display dialog ("Found " & (count myXMLMatches) & " matching XML elements.")
on mySimulateXPath(myXPath)
    set myMatchingElements to {}
    tell application "Adobe InDesign CS6"
        set myRuleProcessor to make XML rule processor with properties
        {rule paths:myXPath}
        set myDocument to document 1
        set myRootXMLElement to XML element 1 of myDocument
        tell myRuleProcessor
            set myMatchData to start processing rule set start element
            myRootXMLElement
            repeat until myMatchData is nothing
                local myMatchData
                local myMatchingElements
                set myXMLElement to item 1 of myMatchData
                set myMatchingElements to myMatchingElements & myXMLElement
                set myMatchData to find next match
            end repeat
        end tell
        return myMatchingElements
    end tell
end mySimulateXPath
```

14 変更のトラック

章の更新ステータス	
CS6	変更なし

原稿作成や編集作業の進行中に、ドキュメントの変更のトラック、表示／非表示、適用、取り消しを実行できます。すべての変更は記録されていて画面で確認できるので、ドキュメントの見直しが簡単に行えます。

このチュートリアルでは、変更のトラックに関する一般的な操作をスクリプトで実行する方法について説明します。

ここでは、読者が『Adobe InDesign スクリプティングチュートリアル』を既に読んでおり、スクリプトを作成、インストール、実行する方法を理解しているものとして説明を行います。また、InDesign でテキストを操作する方法や、組版に関する基本的な用語を理解しているものとしています。

変更のトラック

ここでは、スクリプトを使用して、トラックされた変更に移動したり、変更を適用および取り消しする方法について説明します。

既存のストーリーに含まれているテキストの追加や削除、移動が行われると、その変更がゲラビューおよびストーリービューに示されます。

トラックされた変更への移動

トラックされた変更の記録がストーリーに含まれている場合、ユーザーはトラックされた変更に順次移動することができます。トラックされた変更に移動する方法を、次のスクリプトに示します（完全なスクリプトについては、GetTrackchange を参照してください）。

次のスクリプトでは、変更のインデックス番号を使用して、変更に対して繰り返し処理を行っています。

```
tell application "Adobe InDesign CS6"
    set myDocument to document 1
    tell myDocument
        set myStory to story 1
        tell myStory
            if (track changes) is true then
                set myChangeCounter to count
                set myChange to change 1
            end if
        end tell
    end tell
end tell
```

トラックされた変更の適用と取り消し

ストーリーに対する変更が行われた場合、変更のトラック機能を使用してすべての変更を見直したり、ストーリーにすべての変更を組み込むかどうかを決定できます。ユーザーは、すべてのユーザーが行った変更（テキストの追加や削除、移動）を適用および取り消しすることができます。

次のスクリプトでは、変更を適用します（完全なスクリプトについては、AcceptChange を参照してください）。

```
tell application "Adobe InDesign CS6"
    set myDocument to document 1
    tell myDocument
        set myStory to story 1
        tell  myStory
            set myChange = myStory change 1
            tell  myChange
                accept
            end tell
        end tell
    end tell
end tell
```

次のスクリプトでは、変更を取り消します（完全なスクリプトについては、RejectChange を参照してください）。

```
tell application "Adobe InDesign CS6"
    set myDocument to document 1
    tell myDocument
        set myStory to story 1
        tell  myStory
            set myChange = myStory change 1
            tell  myChange
                reject
            end tell
        end tell
    end tell
end tell
```

トラックされた変更に関する情報

変更の情報には日付や時間などが含まれています。次のスクリプトでは、トラックされた変更の情報を表示します（完全なスクリプトについては、GetChangeInfo を参照してください）。

```
--Shows how to get track change informations.
tell application "Adobe InDesign CS6"
    set myDocument to document 1
    tell myDocument
        set myStory to story 1
        tell myStory
            set myChange to change 1
            tell myChange
            -- change type (inserted text/deleted text/moved text, r/o)
                set myTypes to change type
                set myCharacters to characters
                set myDate to date
                set myInsertionPoints to insertion points
                set myLines to lines
            -- paragraphs A collection of paragraphs.
                set myParagraphs to paragraphs
                set myStoryOffset to story offset
                set myTextColumns to text columns
                set myTextStyleRanges to text style ranges
                set myTextsetiableInstances to text variable instances
            -- The user who made the change. Note: Valid only when changes is true.
                set myUserName to user name
                -- Words  A collection of words
                set myWords to words
            end tell
        end tell
    end tell
end tell
```

変更のトラックの環境設定

変更のトラックの環境設定とは、変更のトラックに関するユーザー設定です。例えば、トラックされる変更の種類（テキストの追加や削除、移動）を定義できます。トラックされた変更の種類別の外観を指定したり、マージンの色付きの変更バーで変更を識別したりできます。これらの環境設定を設定して取得する方法を、次のスクリプトに示します（完全なスクリプトについては、GetChangePreference を参照してください）。

```
tell application "Adobe InDesign CS6"
    set myTrackChangesPreference to track changes preferences
    tell myTrackChangesPreference
        --  added background color choice (change background uses galley background color/change
background uses user color/change background uses change pref color) : The background color option
for added text.
        set myAddedBackgroundColorChoice to added background color choice
        set added background color choice to change background uses change pref color

        --added text color choice (change uses galley text color/change uses change pref color) :
The color option for added text.
        set myAddedTextColorChoice to added text color choice
        set added text color choice to change uses change pref color

        --background color for added text (any) : The background color for added text, specified as
an InCopy UI color. Note: Valid only when added background color choice is change background uses
change pref color.
        set myBackgroundColorForAddedText to background color for added text
        set background color for added text to gray

        --background color for deleted text (any) : The background color for deleted text,
specified as an InCopy UI color. Note: Valid only when deleted background color choice is change
background uses change pref color.
        set myBackgroundColorForDeletedText to background color for deleted text
        set background color for deleted text to red

        --background color for moved text (any) : The background color for moved text, specified as
an InCopy UI color. Note: Valid only when moved background color choice is change background uses
```

```
change pref color.
        set myBackgroundColorForMovedText to background color for moved text
        set background color for moved text to pink
        --change bar color (any) : The change bar color, specified as an InCopy UI color.
        set myChangeBarColor to change bar color
        set change bar color to charcoal
        --deleted background color choice (change background uses galley background color/change
background uses user color/change background uses change pref color) : The background color option
for deleted text.
        set myDeletedBackgroundColorChoice to deleted background color choice
        set deleted background color choice to change background uses change pref color

        --deleted text color choice (change uses galley text color/change uses change pref color) :
The color option for deleted text.
        set myDeletedTextColorChoice to deleted text color choice
        set deleted text color choice to change uses change pref color

        --location for change bar (left align/right align) : The change bar location.
        set myLocationForChangeBar to location for change bar
        set location for change bar to left align

        --marking for added text (none/strikethrough/underline single/outline) : The marking that
identifies added text.
        set myMarkingForAddedText to marking for added text
        set marking for added text to strikethrough

        --marking for deleted text (none/strikethrough/underline single/outline) : The marking
that identifies deleted text.
        set myMarkingForDeletedText to marking for deleted text
        set marking for deleted text to underline single

        --marking for moved text (none/strikethrough/underline single/outline) : The

marking that identifies moved text.
        set myMarkingForMovedText to marking for moved text
        set marking for moved text to outline

        --moved background color choice (change background uses galley background color/change
background uses user color/change background uses change pref color) : The background color option
for moved text.
        set myMovedBackgroundColorChoice to moved background color choice
        set moved background color choice to change background uses galley background color

        -- moved text color choice (change uses galley text color/change uses change pref color) :
The color option for moved text.
        set myMovedTextColorChoice to moved text color choice
        set moved text color choice to change uses change pref color

        -- If true, displays added text.
        set myShowAddedText to show added text
        set show added text to true

        --  If true, displays change bars.
        set myShowChangeBars to show change bars
        set show change bars to true

        -- If true, displays deleted text.
        set myShowDeletedText to show deleted text
        set show deleted text to true

        -- If true, displays moved text.
        set myShowMovedText to show moved text
        set show moved text to true

        -- If true, includes deleted text when using the Spell Check command.
        set mySpellCheckDeletedText to spell check deleted text
```

```
        set spell check deleted text to true

        --The color for added text, specified as an InCopy UI color. Note: Valid only when added
text color choice is change uses change pref color.
        set myTextColorForAddedText to text color for added text
        set text color for added text to blue

        --  text color for deleted text (any) : The color for deleted text, specified as an InCopy
UI color. Note: Valid only when deleted text color choice is change uses change pref color.
        set myTextColorForDeletedText to text color for deleted text
        set text color for deleted text to yellow

        --  The color for moved text, specified as an InCopy UI color. Note: Valid only when moved
text color choice is change uses change pref color.
        set myTextColorForMovedText to text color for moved text
        set text color for moved text to green
    end tell
end tell
```