gusuku Customine

カスタマインの歩き方

レコード更新編

はじめに（この冊子の目的）

使いこなすととても便利な gusuku Customine(以降カスタマインと表記)。しかし、実現できることが多すぎるがゆえに使っていただいているお客様より「難しい」というお声をいただくこともあります。そこで、実際に役立つカスタマイズを作成しながらカスタマインの使い方にも慣れていただくための教材を用意しました。

この冊子では、様々なカスタマイズで利用可能な「レコード更新」機能にフォーカスして、「どのような要件のときに、どのやることを使用するべきか」を理解していただくことを目的とします。様々な「レコード更新」方法を習得していただくことで、実務に汎用的に活用していただければと思います。

この冊子で理解できること

レコード追加・レコード更新に関する「やること」の違い
「やること」ごとの注意点・制約
要件に応じた「やること」の選び方

目次

1. はじめに理解していただきたいこと

1-1. 本書で使用する用語

本書ではカスタマインの画面や用語を以下のように記載します。

アクション

カスタマインの設定における 1 単位です。「やること」と「条件」を組み合わせた 1 セットが 1 つのアクションとなります。

やること

カスタマイン設定画面の左側で選択する「やること」は、本書では [やること：フィールドを無効化する] のように記載します。

条件

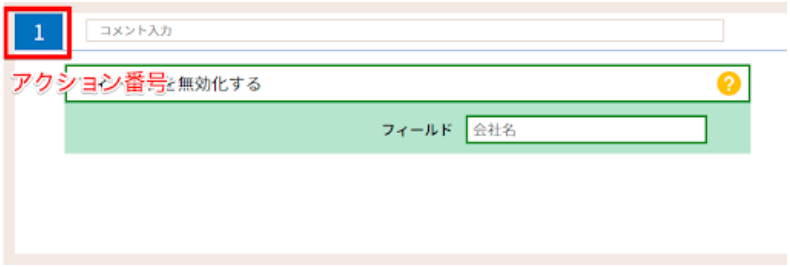
カスタマイン設定画面の右側で選択する「条件」は、本書では＜条件：一覧画面を表示した時＞のように記載します。

追加条件

「条件」の下部で「条件を追加」を選択して設定する「追加条件」は、本書では＜追加条件：フィールド値が特定の値ならば＞のように記載します。

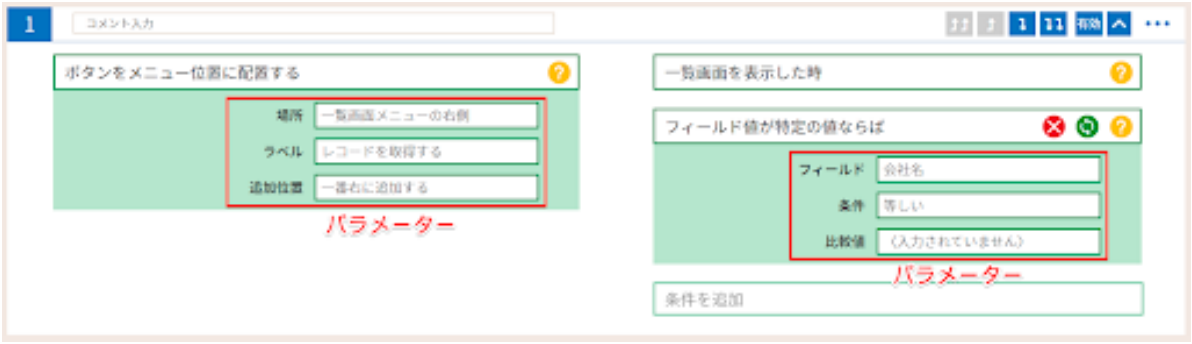
アクション番号

カスタマイン設定画面で各アクション（設定）の左上の数字です。本書内で【アクション：1】と記載した場合には、アクション番号 1 番の設定を指します。



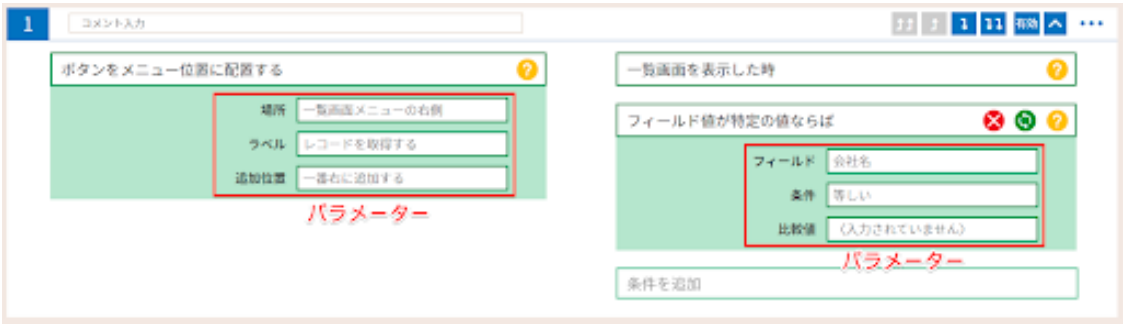
パラメーター

設定する「やること」や「条件」に応じて設定する項目のことをパラメーターと記載します。本書内では《パラメーター：場所》のように記載します。



1-2. カスタマインの基本的な構造

カスタマインの基本的な構造は「〜の時に〜ならば〜する」を [やること] と < 条件 > を組み合わせて



という形で設定します。

この < 条件 > のドキュメント (https://docs-customine.gusuku.io/ja/conditions/) には「(追加条件)」と記載されている一覧があります。これは「〜ならば」に該当するもので、< 条件を追加 > の箇所で指定します。
< 条件 > と < 条件を追加 > は指定する場所が違うのでご注意ください。

1-3. カスタマインで作成したカスタマイズが動く仕組み

カスタマインで作成した画面のカスタマイズは、ユーザーの画面操作に基づいて動く仕組みとなっています。

1-3-1. 画面ごとの特性

kintone にはいくつかの種類の画面があり、その画面ごとにできることが異なります。この画面ごとの特性を理解することで、よりスムーズにカスタマイズを作成できるようになります。

一覧画面は一画面で同時に複数のレコードを閲覧できる、表示に特化した画面です。そのため、カスタマイズにおいてもテーブルデータを表示できる一覧画面の作成など、画面表示に関するカスタマイズに適しています。鉛筆ボタンを押すことで編集もできますが、編集画面・追加画面と比較すると多くの制約があります。編集する際には 1 レコード単位の編集しかできず、編集後は < 条件:

一覧画面を表示した時> に該当しないのでカスタマイズが動作しません。また、一覧の編集画面以外の方法で値を変更した場合はリロードしない限り表示は変わりません。

追加・編集画面ではフィールド値を変更することが出来ますが、保存ボタンを押すまでは変更が確定していないのでキャンセルボタンを押すと変更内容は破棄されます。

詳細画面は kintone の仕様ではフィールド値の更新は出来ないのですが、カスタマインが裏側でレコード更新処理をする事でフィールド値を更新しています。この場合は一覧画面での更新時と同じく値は変わっていますが表示は変わらないのでリロードが必要になります。

1-3-2. 同じやることでも画面によって動作が違う

例として、[やること：**フィールドに値をセットする**] というやること 1 つをとっても、以下のように画面によって違いがあります。

追加画面の場合
　フィールドに値がセットされて、表示も変わる

編集画面の場合
　フィールドに値がセットされて、表示も変わる

詳細画面の場合
　フィールドに値はセットされているが、表示は変わらない
　表示を更新するためにはリロードする必要がある

一覧画面の場合
　画面に表示されている全レコードのフィールドに値がセットされて、表示は変わらない
　表示を更新するためにはリロードする必要がある

一覧画面の鉛筆を押した編集画面
　編集状態になっているレコードのフィールドに値がセットされて、表示も変わる

フィールド値がリアルタイムに変わるのは、kintone での操作と同じく追加画面・編集画面のみということにご注意ください。

コラム：フィールド名とフィールドコード

kintone の仕組みとして、フィールド値を参照する場合は「フィールドコード」を指定します。
kintone のフィールドには「フィールド名」と「フィールドコード」があるのでご注意ください。

簡単な見分け方としては、フィールドの個別設定画面で上にあるのが「フィールド名」で下にあるのが「フィールドコード」です。

「フィールドコード」を明示的に指定しないと「文字列 _1 行 _1」のように自動でフィールドコードが設定されるため、フィールドコードがどのフィールドを指すのか判別しづらくなります。そのため、フィールド名と同じ名称をつけたり、「フィールド名 _ アプリ名」のようにフィールドコードからフィールド名を類推しやすい名前にしておくことをおすすめします。

1-3-3. 自分が操作しているアプリでカスタマイズは動く

カスタマイズを作る際には「どのアプリで操作をしたときにカスタマイズを動かしたいか」を意識して作成する必要があります。

先程の「フィールドに値をセット」だと、表示しているアプリのフィールドに対してセットされますが、例えばこれを仮に別アプリのフィールドにセットしたい場合は「やること」が変わります。(「レコードを更新する(キーの値をフィールドで指定)」など)

コラム：これから「保存」するレコードのみが対象

カスタマインで作成したカスタマイズは、今後操作するときに適用されるものであるため、過去に登録したレコードには適用されません。

例えばあるアプリを作成ししばらく運用した後に、レコード保存後にルックアップを自動更新するカスタマイズを作成したとしても、レコードを 1 つずつ保存していかないとルックアップは更新されません。ですので、カスタマイズ作成前に作成済みのレコードに対して更新しようとすると、大変な手間がかかり実用的ではありません。このような場合には別の仕組みで更新する必要があります。

ルックアップの場合は、ルックアップフィールドの値を更新すれば取得できるので、全レコードを取得してルックアップフィールドだけを書き出せば可能です。また、CSV 読み込みで一括更新する方法でも可能です。

同じような話で、検索用文字列作成があります。これもレコード保存時に文字列を作成しますが、過去のレコードは再保存が必要です。そして、検索文字列の場合は CSV 読み込みでは作成できないので、何らかのカスタマイズが必要になります。
本冊子では詳細な説明を記載しませんが、既に存在しているデータに対して検索用文字列の生成を行いたい場合には、こちらのページをご参照ください。

大量のレコードを安全に処理する方法：検索文字列を作成する事例
（紙の場合：サポートサイトで「検索文字列を作成する事例」で検索してください）
https://support.gusuku.io/ja-JP/support/solutions/articles/36000268083

また、kintone にはユーザーの画面操作を通知する「イベント」という仕組みがあります。カスタマインで作成したカスタマイズは基本的に、この「イベント」に基づいて動きます。

1-3-4. イベント

イベントとは、「ユーザーの画面操作したタイミングでプログラムを動かすための kintone の仕組み」であり、一例として以下のようなイベントが定義されています

1. レコード追加画面を開いた時
2. プロセス管理のアクションを実行した時
3. フィールドの値が変わった時

kintone の「イベント」については cybozu developer network のヘルプをご確認ください。
https://developer.cybozu.io/hc/ja/articles/360000361686

カスタマインにおける「条件」は主にこの「イベント」を日本語に置き換えたものです。また、より使い勝手を向上するために kintone のイベント以外のものも「条件」として利用できるようになっています。
カスタマイズを作成するときには「実現したいこと」とともに「どのイベント」でその処理を実行したいかを意識し、「実現したいこと」→「やること」、「イベント」→「条件」に置き換えていただくとスムーズにカスタマイズが作れます。

例）レコード一覧画面表示後に注意すべき案件を色を変えて目立たせたい

実現したいこと：案件を色を変えて目立たせたい
イベント：レコード一覧画面表示後
↓
[やること：フィールド文字色を変更する]
< 条件：一覧画面を表示した時 >

例）入力時にユーザーに警告を促すために特定のフィールドの背景色を変えたい

実現したいこと：背景色を変えて警告を促したい
イベント：フィールド値変更
↓
[やること：フィールド背景色を変更する]
< 条件：フィールドの値を編集して値が変わった時 >

また、イベントごとの特性や制約を理解することで、kintone の制約上実現できないことでも代替手段を検討することができます。

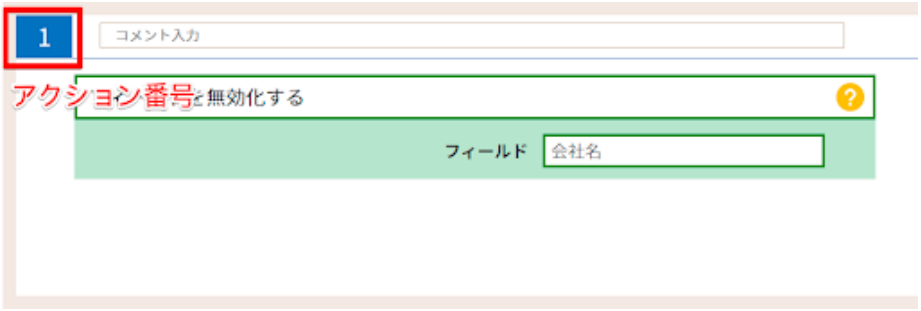
kintone の特性や制約によって実現できないことの例をあげますと、

1 . kintone 側でイベントがない場合
2 . イベントが発生しない場合
3 . そのイベントでは実行できないカスタマイズの場合

などがあり、例えば「フォームブリッジでデータが登録されたことを契機にカスタマイズを動かしたい」という要件は、「2. イベントが発生しない場合」に該当するためカスタマイズが動きません。
また、上記以外でも kintone 基本機能の「ファイルから読み込む」を使ってデータを読み込んだ際にもカスタマイズが動きませんが、これも kintone の制約によるものです。

1-4. 実行順序

カスタマインでアクションを作成すると、番号が自動的に付与されます。
これはアクション番号と呼ばれるもので、カスタマイズの実行順とは関係ありません。

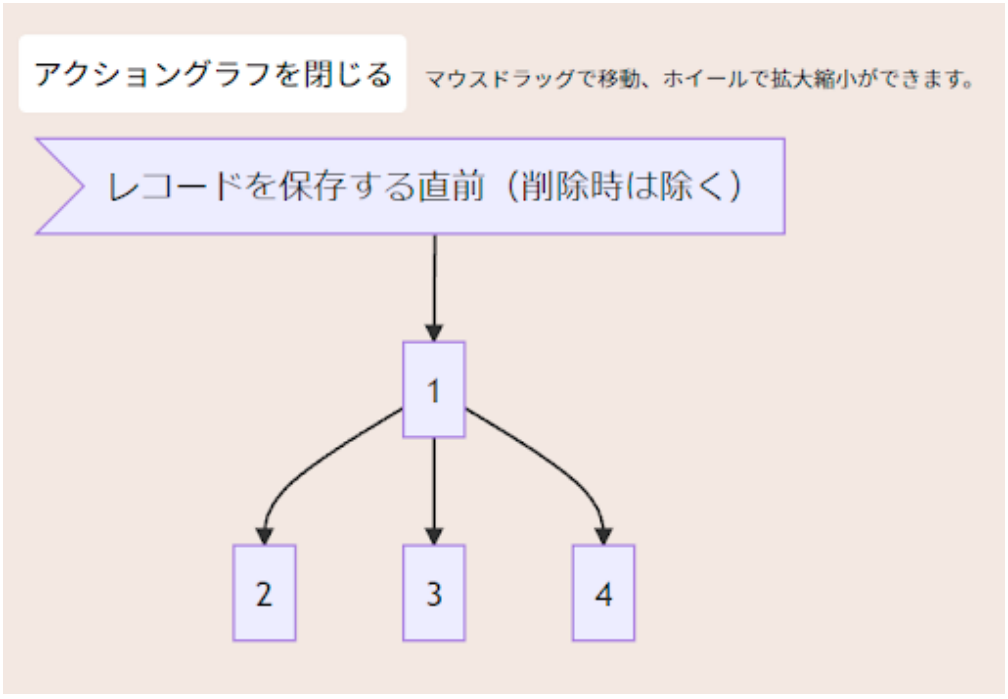


全てのアクションは何かの「イベント」を起点に処理が開始され、
「他のアクションの実行が完了した時」や「確認・入力ダイアログで「OK」を押した時」などで繋がっている順に処理されます。

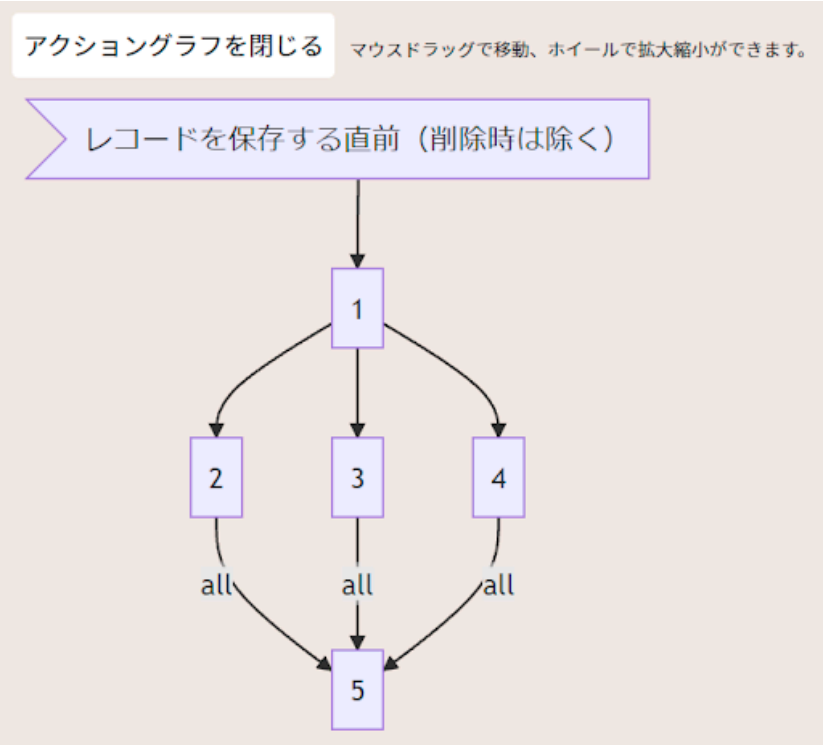
アクションの実行順をわかりやすく確認するにはアクショングラフを表示していただくのが良く、カスタマイズ作成途中や作成後に表示してこまめに確認することをおすすめします。
なお、上手く動かないケースでよくあるのが、線で繋がっていないアクションの結果を参照したり、アクションの繋ぎ先を間違えている場合です。

特にアクションを複製した場合に設定を変え忘れているというケースは良くあります。

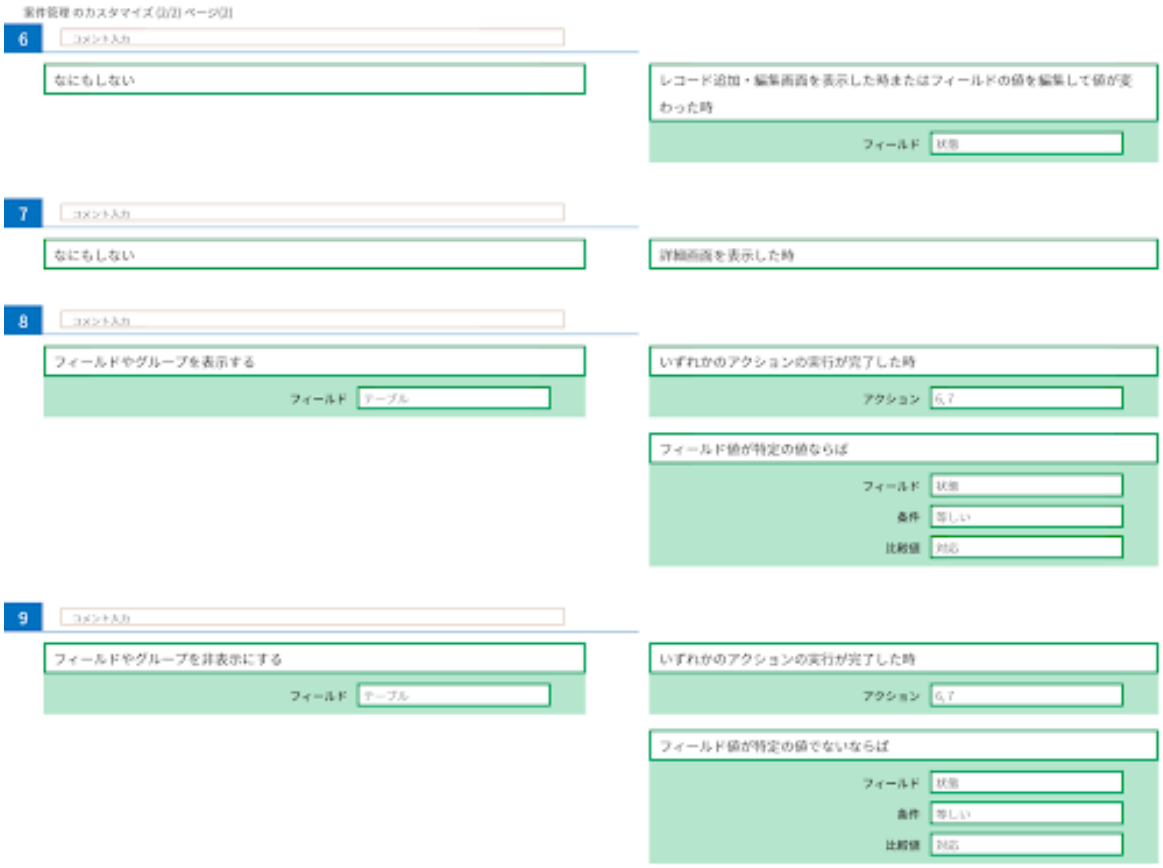
また、以下のように＜条件：他のアクションの実行が完了した時＞で同じアクション番号を指定したアクションが複数ある場合は、並行して処理されます。

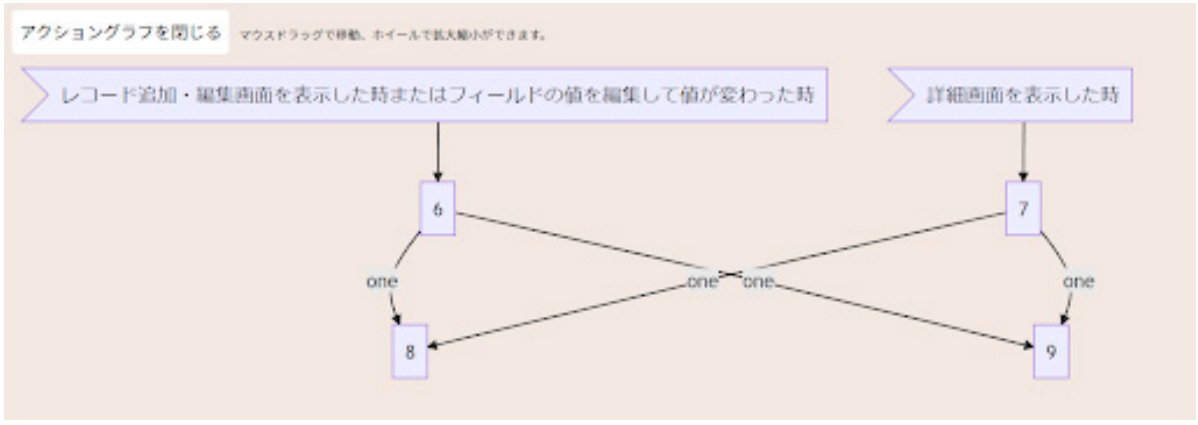


また、これら複数のアクションが全て終わった時に次のアクションを続ける場合は、< 条件：他のアクションの実行が完了した時 > で全てのアクションを指定してください。



その他にも＜条件：いずれかのアクションの実行が完了した時＞を使うと、指定したアクションのいずれかが完了した時にアクションが処理されます。
この条件は例えば、入力した金額によって処理を分ける場合などのようにどれか１つが実行された後、同じ処理が続くような場合に使います。





アクショングラフの詳しい使い方は「**6-4. 原因究明に役立つカスタマインの便利な機能**」に記載がありますので、併せてご確認ください。

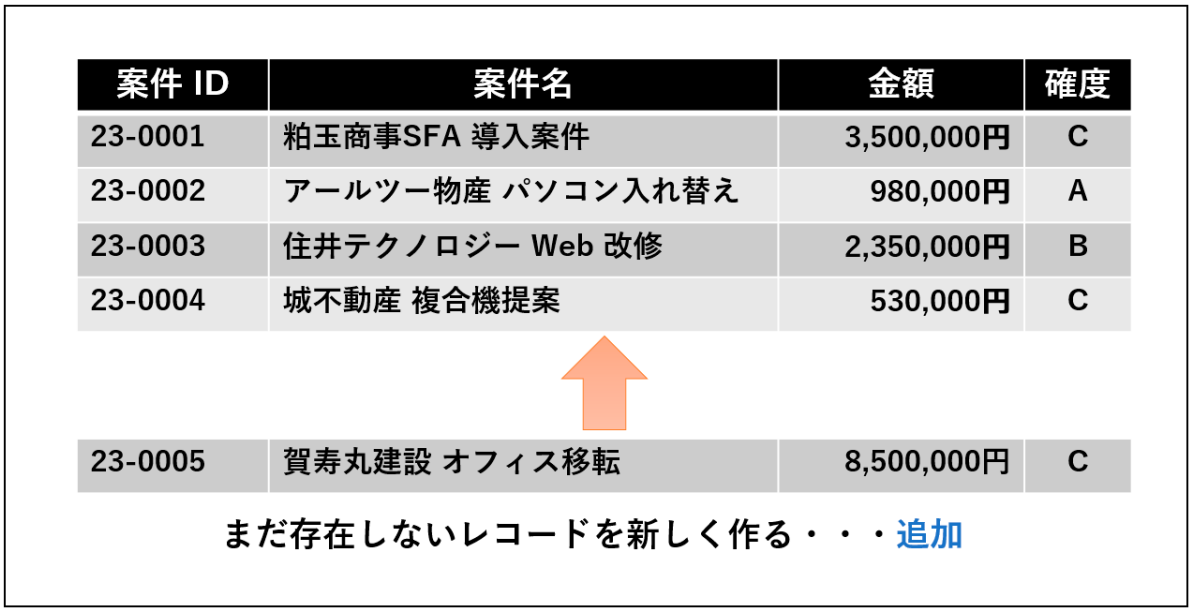
2. レコード操作に関する基礎知識の整理

具体的な「やること」の解説に入る前に、カスタマインでレコードを扱う上で必要となる用語を整理します。

2-1. レコード追加、レコード更新

レコード追加

kintone のアプリ上に新しいレコードを登録･作成することを「レコード追加」と呼びます。本冊子の例では、「活動履歴」アプリで登録した内容をもとにして「案件管理」アプリに転記したいと考えています。そのため､「レコード追加」の機能を使用します。

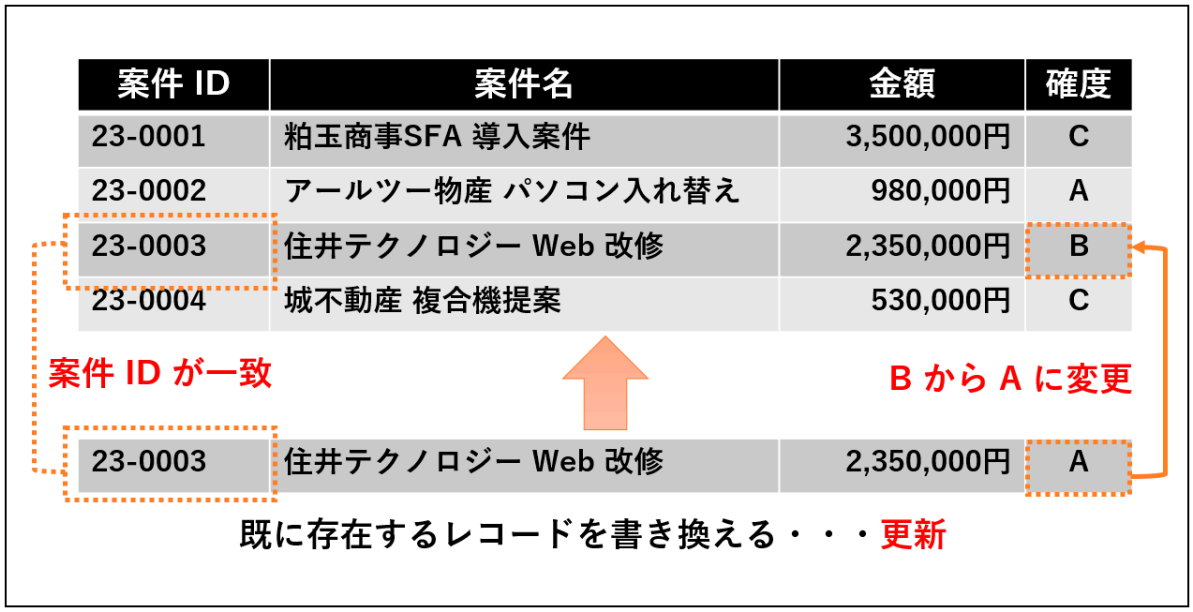


レコード更新

kintone のアプリ上に既に存在するレコードを書き換える（上書きする）ことを「レコード更新」と呼びます。

上記の通り、本冊子の例では「活動履歴」アプリで登録した内容をもとにして「案件管理」アプリに転記したいのですが､既に同一の案件が存在しているケースがありえます。そのときには、新しいレコードを作成（レコード追加）するのではなく､既に登録されているレコードを書き換えたいので「レコード更新」の機能を使用します。

「レコード更新」のときには、既に該当のレコードが登録されているか否か？を判断するための値が必要となります。それを「キー」と呼びます。
この例では、「案件 ID」フィールドが「キー」となります。そのため、「案件 ID」が既に登録されている場合には同一の案件とみなしてレコードを更新を行います。



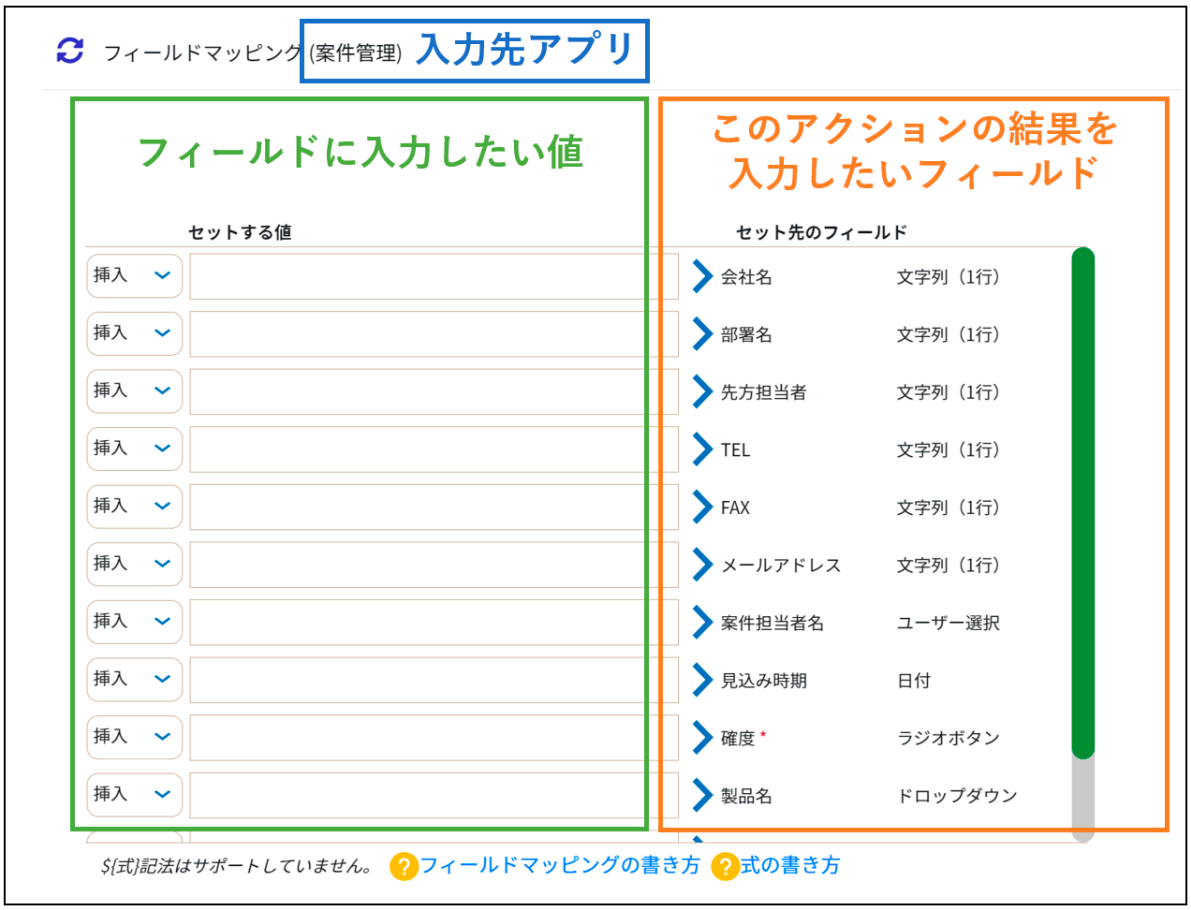
2-2. フィールドマッピングの使い方

フィールドマッピングとは

複数のフィールドにまとめて値を入力するための機能を「フィールドマッピング」と呼びます。「フィールドマッピング」は本書で取り上げているレコード追加・更新以外でも [やること：**フィールド値をまとめてセットする**] などのやることでも使用します。
フィールドマッピング画面の使用方法は**ドキュメント**にも記載がありますが、以下に簡単な例とともに紹介します。

フィールドマッピング画面の見方

フィールドマッピング画面では、画面**右側に指定されたフィールド**に**画面左側で指定した値**を入力します。また、**画面上部の括弧内のアプリ名が入力先のアプリ**を指しています。



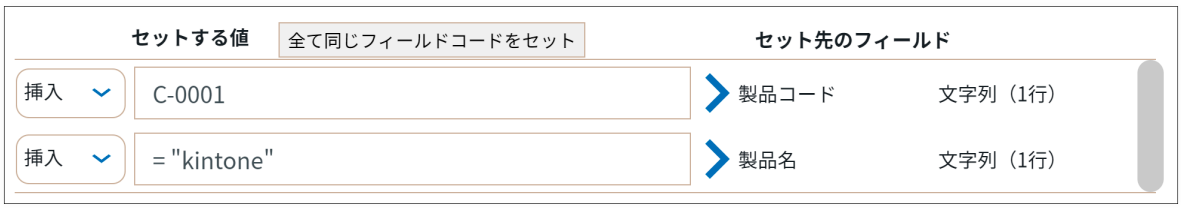
入力できる値

フィールドマッピング画面の左辺では以下の値が指定可能です。

固定値

常に固定の値を指定したい場合には以下のように値を直接記述、もしくは = “ 値 ” の形式でしています。（どちらの記述方法も可能です）

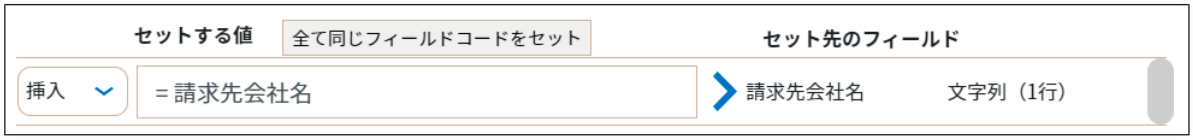
以下の例では、常に製品コードに「C-0001」、製品名に「kintone」が入力されます。



フィールド

他のフィールドに入力されている値を指定したい場合には、以下のように「挿入」のプルダウンよりフィールドを指定します。

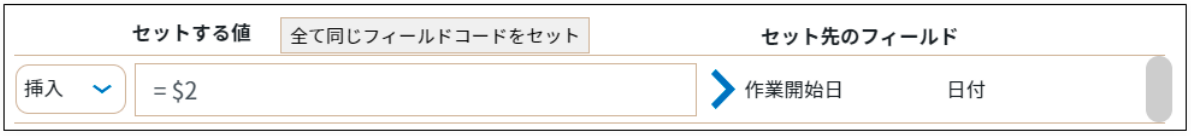
以下の例は、「納品先会社名」というフィールドに対し、「請求書会社名」フィールドの値を入力することを示しています。



他のアクションの結果

他のアクションの実行結果を使用することが可能です。例えば、「特定の日付を取得する」の実行結果で得られる日付や、「入力ダイアログを表示する」によってユーザーに入力された値など様々な指定が可能です。

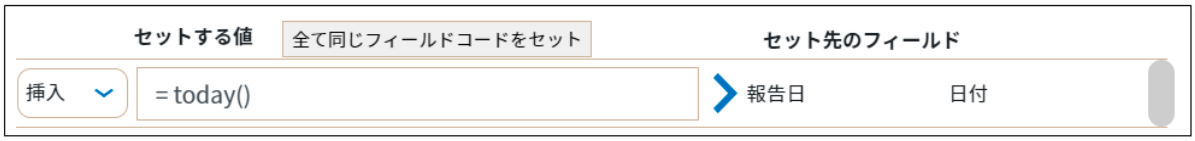
「他のアクションの結果」を指定した場合、以下のように表示されます。「$2」は 2 番のアクションの実行結果を意味しています。



関数

カスタマインで利用可能な関数を指定可能です。指定可能な関数はドキュメント（関数一覧）のとおりです。

以下の例では、「報告日」フィールドに「today 関数」によって本日の日付を入力しています。



3. レコード追加、レコード更新に関する「やること」一挙解説

想定するアプリと解説の流れ

ここからは実際のカスタマイズを作成しながら各やることの利用例、特徴を解説します。それぞれの違いをわかりやすくするために共通のアプリ、ほぼ共通のカスタマイズを作成していきます。

使用するアプリ

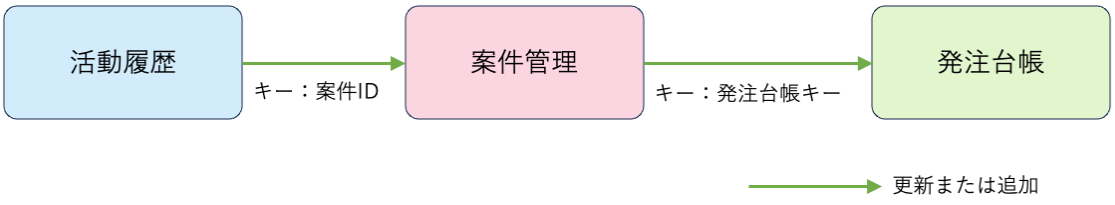
『案件管理』アプリ、『活動履歴』アプリ、『発注台帳』アプリを使用します。

アプリ運用の想定とレコード追加・レコード更新の意味

営業担当者が『活動履歴』アプリに活動業務を登録します。登録というのは kintone でいうとレコード追加のことです。以降は「レコード追加」で説明します。

『活動履歴』アプリにレコードが追加されたとき、まだ『案件管理』アプリに登録がない場合は、『案件管理』アプリに新規案件としてレコード追加します。

『活動履歴』アプリにレコードが追加されたとき、すでに『案件管理』アプリに登録がある場合は、『案件管理』アプリの既存レコードを更新します。これを以降は「レコード更新」と説明します。



作成するカスタマイズ（3-1 ～ 3-9）

3-1 から 3-9 までで作成するカスタマイズは『活動履歴』アプリまたは『案件管理』アプリに適用します。

『発注台帳』アプリは 3-4、3-5 で作成するカスタマイズの更新先アプリとなっています。

カスタマイズの結果の確認方法

ここでは、レコード追加やレコード更新の結果は、実際のアプリのレコードを見ることによって確認することにします。

では、１つ１つのやることについて解説していきます。

3-1. レコードを追加する

一番簡単かつシンプルに利用できるのが [やること：**レコードを追加する**] です。

[やること：**レコードを追加する**] は、レコードを追加する [やること] です。レコードを更新したい場合は、3-2 以降で紹介する別の [やること] を使ってください。設定は、レコードを追加するアプリの指定と、値のセットに関わるマッピング指定の２つだけのため非常にシンプルです。

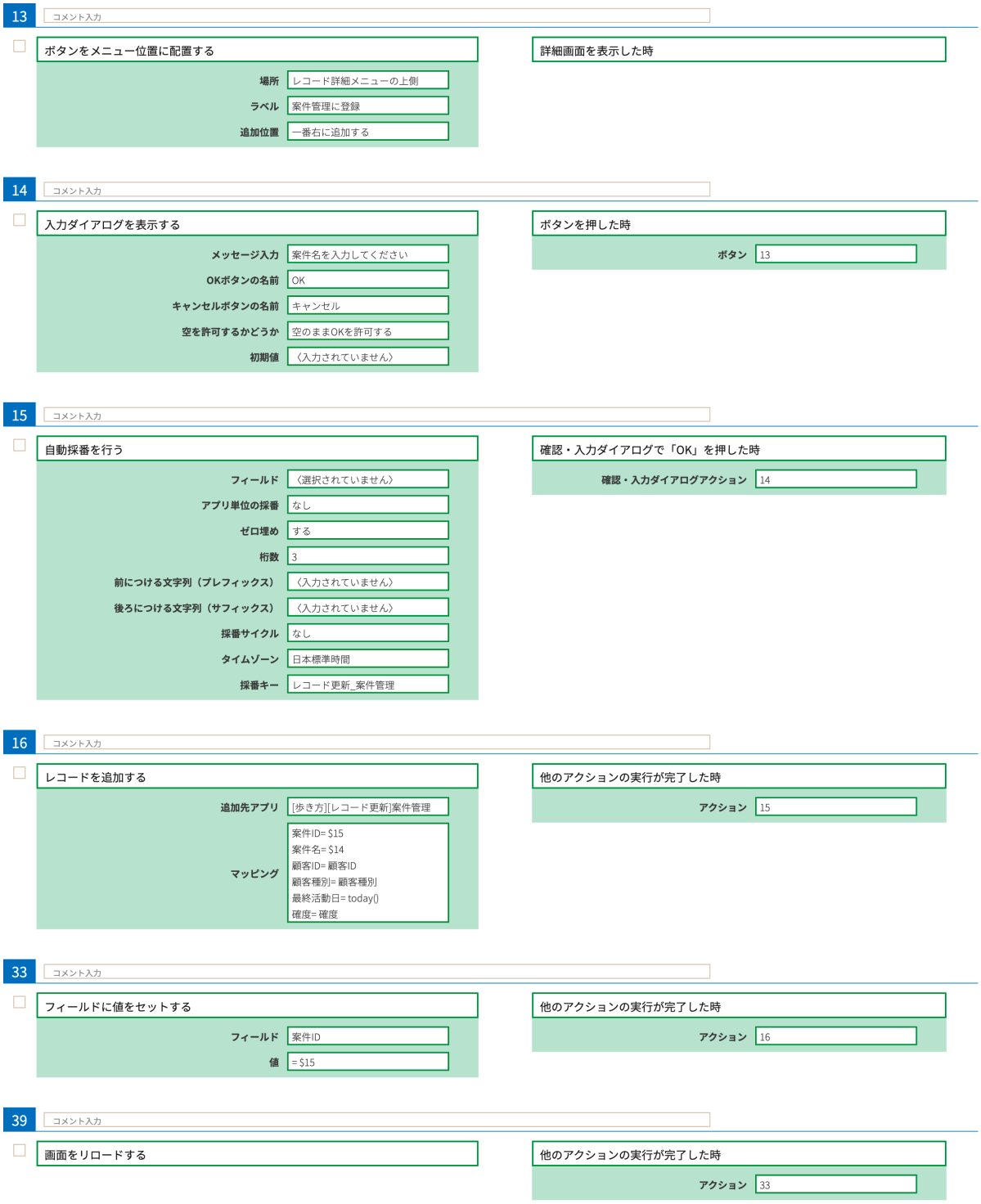
[やること：**レコードを追加する**] では必ず１件のレコードが追加されます。

カスタマイズ例

この例では、『活動履歴』アプリにレコードを追加したときに、一緒に『案件管理』アプリに案件情報を登録します。

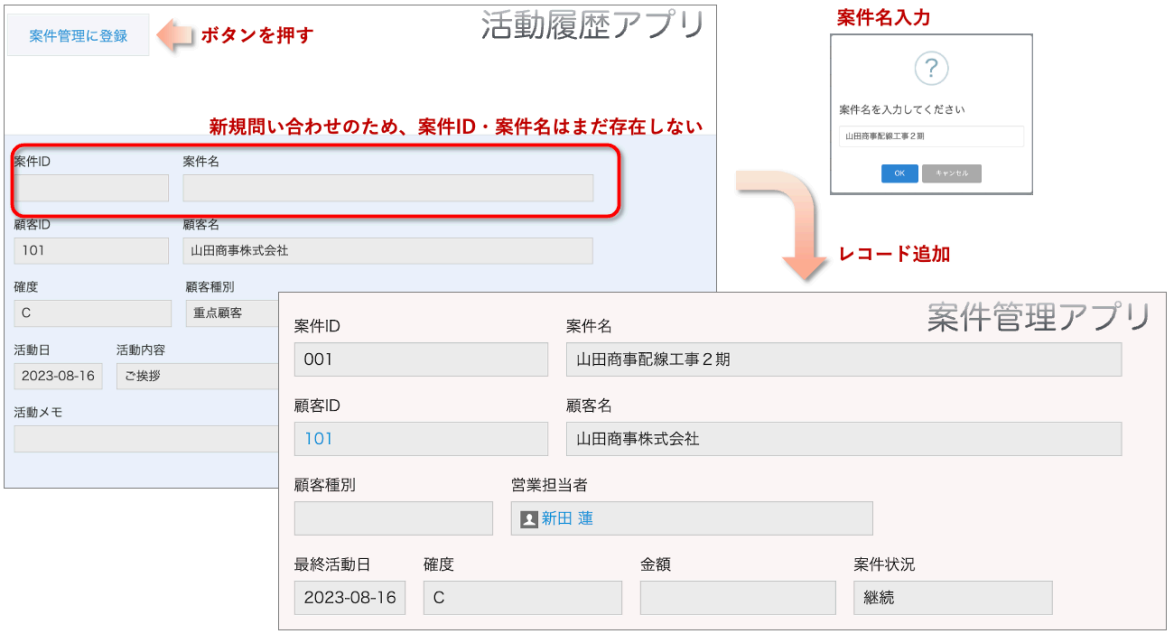
『活動履歴』アプリに「案件登録」ボタンを置いておき、ボタンを押したときに『案件管理』アプリに登録します。

<図：カスタマイズ>


13
コメント入力
ボタンをメニュー位置に配置する
場所
レコード詳細メニューの上側
ラベル
案件管理に登録
追加位置
一番右に追加する
詳細画面を表示した時
14
コメント入力
入力ダイアログを表示する
メッセージ入力
案件名を入力してください
OKボタンの名前
OK
キャンセルボタンの名前
キャンセル
空を許可するかどうか
空のままOKを許可する
初期値
〈入力されていません〉
ボタンを押した時
ボタン
13
15
コメント入力
自動採番を行う
フィールド
〈選択されていません〉
アプリ単位の採番
なし
ゼロ埋め
する
桁数
3
前につける文字列（プレフィックス）
〈入力されていません〉
後ろにつける文字列（サフィックス）
〈入力されていません〉
採番サイクル
なし
タイムゾーン
日本標準時間
採番キー
レコード更新_案件管理
確認・入力ダイアログで「OK」を押した時
確認・入力ダイアログアクション
14
16
コメント入力
レコードを追加する
追加先アプリ
[歩き方][レコード更新]案件管理
マッピング
案件ID= $15
案件名= $14
顧客ID= 顧客ID
顧客種別= 顧客種別
最終活動日= today()
確度= 確度
他のアクションの実行が完了した時
アクション
15
33
コメント入力
フィールドに値をセットする
フィールド
案件ID
値
= $15
他のアクションの実行が完了した時
アクション
16
39
コメント入力
画面をリロードする
他のアクションの実行が完了した時
アクション
33

実行結果

＜図：追加のイメージ＞



解説

『活動履歴』アプリにレコード追加したときに、案件が新しく生まれたということで『案件管理』アプリにもレコードを追加します。

案件名も決めないといけないので、「案件登録」ボタンを押したときに [やること：**入力ダイアログを表示する**] を使って案件名を入力します。

案件 ID は [やること：**自動採番を行う**] を使って自動的に採番しています。

3-2. レコードを更新する（キーの値を直接指定）

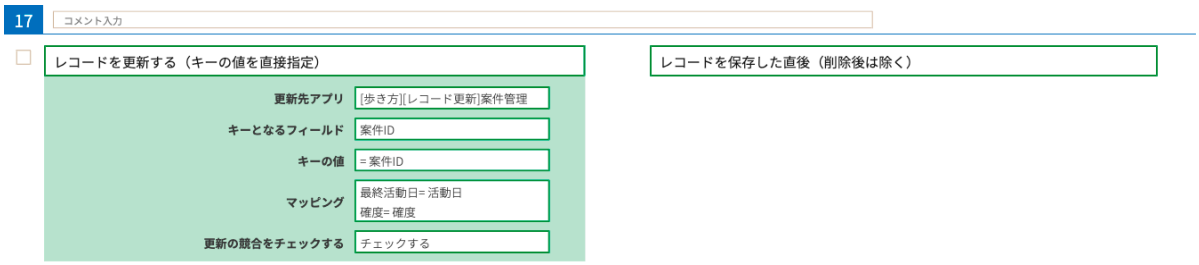
カスタマイズ対象のレコードと紐づくレコードを更新する際に使用するのが、[やること：**レコードを更新する（キーの値を直接指定）**] です。**2-1. レコード追加、レコード更新**で説明したように、レコード更新をする際には「キー」が必要になります。レコード更新でパラメーターに指定した「キー」を持つレコードを探して見つかった場合にレコード更新を行います。指定した「キー」を持つレコードが見つからない場合には、更新処理は行われません。

[やること:**レコードを更新する（キーの値を直接指定）**] では、「キー」の値に、表示されているフィールド値をはじめ、他のアクションの結果や固定値などを指定することができます。
キーを指定してレコード更新をしたい場合は、まず [やること：**レコードを更新する（キーの値を直接指定）**] を使うことを考えてみましょう。

カスタマイズ例

ここでは、『活動履歴』アプリのレコードを保存したときに、「案件 ID」で紐づいている『案件管理』アプリのレコードの「確度」「最終活動日」も更新する例を紹介します。
「確度」「活動日」は活動履歴アプリに登録しているので、その内容で上書き更新します。

＜図：カスタマイズ＞



実行結果

＜図：更新のイメージ＞

解説

今回の例では、『活動履歴』アプリの「案件 ID」フィールドと、『案件管理』アプリの「案件 ID」フィールドをそれぞれのキーとして更新しました。

更新するタイミングは、『活動履歴』アプリのレコード保存後＜条件：**レコードを保存した直後（削除後は除く）**＞としています。こうすると、保存が完了してから更新する設定となります。『活動履歴』アプリのレコード保存をキャンセルした場合は『案件管理』アプリは更新されません。

3-3. レコードを更新または追加する（キーの値を直接指定）

カスタマイズ対象のレコードと紐づくレコードを更新するという点においては、[やること：**レコードを更新する（キーの値を直接指定）**] と同じですが、紐づくレコードが更新先アプリに存在しない際にレコード追加ができるのが、[やること：**レコードを更新または追加する（キーの値を直接指定）**] です。

カスタマイズ例

3-1. レコードを追加する で『案件管理』アプリに『活動履歴』アプリのレコード内容を元に『案件管理』アプリにレコード追加するボタンをカスタマイズで作成し、
3-2. レコードを更新する（キーの値を直接指定） では『活動履歴』アプリのレコード保存後に『案件管理』アプリの「確度」「最終活動日」を更新するカスタマイズを作成しました。

[やること:**レコードを更新または追加する（キーの値を直接指定）**] を使うと、この２つを１つのアクションで実現できます。

＜図：カスタマイズ＞

実行結果

＜図：更新または追加のイメージ＞

解説

[やること：**レコードを更新する（キーの値を直接指定）**] と似ていますが、違うところは、更新の対象となる、キーで紐づくレコードがなかった場合の動きです。

2-1. レコード追加、レコード更新で説明したように、レコード更新をする際には「キー」が必要になります。レコード更新でパラメーターに指定した「キー」を持つ更新先のアプリのレコードを探して見つかった場合にレコード更新を行います。ここまでは [やること：**レコードを更新する（キーの値を直接指定）**] と同じですが、[やること：**レコードを更新または追加する（キーの値を直接指定）**] の場合は、指定した「キー」を持つレコードが見つからなかったときに「レコード追加」をする点が大きな違いです。

今回の例では、**3-2. レコードを更新する（キーの値を直接指定）**と同様に『活動履歴』アプリの「案件 ID」フィールドと、『案件管理』アプリの「案件 ID」フィールドをそれぞれのキーとしています。

『活動履歴』アプリの「案件 ID」フィールドに一致する値が『案件管理』アプリの「案件 ID」フィールドに存在しない場合には、『案件管理』アプリにレコード追加をします。

一方、『活動履歴』アプリの「案件 ID」フィールドの値と、『案件管理』アプ

リの「案件 ID」フィールドが一致するレコードが存在する場合には、そのレコードに対してレコード更新を行います。

3-4. レコードを更新する（キーの値をフィールドで指定）

[やること：**レコードを更新する（キーの値を直接指定）**] と同様にカスタマイズ対象のレコードと紐づくレコードを更新することができ、キーにフィールドを指定するのが [やること：**レコードを更新する（キーの値をフィールドで指定）**] です。

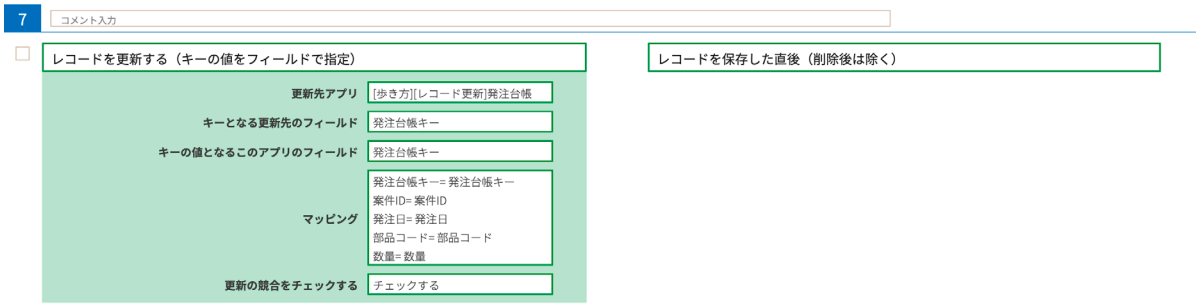
これは [やること：**レコードを更新する（キーの値を直接指定）**] と似ていますが、利用できるシーンに一部違いがあります。具体的には、次の例のようにテーブル内の値をキー項目として設定した場合に、テーブル行の件数分だけ処理を繰り返すことができます。

ただし、[やること：**レコードを更新する（キーの値を直接指定）**] ではキーとして「フィールド」「他のアクションの結果」「関数」などが指定可能なのに対して、[やること：**レコードを更新する（キーの値をフィールドで指定）**] ではキーに指定できるのは「フィールド」のみとなる点に注意が必要です。

カスタマイズ例

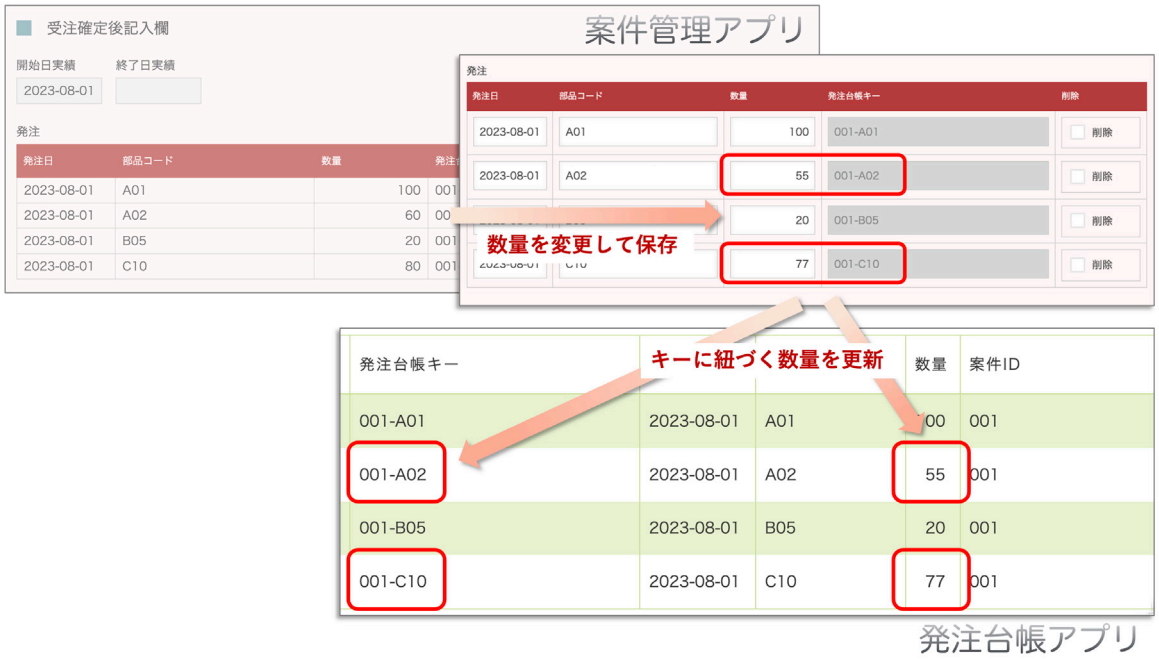
この例では、『案件管理』アプリのテーブルの内容をレコードとして書き出した『発注台帳』アプリを使います。
『案件管理』アプリのテーブルの内容が変わったときに、同時に『発注台帳』アプリの内容も更新します。
レコード更新では「キー」が必要なため「発注台帳キー」フィールドを用意しています。

＜図：カスタマイズ＞



実行結果

＜図：更新のイメージ＞



解説

このように、[やること：**レコードを更新する（キーの値をフィールドで指定）**] の《パラメーター：キーの値となるこのアプリのフィールド》にテーブルの中のフィールドを指定すると、テーブルの件数分だけ処理を繰り返すことができます。これは、[やること：**レコードを更新する（キーの値を直接指定）**] では実現できないことです。

今回の例ではテーブル行が４行ありますが、１つのアクションだけで４行すべてをキーとして指定し、レコード更新をすることが可能です。

今回の例では、テーブル内のフィールド「発注台帳キー」を「キー」としました。

注意点

この [やること：**レコードを更新する（キーの値をフィールドで指定）**] を一覧画面で使用する場合、一覧画面に表示された数だけ処理が繰り返されます。ただし、一覧画面に表示されているレコードだけが対象になりますので注意が必要です。一覧画面の条件に合致していても、表示されていない次ページ以降のレコードは処理対象になりません。
次ページ以降のレコードも含めた処理を行いたい場合は、これ以降の節で紹介する他のやることを使用してください。

3-5. レコードを更新または追加する（キーの値をフィールドで指定）

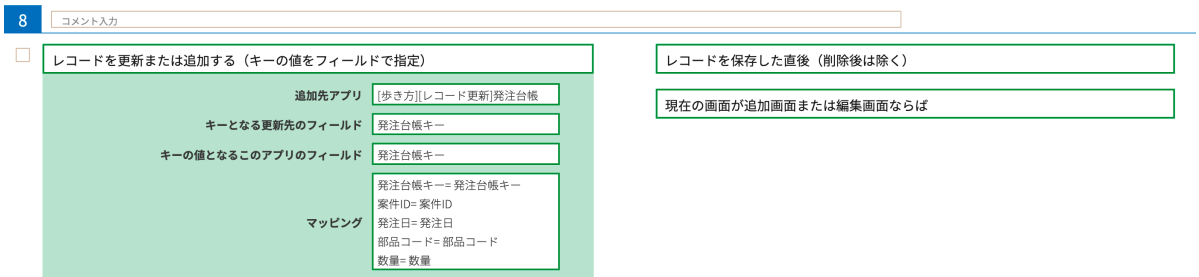
カスタマイズ対象のレコードと紐づくレコードを更新するという点においては、[やること：**レコードを更新する（キーの値を直接指定）**] と同じですが、紐づくレコードが更新先アプリに存在しない際にレコード追加ができるのが、[やること：**レコードを更新または追加する（キーの値をフィールドで指定）**] です。

これは [やること：**レコードを更新または追加する（キーの値を直接指定）**] と似ていますが、利用できるシーンに一部違いがあります。具体的には、次の例のようにテーブル内の値をキー項目として設定した場合に、テーブル行の件数分だけ処理を繰り返すことができます。
ただし、[やること：**レコードを更新または追加する（キーの値を直接指定）**] ではキーとして「フィールド」「他のアクションの結果」「関数」などが指定可能なのに対して、[やること：**レコードを更新または追加する（キーの値をフィールドで指定）**] ではキーに指定できるのは「フィールド」のみとなる点に注意が必要です。

カスタマイズ例

3-4. レコードを更新する（キーと値をフィールドで更新） でテーブルを書き出したレコードに対して、変更内容を更新する処理を作りましたが、この方法ではあとから追加したテーブル行に対しては対応ができません。
[やること：**レコードを更新または追加する（キーの値をフィールドで指定）**] を使ったカスタマイズでは次の例のようになります。

＜図：カスタマイズ＞



実行結果

＜図：更新または追加のイメージ＞

解説

[やること：**レコードを更新する（キーの値をフィールドで指定）**] と似ていますが、違うところは、更新の対象となるレコードがなかった場合の動きです。**2-1. レコード追加、レコード更新** で説明したように、レコード更新をする際には「キー」が必要になります。レコード更新でパラメーターに指定した「キー」を持つ更新先アプリのレコードを探して、レコードが見つかった場合にレコード更新を行います。ここまでは [やること：**レコードを更新する（キーの値をフィールドで指定）**] と同じなのですが、[やること：**レコードを更新または追加する（キーの値をフィールドで指定）**] の場合は、指定した「キー」を持つレコードが見つからなかったときに「レコード追加」をする点が大きな違いです。

今回の例では、すでに登録されていたテーブル行に対しては変更内容をレコード更新、新しく追加されたテーブル行に対してはレコード追加しています。レコードがすでに登録されているかどうかは、「キー」で判断します。

3-6. レコードを書き出す

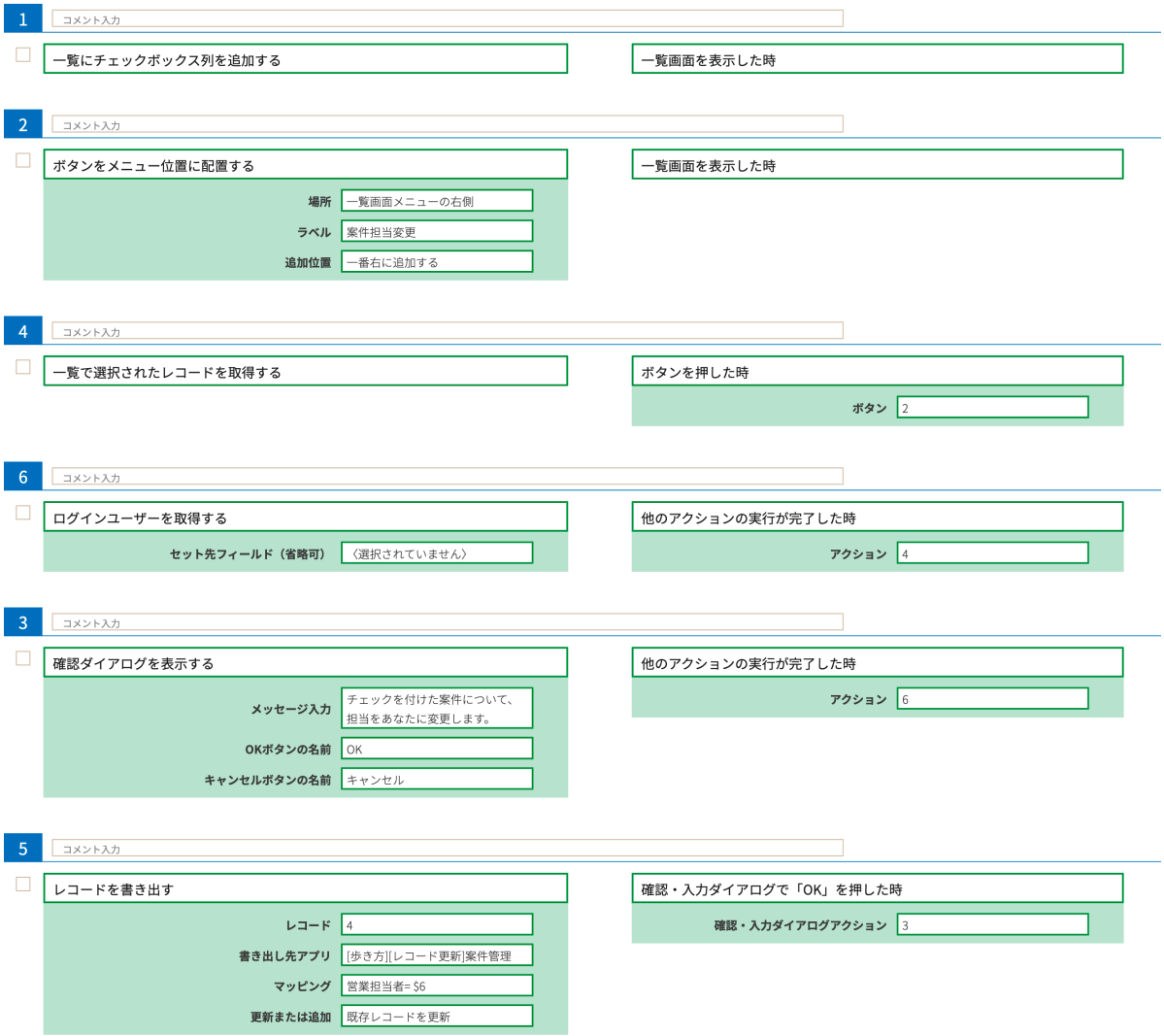
[やること:**レコードを更新する（キーの値を直接指定）**] や [やること:**レコードを更新または追加する（キーの値をフィールドで指定）**] と同様にカスタマイズ対象のレコードと紐づくレコードを更新、または追加することができるのが、[やること：**レコードを書き出す**] です。《パラメーター：レコード》で元になるレコードを指定するため、複数レコードを対象にでき、そのレコードの値を元にレコードを更新することが可能です。

なお、カスタマイズ対象のアプリと同じアプリに対して [やること：**レコードを書き出す**] を使う場合は、レコード更新・レコード追加のどちらも行うことができますが、違うアプリに対して [やること：**レコードを書き出す**] を使う場合は、レコード追加のみ可能です。

カスタマイズ例

この例では、『案件管理』アプリの一覧画面でチェックをつけた案件に対して、営業担当者をログインユーザーに一括更新します。

<図：カスタマイズ>



実行結果

<図：更新のイメージ>

解説

レコード更新には「キー」が必要なことは **2-1. レコード追加、レコード更新** で説明していますが、今回の [やること：**レコードを書き出す**] では「キー」は指定しません。これはなぜかというと、[やること：**レコードを書き出す**] でレコード更新をするときはレコード番号を「キー」にして更新が行われているからです。このため、パラメーターの「既存レコードを更新」は更新先アプリが元になるレコードのアプリと同じ場合にのみ指定が可能です。

今回の例では、チェックを付けたレコードに対して更新をしました。このように元になるレコードに対して更新したり、元になるレコード全件を対象にレコード追加をする場合には、[やること：**レコードを書き出す**] を使うと処理速度も速く、API リクエスト数も少なく済むのでおすすめです。

今回の例では、[やること：**レコードを書き出す**] を使ってレコードの更新を行いましたが、パラメーターに「新規にレコードを追加」を選択することでレコード追加を行うこともできます。なお、「新規にレコードを追加」の場合は、レコードを追加する先のアプリが元になるレコードと同じアプリでも違うアプリでも実行する事ができます。

3-7. レコードを 1 つずつ書き出す

[やること：**レコードを書き出す**] と同様に複数レコードを対象にレコードを追加・更新することが可能なのが [やること：**レコードを 1 つずつ書き出す**] です。

[やること：**レコードを書き出す**] との違いは、 1 レコードずつ書き出し処理を行うため、kintone の Webhook 通知の送信が可能な点です。それ以外の機能は [やること：**レコードを書き出す**] と同じですので、以下の利用シーン以外の場合は [やること：**レコードを 1 つずつ書き出す**] ではなく [やること：**レコードを書き出す**] を選択するようにしましょう。

利用シーン

kintone の Webhook 通知の送信ができると、どういった事が実現できるのでしょうか？

たとえば、次のような利用シーンが考えられます。

・kintone のレコード追加をきっかけとして連携サービスを利用したいとき。
・kintone の Webhook 通知の送信を利用して、他システムにデータ連携したいとき。
・カスタマインの kintone アプリの Webhook のカスタマイズを動かしたいとき。

ほかにも、kintone の Webhook 通知の送信を利用した処理を動かしたいときが、利用シーンとして考えられます。

カスタマイズ例

3-6. レコードを書き出す の説明で、チェックを付けたレコードに対して、担当者を一括変更する例を紹介しました。この方法だとレコードの一括更新がスムーズに行えるのですが、Webhook の通知が送信されません。

こういった時に [やること：**レコードを 1 つずつ書き出す**] を利用すると、一度にまとめて更新しつつ、1 件毎に Webhook 通知を送信することができます。

＜図：カスタマイズ＞

実行結果

＜図：更新のイメージ＞

このカスタマイズを動かした kintone レコードの更新結果としては [やること：**レコードを書き出す**] と同じになります。
違いは Webhook 通知の送信が起きる点です。

解説

[やること：**レコードを書き出す**] に似ていますが、[やること：**レコードを書き出す**] はレコードをまとめて処理するのに対し、[やること：**レコードを 1 つずつ書き出す**] では 1 レコードずつ処理する点が違います。そのため、扱うレコード件数が多い場合は kintone API リクエスト数が多くなりやすく、処理時間も長くなりやすいです。
原則として、[やること：**レコードを書き出す**] を使用するようにし、Webhook 通知の送信を作動させたい場合にのみ [やること：**レコードを 1 つずつ書き出す**] を選ぶようにしましょう。

注意点

kintone の仕様により Webhook の通知は、ドメイン単位で 1 分間に 60 回まで送信されます。1 分間に 60 回を超えてレコードを操作すると、61 回目以降の操作では Webhook の通知が送信されません。
この制限はアプリ単位ではなくドメイン単位での制限であることは意識しておきましょう。

3-8. レコードをもとに別のレコードを更新する

複数のレコードをもとにレコードを更新することに関しては [やること：**レコードを書き出す**] と同じですが、更新先アプリがもとになるレコードのアプリと別のアプリであっても更新が可能な点が違います。

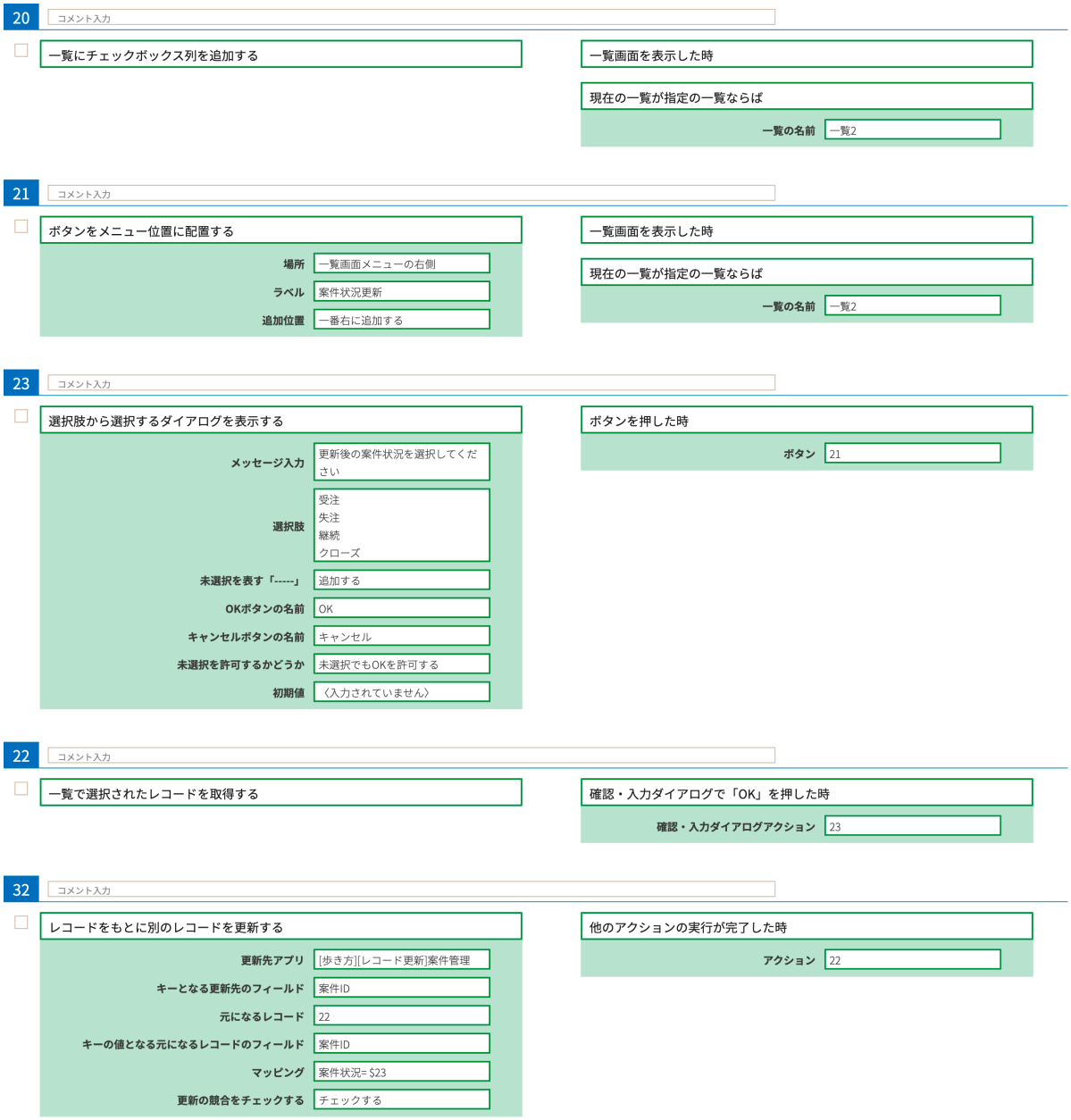
[やること:**レコードをもとに別のレコードを更新する**] は、[やること:**レコードを書き出す**] に比べると処理時間が長くなりやすく、API リクエスト数も多くなります。同じアプリに対する更新の場合は、[やること：**レコードを書き出す**] を利用することをおすすめします。

一方、違うアプリに対して更新する場合には [やること:**レコードを書き出す**] では実現できませんので、[やること：**レコードをもとに別のレコードを更新する**] を利用してください。

カスタマイズ例

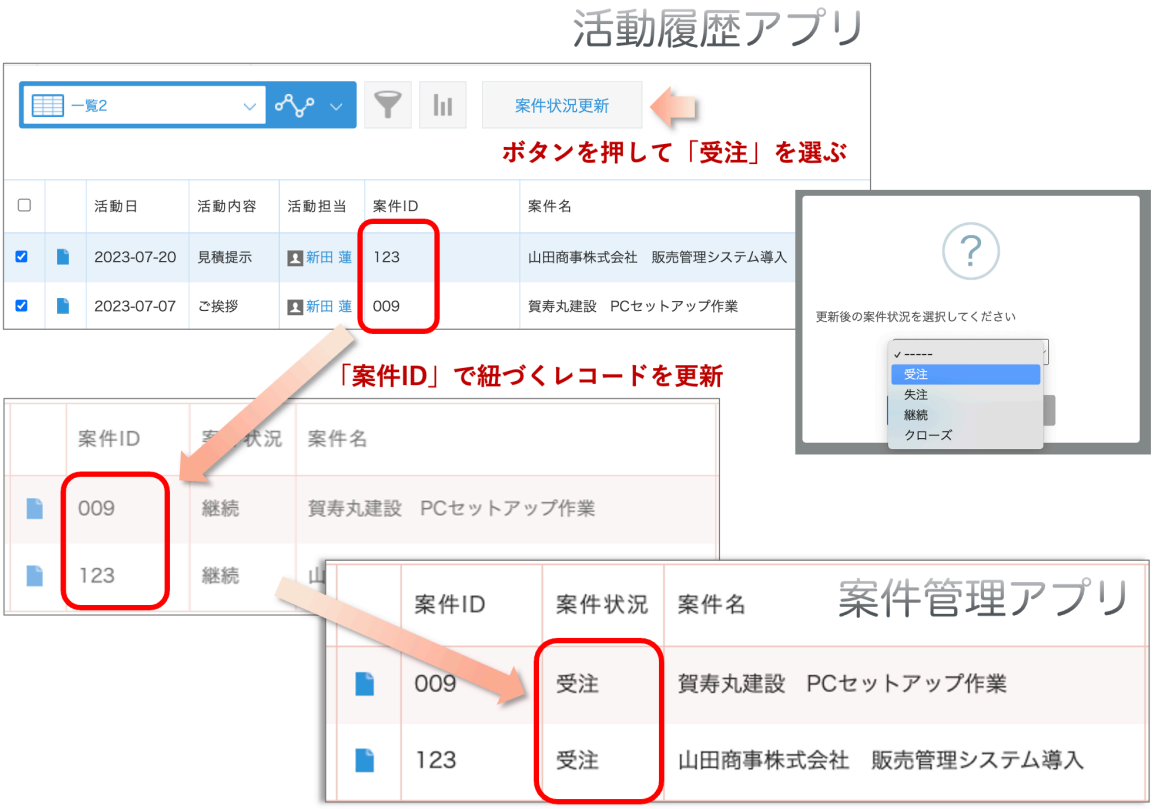
この例では、『活動履歴』アプリの一覧画面で選択したレコードを元に、『案件管理』アプリの「案件状況」フィールドの内容を更新するカスタマイズを作りました。

<図：カスタマイズ>


20
コメント入力
一覧にチェックボックス列を追加する
一覧画面を表示した時
現在の一覧が指定の一覧ならば
一覧の名前
一覧2
21
コメント入力
ボタンをメニュー位置に配置する
場所
一覧画面メニューの右側
ラベル
案件状況更新
追加位置
一番右に追加する
一覧画面を表示した時
現在の一覧が指定の一覧ならば
一覧の名前
一覧2
23
コメント入力
選択肢から選択するダイアログを表示する
メッセージ入力
更新後の案件状況を選択してください
選択肢
受注
失注
継続
クローズ
未選択を表す「-----」
追加する
OKボタンの名前
OK
キャンセルボタンの名前
キャンセル
未選択を許可するかどうか
未選択でもOKを許可する
初期値
〈入力されていません〉
ボタンを押した時
ボタン
21
22
コメント入力
一覧で選択されたレコードを取得する
確認・入力ダイアログで「OK」を押した時
確認・入力ダイアログアクション
23
32
コメント入力
レコードをもとに別のレコードを更新する
更新先アプリ
[歩き方][レコード更新]案件管理
キーとなる更新先のフィールド
案件ID
元になるレコード
22
キーの値となる元になるレコードのフィールド
案件ID
マッピング
案件状況= $23
更新の競合をチェックする
チェックする
他のアクションの実行が完了した時
アクション
22

実行結果

＜図：更新のイメージ＞



解説

この例では『活動履歴』アプリの一覧画面で選択したレコードを元に、『案件管理』アプリを更新しました。

今回の例では「キーとなる更新先のフィールド」には「案件 ID」フィールドを指定しています。一覧で選択した複数レコードの「案件 ID」が同じレコードに対して、一括で『案件管理』アプリの「案件状況」フィールドの内容を更新します。

「案件状況」の値は、[やること：**選択肢から選択するダイアログを表示する**] を使って、プルダウン形式で選択しています。

[やること：**レコードを書き出す**] の場合は、元になるレコードと違うアプリに対してレコード更新することができませんでしたが、[やること：**レコードをもとに別のレコードを更新する**] を使うと、それが可能になります。

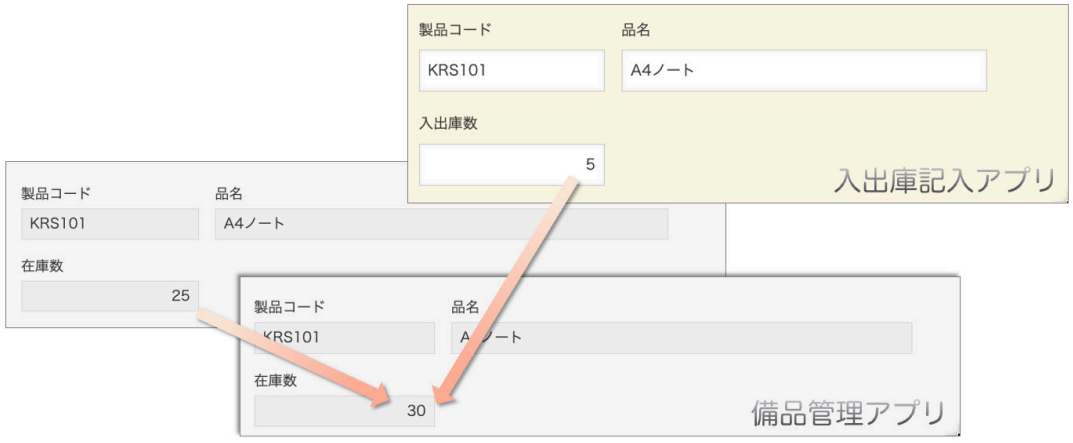
元になるレコードを指定するために、[やること：**レコードをもとに別のレコードを更新する**] のアクションの前にレコードを取得する [やること]（[やるこ

と：**一覧の条件でレコードを全件取得する**] など）を使ったアクションを実行しておく必要があります。

コラム：1 件ずつ処理することが適しているケース

[やること：**レコードをもとに別のレコードを更新する**] のアクションでは、 1 件ずつ更新先のレコードを取得したあとにレコード更新を行っており、処理時間と API リクエスト数が多くなりやすいです。
一方でメリットもあり、 1 件ずつ更新先のレコードを取得することで、下図のようにマッピングで更新先のレコードの値を使うことができます。

たとえば、在庫数の管理をしている『備品管理アプリ』と入出庫数を記入する『入出庫記入アプリ』を例に説明します。以下の例だと、すでに『備品管理アプリ』の A4 ノートは 25 個の在庫があります。そのため、入庫があった際には入庫数の 5 を『備品管理アプリ』に登録するのではなく、すでに在庫数登録している 25 個に入庫数の 5 個を足す必要があります。

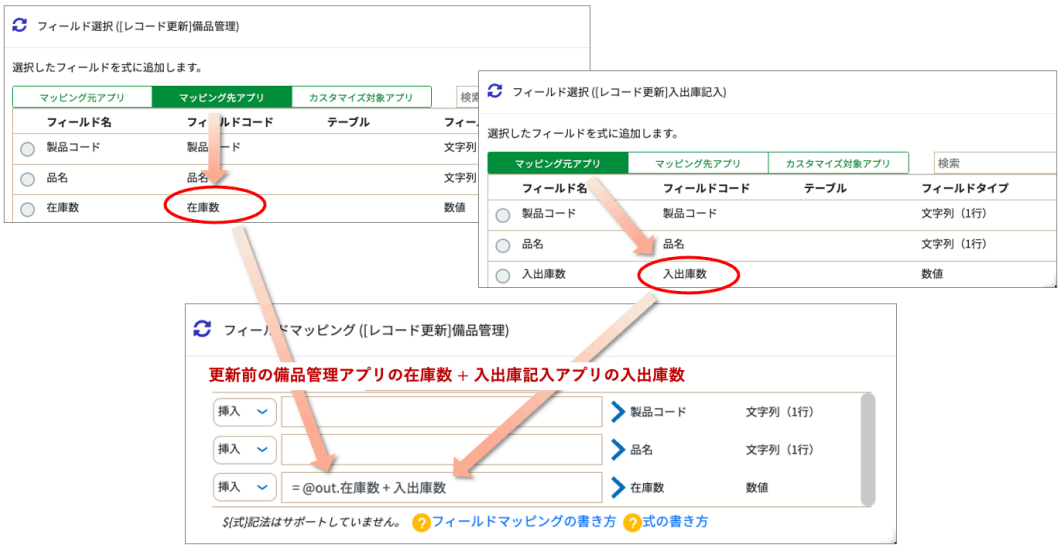


更新前の備品管理アプリの在庫数 + 入出庫記入アプリの入出庫数

[やること：**レコードをもとに別のレコードを更新する**] が実行されると、「製品コード」のフィールド値が同じレコードを『備品管理アプリ』から 1 つ取得します。取得したレコードの値はマッピングで使うことができるため「更新後の在庫数 ＝ 更新前の在庫数 ＋ 入出庫数」のように、

計算結果をレコードに反映するような処理が実現できます。

＜図：在庫数計算の例＞



上図のように「マッピング先アプリ」からフィールドを選ぶと「@out.」がつき、更新先アプリのフィールド値を使用できます。また、下図のように「カスタマイズ対象アプリ」からフィールドを選ぶと「@this.」がフィールドコードの前に付き、カスタマイズ対象アプリのフィールド値を使用できます。

＜図：マッピング＞

3-9. レコードをもとに別のレコードを更新または追加する

複数のレコードをもとにレコードを更新する点に関しては、[やること:**レコードをもとに別のレコードを更新する**] と同じですが、紐づくレコードが更新先アプリに存在しない際にレコード追加ができるのが [やること：**レコードをもとに別のレコードを更新または追加する**] です。

カスタマイズ例

3-8. レコードをもとに別のレコードを更新する で『活動履歴』アプリの一覧画面で選択したレコードを元に、『案件管理』アプリの「案件状況」フィールドの内容を更新するカスタマイズを作りましたが、これでは『案件管理』アプリに登録がない案件は更新されません。
これに対し、[やること:**レコードをもとに別のレコードを更新または追加する**] を使ったカスタマイズだと、『案件管理』アプリに登録がある案件についてはレコード更新、『案件管理』アプリに登録がない案件についてはレコード追加を行います。

<図：カスタマイズ>

34 コメント入力

一覧にチェックボックス列を追加する

一覧画面を表示した時

35 コメント入力

ボタンをメニュー位置に配置する

場所	一覧画面メニューの右側
ラベル	案件状況　更新または追加
追加位置	一番右に追加する

一覧画面を表示した時

36 コメント入力

選択肢から選択するダイアログを表示する

メッセージ入力	更新後の案件状況を選択してください
選択肢	受注 失注 継続 クローズ
未選択を表す「-----」	追加する
OKボタンの名前	OK
キャンセルボタンの名前	キャンセル
未選択を許可するかどうか	未選択でもOKを許可する
初期値	〈入力されていません〉

ボタンを押した時

ボタン	35

37 コメント入力

一覧で選択されたレコードを取得する

確認・入力ダイアログで「OK」を押した時

確認・入力ダイアログアクション	36

38 コメント入力

レコードをもとに別のレコードを更新または追加する

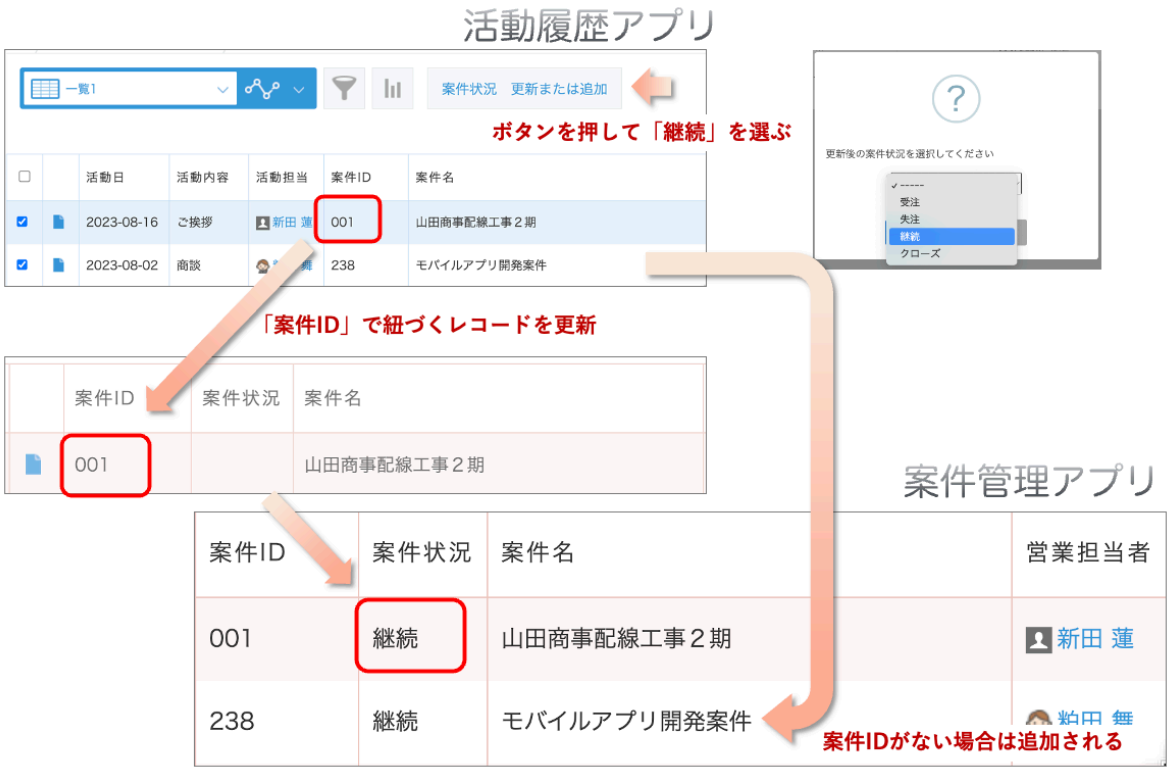
追加先アプリ	[歩き方][レコード更新]案件管理
キーとなる更新先のフィールド	案件ID
元になるレコード	37
キーの値となる元になるレコードのフィールド	案件ID
マッピング	顧客ID= 顧客ID 顧客名= 顧客名 営業担当者= 活動担当 最終活動日= 活動日 確度= 確度 案件状況= $36

他のアクションの実行が完了した時

アクション	37

実行結果

＜図：更新・追加のイメージ＞



解説

[やること：**レコードをもとに別のレコードを更新する**] と似ていますが、違うところは、更新の対象となるレコードがなかった場合の動きです。

この例では、**3-8. レコードをもとに別のレコードを更新する** と同様に『活動履歴』アプリの一覧画面で選択したレコードを元に、『案件管理』アプリを更新しました。さらに、《パラメーター：キーの値となる元になるレコードのフィールド》である「案件 ID」に一致する案件 ID が『案件管理』アプリに存在しない場合には、レコード追加が行われます。

[やること：**レコードをもとに別のレコードを更新する**] と同じく、元になるレコードを指定するために、[やること：**レコードをもとに別のレコードを更新または追加する**] のアクションの前にレコードを取得する「やること」（**一覧の条件でレコードを全件取得するなど**）を使ったアクションを実行しておく必要があります。

[やること：**レコードをもとに別のレコードを更新または追加する**] は、[やること：**レコードをもとに別のレコードを更新する**] と同様に、他の更新・追加系のやることに比べると処理時間が長くなりやすく、使用する API リクエスト数も多くなります。

注意点

更新元レコードのキーの値《パラメーター：キーの値となる元になるレコードのフィールド》が空のレコードが存在する場合には注意が必要です。
キーの値が空のレコードが複数存在する場合すべてキーが同一であるとみなされ、同じレコードを繰り返し更新し、最後に更新された値のみが有効となります。

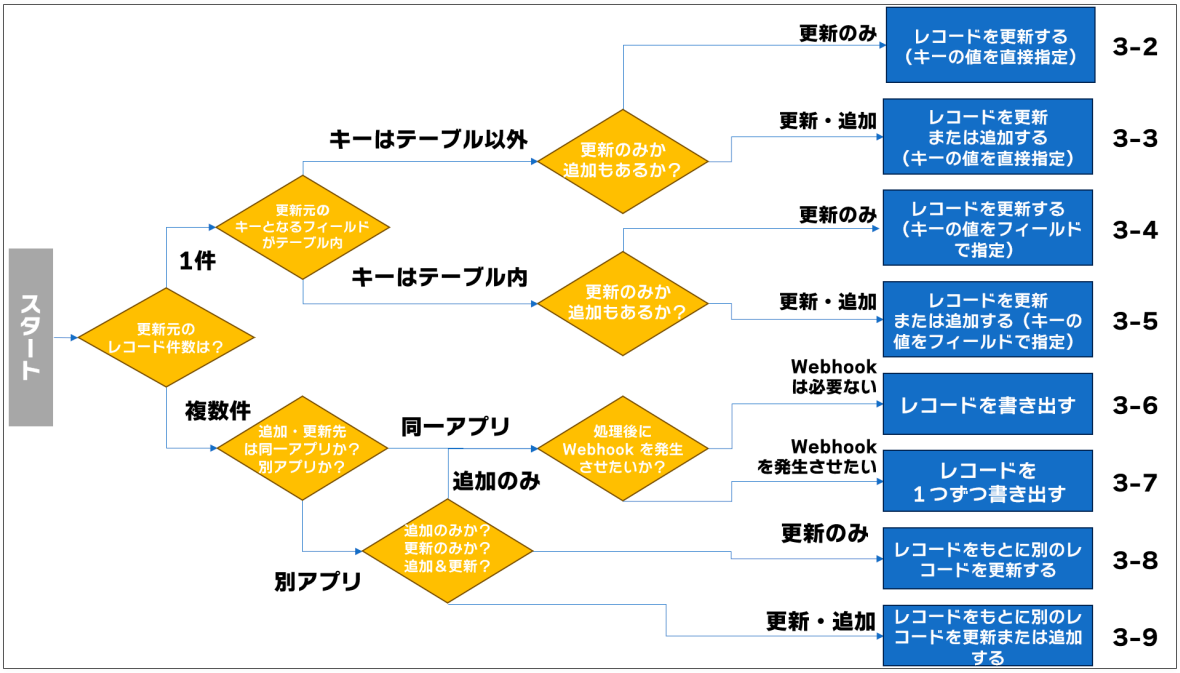
具体的には以下のような動きとなります。

上記のような動作を防ぐためには、原則としてキーとなるフィールドは重複禁止、かつ必須に設定することが望ましいです。

4.「やること」使い分けのポイント

本冊子ではレコード更新に関する多くの「やること」を紹介しました。さまざまな「やること」の利用例や注意点を読んでも、「結局どれを使ったら良いの？」と悩まれることもあるかと思います。ここでは、そんな方のためにフローチャートを用意しました。

初めての方にも理解していただきやすいよう、詳細な説明は省いております。要件によっては一部合致しないケースもありますが、そこはご理解のうえ簡易的なチャートとしてご利用ください。「レコードを追加する」は用途がシンプルで悩まれるケースがほとんど無いと考えられるため、以下の図からは省略しています。



また、各「やること」ごとに実現できること・実現できないことを一覧としてまとめました。上記のフローチャートと合わせてご活用ください

やること名	対象レコード件数		更新元と同じアプリへの処理		更新元と別のアプリへの処理		本書での紹介
	更新元	更新先	追加	更新	追加	更新	
レコードを更新する（キーの値を直接指定）	1	1 or 多	X	○	X	○	3-2
レコードを更新または追加する（キーの値を直接指定）	1	1 or 多	○	○	○	○	3-3
レコードを更新する（キーの値をフィールドで指定)	1	1 or 多	X	○	X	○	3-4
レコードを更新または追加する（キーの値をフィールドで指定）	1	1 or 多	○	○	○	○	3-5
レコードを書き出す	多	多	○	○	○	X	3-6
レコードを１つずつ書き出す	多	多	○	○	○	X	3-7
レコードをもとに別のレコードを更新する	多	多	X	○	X	○	3-8
レコードをもとに別のレコードを更新または追加する	多	多	○	○	○	○	3-9

5. さらにアプリを便利に活用するために

ここまでの例を参考にカスタマイズをして運用してみると、当初は想定していなかったパターンに気がつくかもしれません。
また、本冊子は「レコード更新」の説明に重点を置いているため、実運用で利用するために必要となる他の設定は省いている部分があります。
以下にいくつかの例を紹介します。

キーが空欄のままレコード追加しないための対策

3-3. レコードを更新または追加する（キーの値を直接指定）の例では、『活動履歴』アプリの「案件 ID」フィールドが空欄の場合に『案件管理』アプリにレコード追加をすると、「案件 ID」フィールドが空欄の状態のレコードとなってしまいます。「案件 ID」フィールドはキーとして使いたいので、空欄ではなく、重複しない値を入れておきたいです。
こんな時には、レコード追加する際に [やること：**自動採番を行う**] で採番を行うのも 1 つの方法です。
また、＜条件：ボタンを押した時＞に「案件 ID」フィールドが空欄かどうかで、空欄であれば **3-1. レコードを追加する**、空欄でなければ **3-2. レコードを更新する（キーの値を直接指定）** に分岐するという方法もあります。

テーブル行を削除したいときの対策

3-4. レコードを更新する（キーの値をフィールドで指定）、**3-5. レコードを更新または追加する（キーの値をフィールドで指定）**の例では、テーブル行の内容を別アプリのレコードに追加や更新をしました。ただ、テーブル行は行追加だけではなく、「-」ボタンを押すことで削除することも可能です。
『案件管理』アプリのテーブル行を削除した際、連携している『発注台帳』アプリのレコードも削除したいところなのですが、カスタマインにはレコードを削除する [やること] はありません。そのため、[やること：**テーブルの行増減ボタンを非表示にする**] を使ってテーブル行の削除ボタンは無効とし、その代わりに該当のテーブル行が無効であることを示す「削除チェックボックス」を使うことをおすすめします。
詳しくは、サポートサイト内「**テーブルを別アプリのレコードに書き出し、常に同期を取る方法**」をご覧ください。

ここに紹介しているのは一部の例です。運用していて、作ったカスタマイズでは足りないパターンを発見したら、どのようなカスタマイズを作れば対応ができるのか、考えてみましょう。
いろいろな方法があると思いますので、動作を試しながら、運用に合ったカスタマイズを検討してみてください。

6. うまく動かないときは?

この章では、うまく動かない時に原因を見つけるための方法と、便利なカスタマインの機能について紹介します。

6-1. カスタマイズは少し作っては確認、少し作っては確認の繰り返しで育てよう

カスタマインのカスタマイズは複数のアクションを組み合わせて作成することが多く、慣れてくると多くのアクションによって構成される大きなカスタマイズを、一度も kintone アプリで試して動かさずに作ってしまいがちです。
このようにして一度も実行結果を試さずに作成した大きなカスタマイズは、うまく動かなかった場合にどのアクションに原因があるかを見つけるのが非常に大変なことになってしまいます。

複数のアクションを組み合わせたカスタマイズでは、少し作ったら実行して動きを確認する→問題があれば直す→動きを確認する→問題なければ次のカスタマイズを追加する、という手順の繰り返しで作成することをおすすめします。

6-2. うまく動かない時に最初に確認するポイント

カスタマイズを「kintone アプリへ登録」して実行してみると、エラーが出たり、想定とは異なる結果になったりと、うまく動かないことがよくあります。
うまく動かないときは、ひとつひとつ原因となりそうな箇所を確認し、どこに原因があるのかを絞り込んでいくことになります。
カスタマインのほかに JavaScript やプラグインが入っている時
JavaScript やプラグインは、カスタマインのカスタマイズの動作に影響を与える場合があります。
そのためカスタマイズがうまく動かないときは、まずは JavaScript やプラグインを削除・無効化し、kintone アプリにカスタマインのカスタマイズのみが設定されている状態にして動作を確認してみてください。これは、原因がカスタマインのカスタマイズにあるのか、それとも JavaScript やプラグインが影響して動かないのかを判別し、問題を切り分けるために行います。

kintone アプリにカスタマインのカスタマイズのみが登録されている状態で問題なく動作するのであれば、JavaScript やプラグインが影響してカスタマイズが動いていない可能性が高いです。
逆に、JavaScript やプラグインを削除・無効化してもカスタマイズがうまく動かない時はカスタマインのカスタマイズが誤っている可能性が高いです。カスタマイズを見直してみてください。

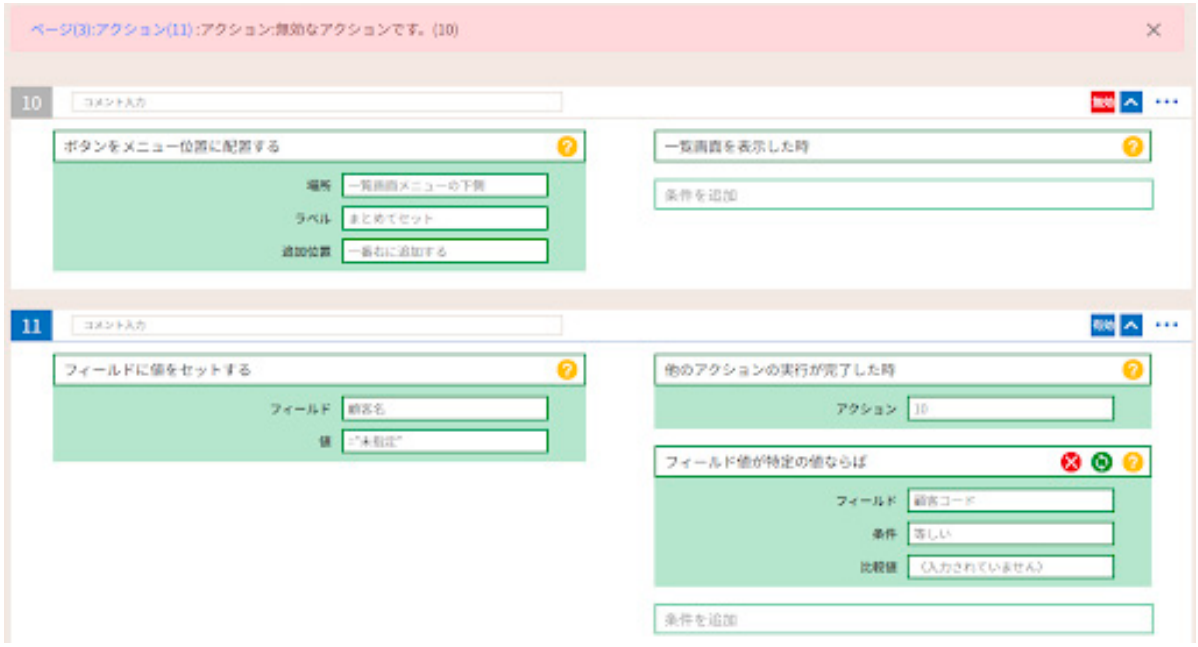
※ドキュメントの「**外部サービス連携**」で連携している以外のプラグイン、及び自作 JavaScript ファイルが含まれた状態では、カスタマインの動作は保証外となります。
外部サービス連携：
https://docs-customine.gusuku.io/ja/actions/external/
（ドキュメントで「外部サービス連携」を御覧ください）
※ JavaScript は後でもとに戻せるように、削除する前にダウンロードするなどして保存しておいてください。

6-3. エラーが表示されたとき

エラーが発生した際に表示されるメッセージには、原因を究明するための情報が含まれています。エラーダイアログが表示されると驚いてしまう事もあると思いますが、エラーの原因がそのまま記載されている事もありますので、一度心を落ち着けて読んでみてください。

カスタマインのカスタマイズ画面で表示されるエラー

カスタマインのカスタマイズ画面で表示されるエラーです。
下記の例では「ページ (3): アクション (11) : アクション : 無効なアクションです。(10)」とあります。
冒頭の「ページ (3): アクション (11) :」の部分がエラーとなっているアクション、その後がエラーの内容を示しています。
今回はアクション番号 11 で＜条件：他のアクションの実行が完了した時＞に指定されたアクションが使用できない、というエラーです。
カスタマイズを見てみますと、アクション番号 10 が無効化されているため、使用できない状態になっていることがわかります。



カスタマイズ画面に表示されるエラーが解消するまでは「kintone アプリへ登録」を行うことができません。メッセージの内容を確認して、アクションの設定を確認・修正してみてください。

kintone アプリで表示されるエラーダイアログ

このような白いダイアログは、カスタマインによって表示されるダイアログです。こういったダイアログ表示は、カスタマインのカスタマイズでやることや条件の使い方が誤っている場合に表示されるエラーメッセージです。

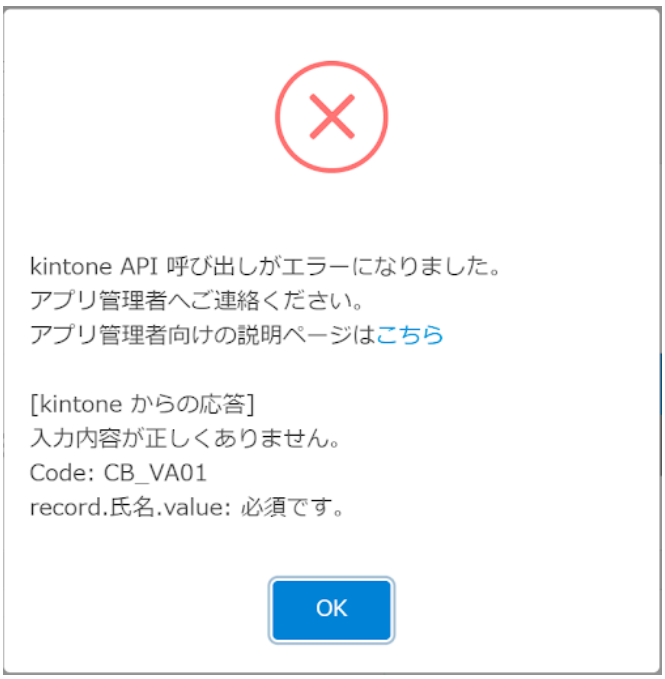
このエラーは、[やること:**ルックアップを取得しなおす**] を、レコード保存前・保存後のタイミングで実行した時に表示されるエラーです。
エラーメッセージの前にある（25）のように、（　）で囲まれた数字が記載されている場合は、このアクション番号のアクションでエラーが起きている事を意味しています。今回の例では（25）ですので、【アクション:25】のアクションでエラーが起きていることがわかります。

このようなエラーが表示されたら、**ドキュメント**の「制限事項」の項目を確認してみてください。
[やること：**ルックアップを取得しなおす**] の制限事項は下記のように記載されています。

> ## 制限事項
>
> - レコード編集画面、レコード追加画面のみで動作します。それ以外の画面で使用するとエラーが表示されます。
> - レコードを保存するタイミングでは動作しません。レコードを保存するタイミングで「ルックアップを取得しなおす」した場合の動作は以下のようになります。
>   - バージョン 1.131(2021年5月13日リリース)以前の場合、エラーは起こりませんが、ルックアップの取得は行われません。
>   - バージョン 1.132(2021年5月20日リリース)以降の場合、エラーが発生します。保存処理は停止しません。
> - ルックアップの現在値から、ルックアップの検索結果が 1 件に決まらない場合は、ルックアップの取得は行われません。

今回のエラーは、[やること：**ルックアップを取得しなおす**] が、レコードを保存するタイミングの＜条件：**レコードを保存する直前（削除時は除く**）＞では使用できないため起こりました。使用するやることを変更するか、条件を変える必要があります。

次のエラーは先ほどのエラーダイアログと同様の白いダイアログですが、こちらのメッセージはカスタマインでカスタマイズを実行した結果、kintone からエラーが返ってきた場合のダイアログです。



見分け方は [kintone からの応答] というメッセージが記載されているかどうかです。[kintone からの応答] とあれば、kintone からエラーが返ってきた場合のダイアログであることがわかります。
この場合は、kintone 側からエラーが返ってきているため、kintone のプロセス管理の設定とカスタマイズで指定したステータスの整合性が取れていない、カスタマイズを実行したユーザーが kintone のアプリやレコードに対して権限がないなど、kintone の設定とカスタマイズに齟齬がないか確認する必要があります。

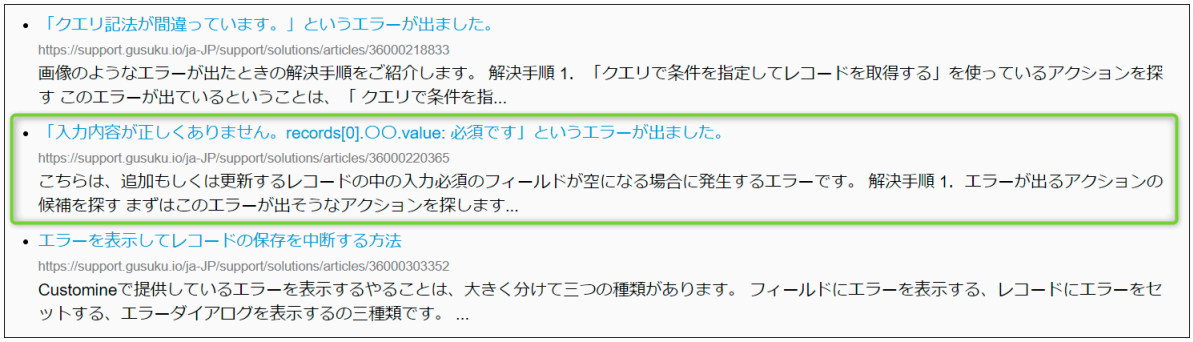
このエラーの場合、[kintone からの応答] に、「入力内容が正しくありません。Code:CB_VA01 record. 氏名 .value: 必須です。」とありますので、レコードを追加・更新する際に kintone アプリで必須と指定されているフィールドに値をセットしていないためエラーが発生していることがわかります。

このように、エラーメッセージを読むとエラーの原因が判明する、または解決の糸口が見つかることはよくあります。一度心を落ち着けて、エラーメッセージを読んでみてください。
また、**サポートサイト**では、カスタマインでカスタマイズしているとよく出会うエラーメッセージとその原因を記事としてまとめています。

サポートサイト
（カスタマインの設定画面右上「サポートサイト」からも開くことができます）
https://support.gusuku.io/

サポートサイト上部の検索窓で、エラーと検索してみてください。

検索すると、エラーに関する記事が表示されます。例えば、先ほどご紹介した必須エラーの例は下記の**「入力内容が正しくありません。records[0].〇〇 .value: 必須です」というエラーが出ました。**に詳しく記載していますので、ぜひ確認してみてください。



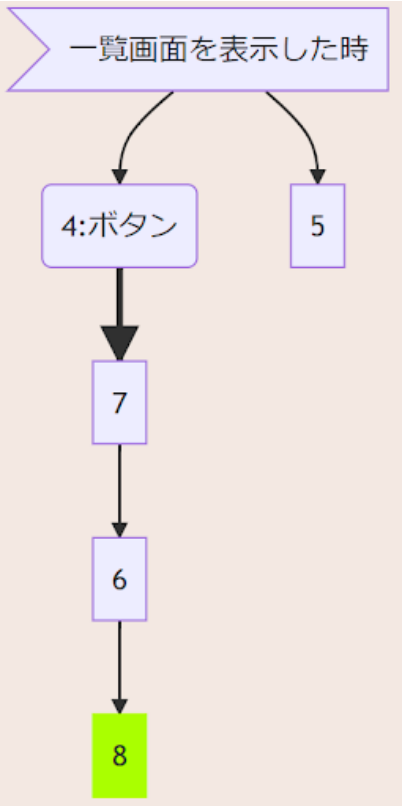
ある程度の長い文で検索して想定する記事がヒットしない場合は、検索キーワードの単語をスペースで区切って指定いただくのがおすすめです。
例えば今回の必須エラーであれば、「必須　エラー」のように区切って検索してみてください。自然な文章をキーワードとして検索するより目当ての記事が検索されやすいことが多いです。

6-4. 原因究明に役立つカスタマインの便利な機能

カスタマインは、うまく動かないときの原因究明に役立つ機能を提供しています。

アクショングラフ

アクショングラフは、＜条件：**他のアクションの実行が完了した時**＞ などで順番を指定したアクションが、どの順番で実行されるのかを図で表示してくれる機能です。
図でアクションの関連性を見て確認できるので、アクションが想定通りにつながっているかを確認したい場合に便利です。
カスタマイズを実行した時、順番に実行されるアクションが途中で動かない場合は、アクショングラフを確認してみてください。矢印が途切れていたり、順番が間違っているなど、一目でわかるためとても便利です。



アクショングラフ上に表示されている番号は、アクション番号です。なおアクショングラフ上のアクション番号をクリックすると、該当するアクションの設定にジャンプすることができます。

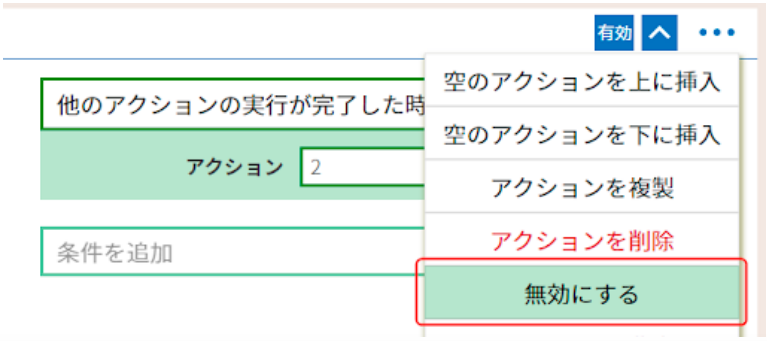
アクショングラフは、ページメニューの「アクショングラフ」を選択または「アクショングラフボタン」をクリックすると表示されます。

アクションの無効化

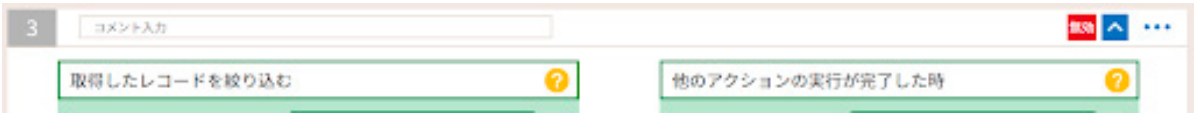
一時的にアクションが実行されないようにしたいときに使用します。
アクションを無効化した後で「kintone アプリへ登録」を行った際に、無効化したアクションは kintone アプリに登録されず、実行されなくなります。

アクションの左上のアクション番号または右上の有効ボタンをクリックすると、そのアクションを無効化できます。

右上のメニューから 無効にする を選択する方法もあります。

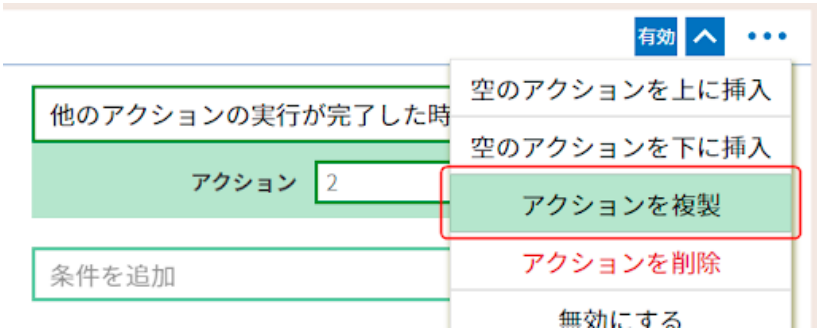


無効化したアクションは、左上のアクション番号がグレーになり、右上には無効と表示されます。



他のアクションの動きを確認するために無効化したり、アクションの設定を変更して動きを確認する際に、変更前の設定を残しておきたい場合にも使用できます。
その際は、アクションを複製して、複製元のアクションを無効化して残しておく、という方法がおすすめです。

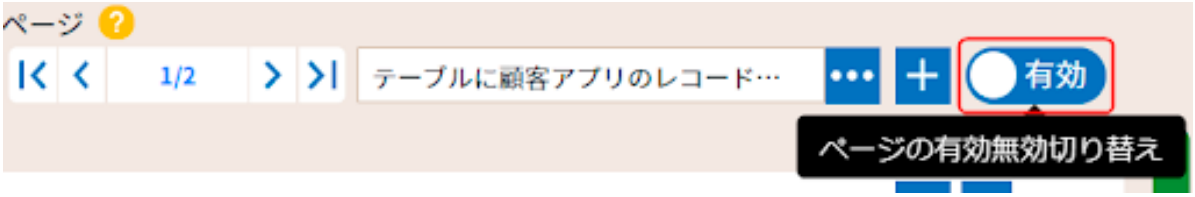
アクションの複製は下記のメニューから行ってください。



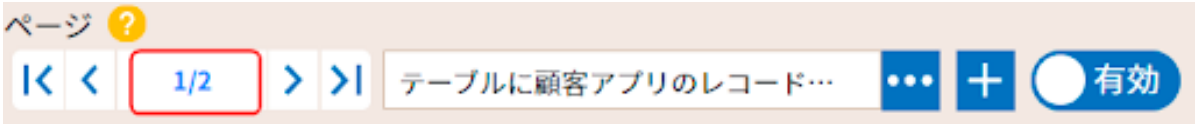
ページの無効化

個別のアクションではなく、ページをまるごと無効化・有効化することができます。
無効化したページは、「kintone アプリへ登録」する際に kintone アプリに登録されず、実行されません。

ページの無効化は、カスタマイズ画面右上の 有効 をクリックすると無効に切り替えできます。



または、ページの選択・並べ替え をクリックすると表示される ページ選択 画面でも無効化できます。



このページ選択 画面では、複数ページをまとめて設定できます。ページが一覧で表示されますので、どのページが有効・無効になっているか確認する場合にも便利です。

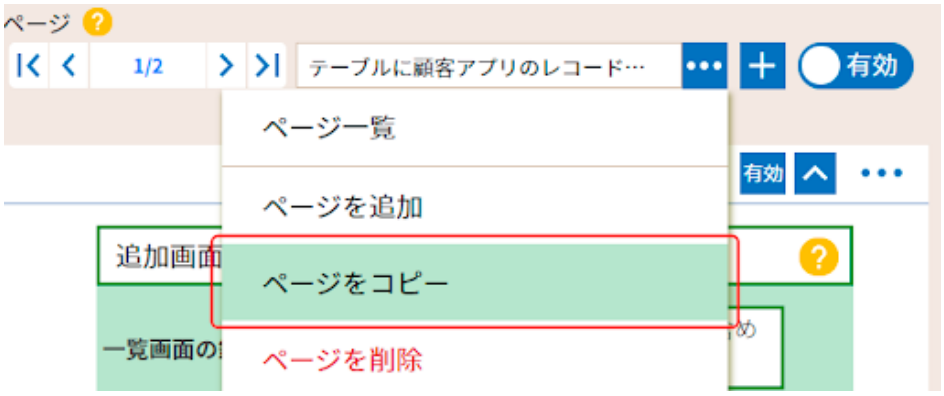
ページメニューから無効化することもできます。

ページの無効化は、他のページの動作を確認したいときに、一旦ページ全体を無効化しておきたい場合などに使用します。この機能を効果的に使うためには、機能ごと、画面ごとにこまめにページを分けてカスタマイズを作成しておくことが必要です。

またページの無効化は、次に記載するページのコピーと組み合わせて使用することも多い機能です。

ページのコピー

ページ内のカスタマイズに変更を加える場合は、ページをコピーして無効化し、変更前のカスタマイズの状態を保存しておくことをおすすめします。
変更前の状態を保存しておけば、カスタマイズの変更に失敗した場合などに、コピーしておいたページを有効化し、変更したページを無効化することでカスタマイズを変更前の状態に簡単に戻すことができます。

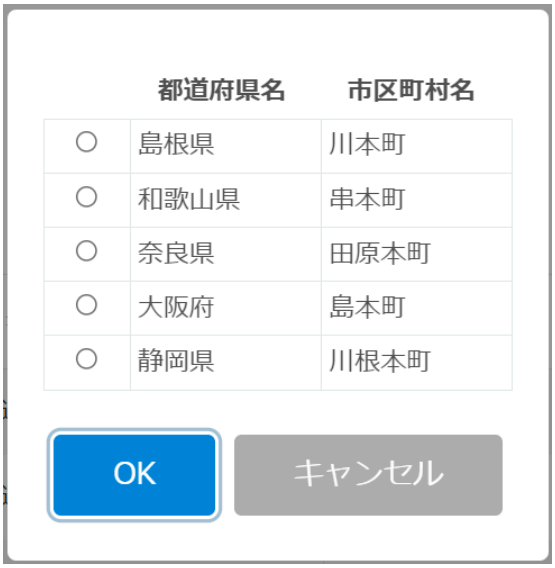


6-5. レコードが想定通り取得・絞り込みできているか ポップアップ表示して確認する

[やること：**キーを指定してレコードを取得する**] や [やること：**クエリで条件を指定してレコードを取得する**] のように、条件を指定してレコードを取得するやることを使用した場合や、取得したレコードを [やること：**取得したレコードを絞り込む**] で絞り込んだ場合、想定通りのレコードが取得・絞り込みできているか確認したい事がよくあります。

そういった場合は、本書の例でも取り上げているように取得・絞り込んだ後のレコードを [やること：**レコードの一覧をポップアップで表示する**] で確認してみてください。

このように、絞り込んだ結果を一覧で表示できるため、想定通りの絞り込みが行われているかを一目で確認することが可能です。



確認のための仮のアクションですので、確認が終わった時にはアクションを無効にしておけば、仮にそのあと条件が変わった場合など、再確認の際にも役立ちます。

6-6. テスト用アプリ作成

カスタマイズをガンガン試してと言われても、既にユーザーが使っている kintone アプリ (以下 本番用アプリ) にカスタマイズすると、誤ってレコードを更新してしまうリスクがあったり、フィールドの値やステータスなど、カスタマイズを試したい状態のレコードがなく実行確認が難しいことなどがよくあります。
そういった時のために、実際にユーザーが使うアプリとは別に、そのアプリをコピーしたテスト用アプリを作成することをおすすめします

gusuku CUSTOMINE

レコード更新編