新しいワクチン作成

この章は必要になったときにお読みください。

一般化ワクチン名は**ワクチン種類名**でその種類には１個のみ定義してください。以下のワクチンは定義済みです。

一般的なもの

麻しん風しんワクチン
日本脳炎ワクチン
四種混合ワクチン
BCGワクチン
Hibワクチン
サーバリックス
ガーダシル
二種混合ワクチン
おたふくかぜワクチン
水痘ワクチン
ロタリックス
ロタテック
インフルエンザワクチン
HBワクチン
肺炎球菌ワクチン
風しんワクチン
麻しんワクチン
HA
DTaP
COVID-19 （VRS）
成人用肺炎球菌

※ワクチン接種記録システムVRS
新型コロナワクチンの際に、自治体が独自でばらばらに記録していた予防接種記録を一つにするため、官邸主導でDXの一貫として始まった記録システムです。自治体での接種実績をVRSにアップロードが必要となるため、市町村コードや接種券番号あるいは接種ロットなど予防接種に関する記録をAPIあるいはCSVにて提出するものと思われます。コロナワクチンからバーコードの付いた接種券が配布されるようになっています。当ワクチンProでもこのシステムを将来的に使えるようにしています。

（以上は公費の請求が出来るように定義済み）

その他

成人用三種混合

~~成人用肺炎球菌~~

不活化ポリオワクチン

4価髄膜炎菌ワクチン

黄熱ワクチン

乾燥細胞培養痘そうワクチン

乾燥組換え帯状疱疹ワクチン（チャイニーズハムスター卵巣細胞由来）

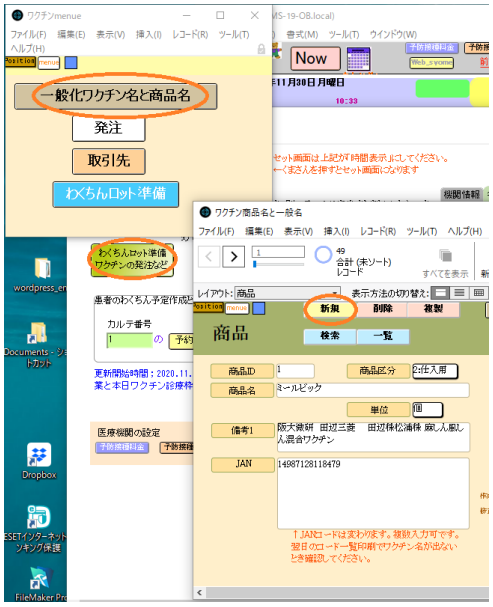
乾燥組織培養不活化狂犬病ワクチン

成人用沈降ジフテリアトキソイド

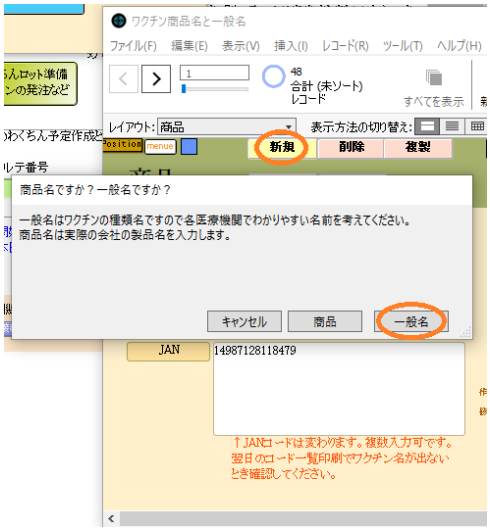
沈降破傷風トキソイド

（以上は予約は出来ますが、公費請求としての定義は出来ていません。）

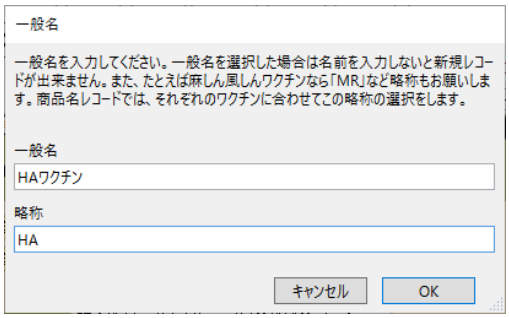
ここでは A 型肝炎ワクチンを追加してみます。ワクチンロット準備ワクチン発注など＞一般化ワクチン名と商品名＞新規と進めます。(図)



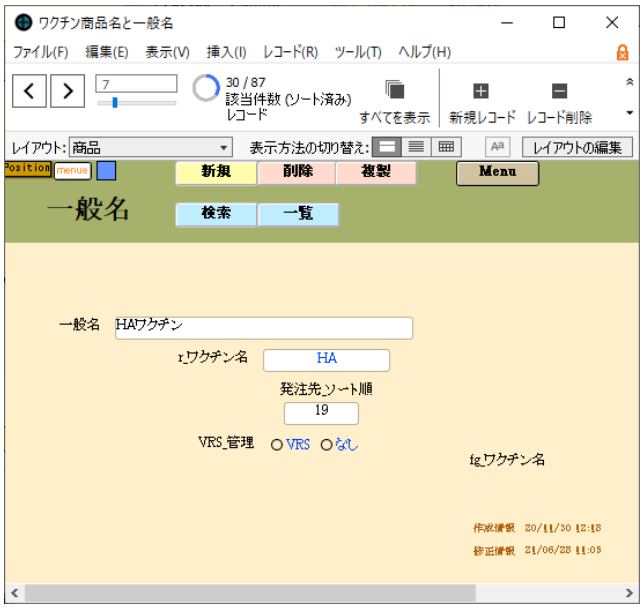
ダイアログが現れます(図)ので「一般名」を選択します。



(図)が次の画面です。A 型肝炎に対する一般名（HA ワクチン）と他のワクチンに重ならないように略称（HA）を付けます。



「作成」を押すと一般名がセットされます。(図)



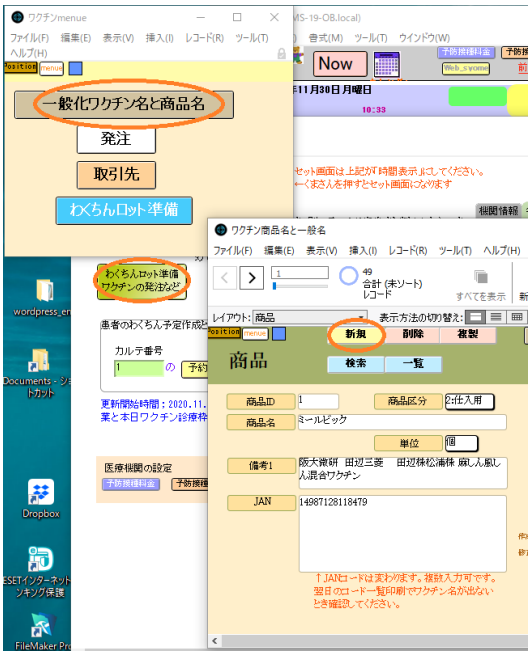
下部にある VRS は新型コロナのようにバーコード付の接種券が配布されるものは「VSR」にチェックを入れておくと、バーコードリーダーにて接種券番号などの読み取りが出来るようになりま

す。

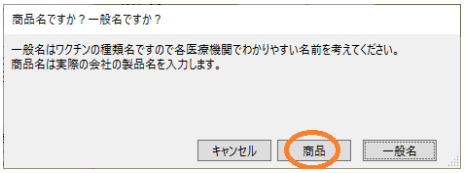
商品名（製品名）データの作成

発注表への登録　例；A 型肝炎ワクチン

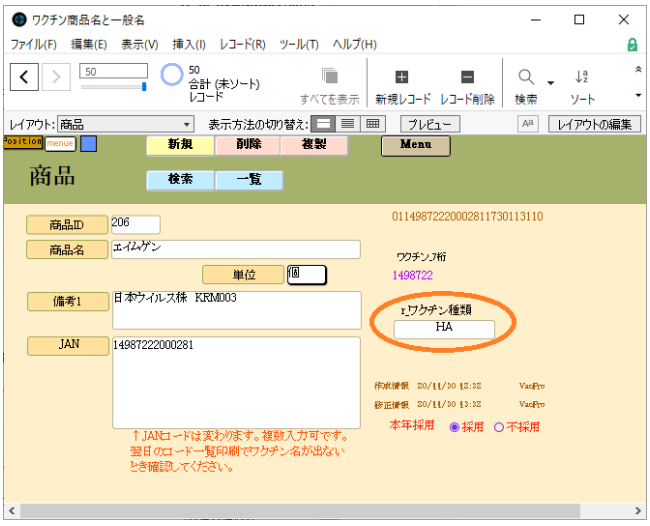
ここではA型肝炎ワクチンの商品名データを追加してみます。ワクチンロット準備ワクチン発注など＞一般化ワクチン名と商品名＞新規と進めます。(図)



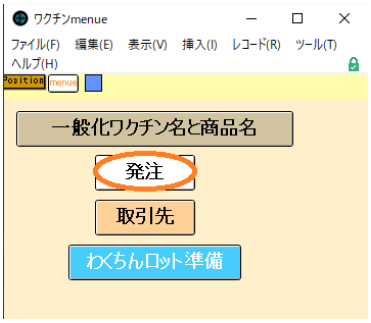
ダイアログ(図)が出ますので「商品」を選択します。



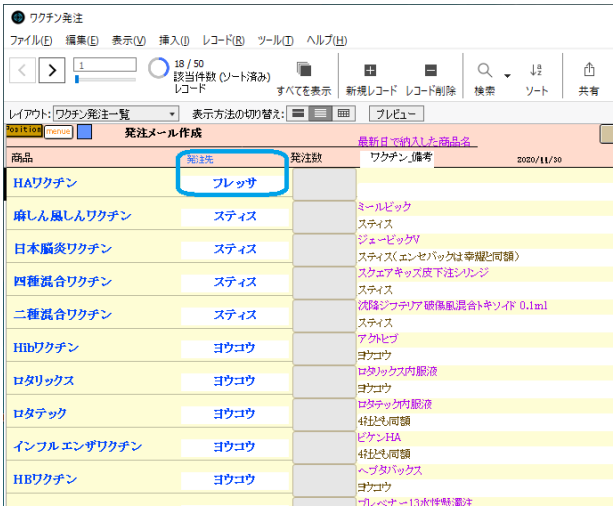
商品名、個数、備考、JAN コード、ワクチン種類、採用不採用を入力します。(図)**採用をチェックすると、r_ワクチン種類が表示されて、発注時のワクチンの商品名に反映します。**



次に発注表を見てみます。「発注」ボタンを押します。(図)



発注先や必要あれば備考を入力しておいてください。(図)



共通

フィールド定義）

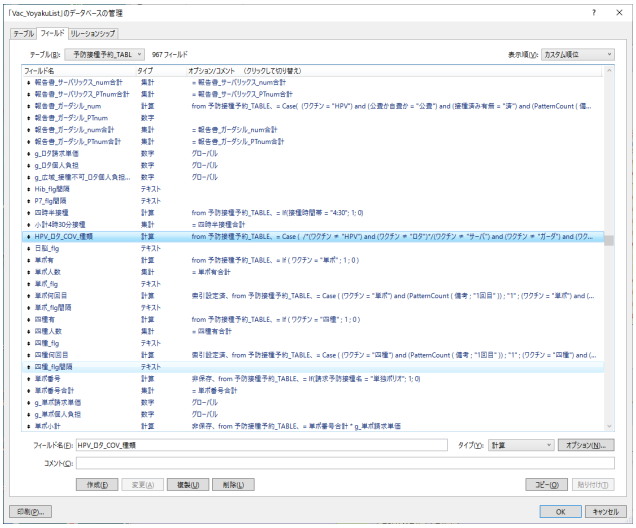
製品が1つのワクチン種に複数存在するとき　例；COVID−19 追加
ロタウイルス用のワクチンならロタリックス、ロタテックがあり、新型コロナウイルス用のワクチンならファイザー、モデルナ、アストラゼネカなど多数ある場合のフィールド定義です。

フィールド　HPV_ロタ_COV_種類（名前も「_COV」をその時に追加）は、ワクチンによっては製品が複数ある時にきちんと表示して間違わないようにするフィールドです。カルテ(図)

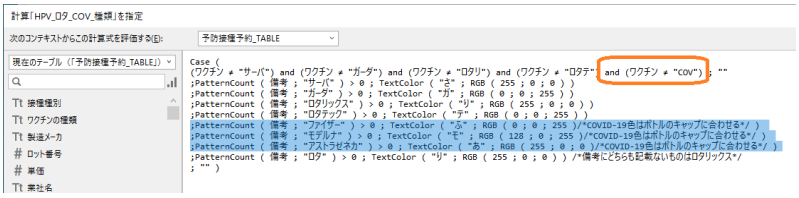
(図)予防接種一覧(例はアストラゼネカ社のもので「あ」の表示が出ています。)
フィールドの場所）

Vac_YoyakuList>予防接種予約_TABLE> フィールド定義HPV_ロタ_COV_種類

HPV_ロタ_COV_種類は、カスタム順位の表示順で中ほどにありますので探してください。(図)



COVID-19 ではボトルのふたの色が違いますので、間違ったときに少しでも気づくようにふたの色で字の色を調節しています。(図)



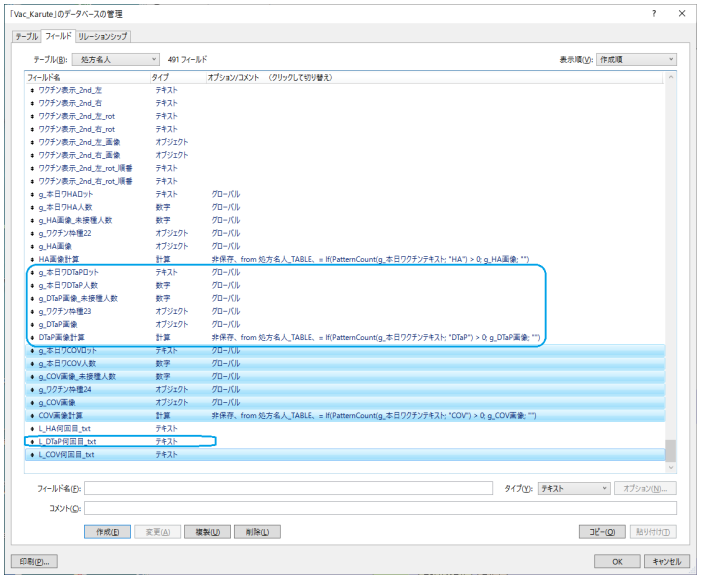
```
Case (
(ワクチン ≠ "サーバ") and (ワクチン ≠ "ガーダ") and (ワクチン ≠ "ロタリ") and (ワクチン ≠ "ロタテ" and (ワクチン ≠ "COV") ; ""
;PatternCount ( 備考 ; "サーバ" ) > 0 ; TextColor ( "さ" ; RGB ( 255 ; 0 ; 0 ) )
;PatternCount ( 備考 ; "ガーダ" ) > 0 ; TextColor ( "ガ" ; RGB ( 0 ; 0 ; 255 ) )
;PatternCount ( 備考 ; "ロタリックス" ) > 0 ; TextColor ( "リ" ; RGB ( 255 ; 0 ; 0 ) )
;PatternCount ( 備考 ; "ロタテック" ) > 0 ; TextColor ( "テ" ; RGB ( 0 ; 0 ; 255 ) )
;PatternCount ( 備考 ; "ファイザー" ) > 0 ; TextColor ( "ふ" ; RGB ( 0 ; 0 ; 255 )/*COVID-19色はボトルのキャップに合わせる*/ )
;PatternCount ( 備考 ; "モデルナ" ) > 0 ; TextColor ( "モ" ; RGB ( 128 ; 0 ; 255 )/*COVID-19色はボトルのキャップに合わせる*/ )
;PatternCount ( 備考 ; "アストラゼネカ" ) > 0 ; TextColor ( "あ" ; RGB ( 255 ; 0 ; 0 )/*COVID-19色はボトルのキャップに合わせる*/ )
;PatternCount ( 備考 ; "ロタ" ) > 0 ; TextColor ( "リ" ; RGB ( 255 ; 0 ; 0 ) ) /*備考にどちらも記載ないものはロタリックス*/
; "" )
```

ロットなどワクチン表示用

新しいワクチンの際には必要な一連のフィールド定義がありますので既存のワクチンのフィールド定義群を煮付けて複製し、改変します。

Vac_Karute>処方名人＞ フィールド定義の追加

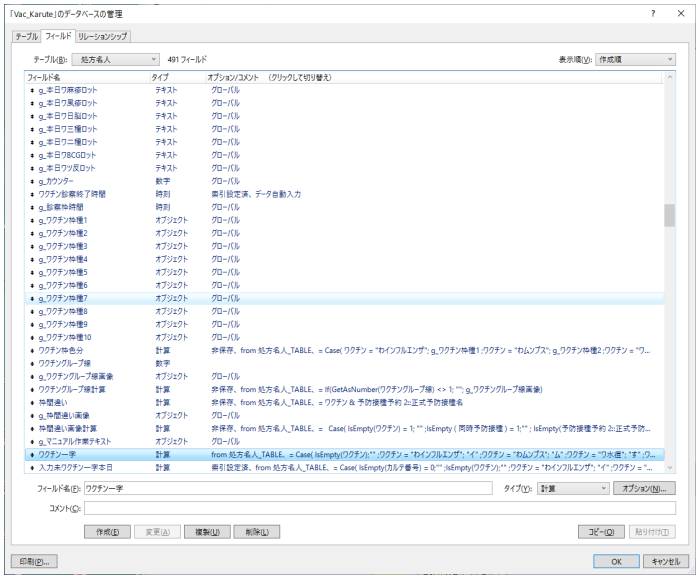
表示順の作成順で、一番下の方にあるDTaPのフィールド群を右下のボタン 【コピー】と【貼り付け】で複製しCOVID-19用にしています。(図)



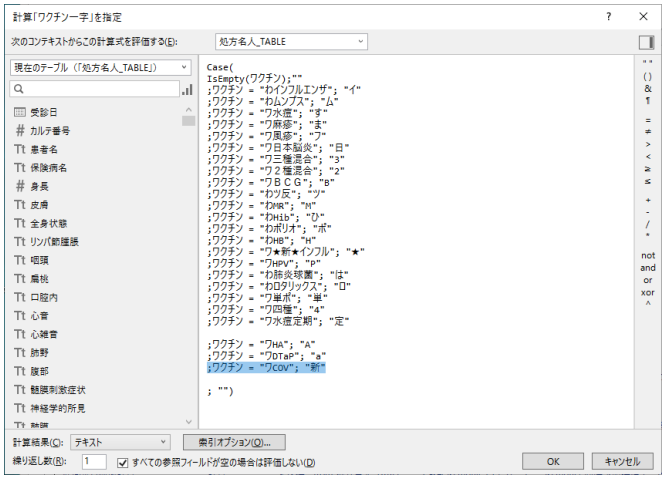
フィールド「ワクチン一字 」の変更 例；COVID-19 追加

Vac_Karute＞処方名人＞ フィールド

表示順「作成順」で(図)のような位置にありますので、開いてください。



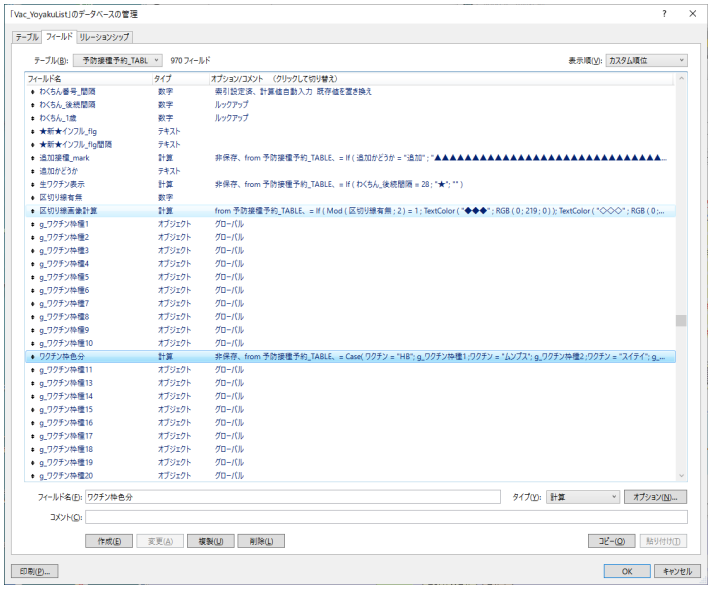
(図 2)のように COVID-19 のものを追加します。



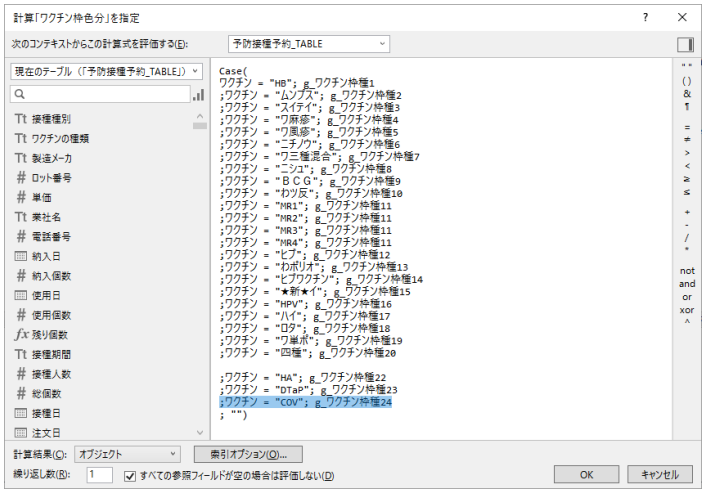
ワクチン枠色分の変更　例；COVID-19 追加

Vac_YoyakuList>予防接種予約_TABLE>　フィールド定義

表示順「カスタム順位」で下 1/3 位のところのワクチン枠色分を開きます。(図)



COVID-19　用に加えます。（前もってフィールド定義で「g_ワクチン枠種　24」も加えておきました。）(図)

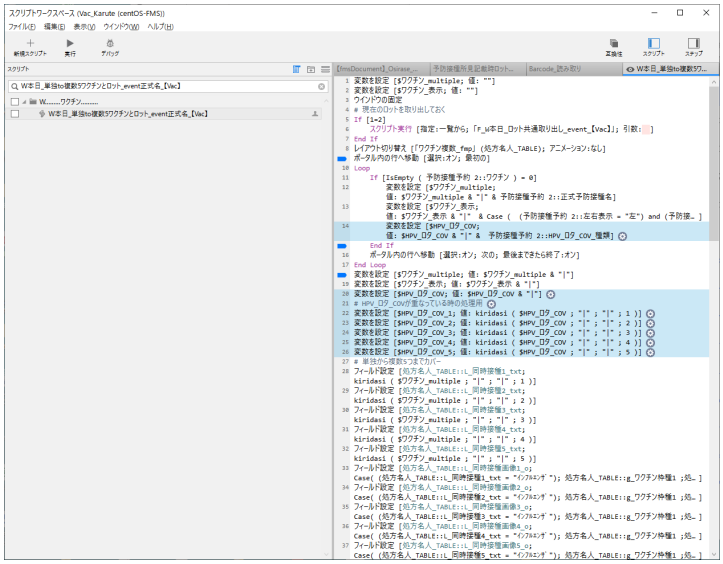

計算「ワクチン枠色分」を指定
次のコンテキストからこの計算式を評価する(E):
予防接種予約_TABLE
現在のテーブル（「予防接種予約_TABLE」）
接種種別
ワクチンの種類
製造メーカ
ロット番号
単価
業社名
電話番号
納入日
納入個数
使用日
使用個数
残り個数
接種期間
接種人数
総個数
接種日
注文日
Case(
ワクチン = "HB"; g_ワクチン枠種1
;ワクチン = "ムンプス"; g_ワクチン枠種2
;ワクチン = "スイテイ"; g_ワクチン枠種3
;ワクチン = "ワ麻疹"; g_ワクチン枠種4
;ワクチン = "ワ風疹"; g_ワクチン枠種5
;ワクチン = "ニチノウ"; g_ワクチン枠種6
;ワクチン = "ワミ種混合"; g_ワクチン枠種7
;ワクチン = "ニシュ"; g_ワクチン枠種8
;ワクチン = "ＢＣＧ"; g_ワクチン枠種9
;ワクチン = "ひツ反"; g_ワクチン枠種10
;ワクチン = "MR1"; g_ワクチン枠種11
;ワクチン = "MR2"; g_ワクチン枠種11
;ワクチン = "MR3"; g_ワクチン枠種11
;ワクチン = "MR4"; g_ワクチン枠種11
;ワクチン = "ヒブ"; g_ワクチン枠種12
;ワクチン = "わポリオ"; g_ワクチン枠種13
;ワクチン = "ヒブワクチン"; g_ワクチン枠種14
;ワクチン = "★新★イ"; g_ワクチン枠種15
;ワクチン = "HPV"; g_ワクチン枠種16
;ワクチン = "ハイ"; g_ワクチン枠種17
;ワクチン = "ロタ"; g_ワクチン枠種18
;ワクチン = "ワ単ポ"; g_ワクチン枠種19
;ワクチン = "四種"; g_ワクチン枠種20
;ワクチン = "HA"; g_ワクチン枠種22
;ワクチン = "DTaP"; g_ワクチン枠種23
;ワクチン = "coV"; g_ワクチン枠種24
; "")
not
and
or
xor
計算結果(C):
オブジェクト
索引オプション(O)...
繰り返し数(R):
1
すべての参照フィールドが空の場合は評価しない(D)
OK
キャンセル

スクリプト定義）

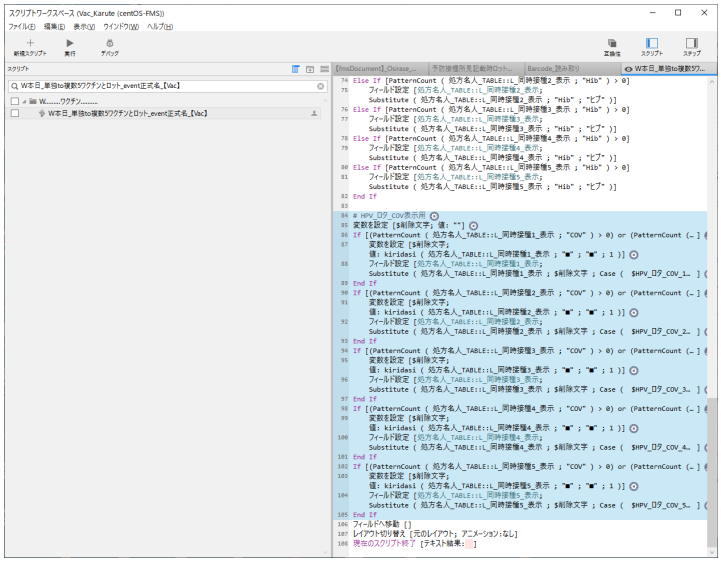
W 本日_単独 to 複数 5 ワクチンとロット_event 正式名_【Vac】の変更　例；COVID-19 追加

Vac_Karute＞スクリプトワークスペース

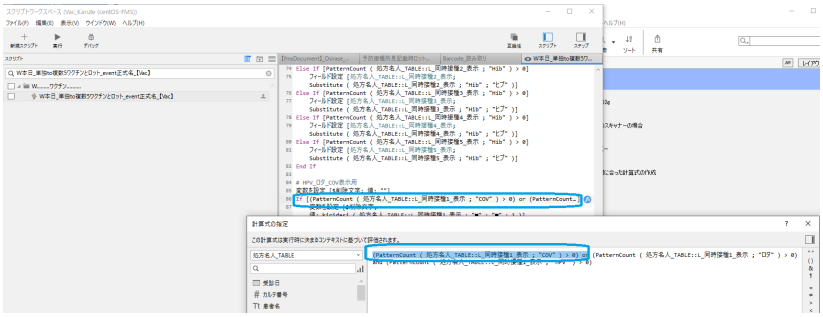
上記の「**W 本日_単独 to 複数 5 ワクチンとロット_event 正式名_【Vac】**」をコピーしてスクリプトワークスペースの検索スペースにペーストして見つけます。(図)



青く表示されたところが今回の見直す部分ですが、(図)



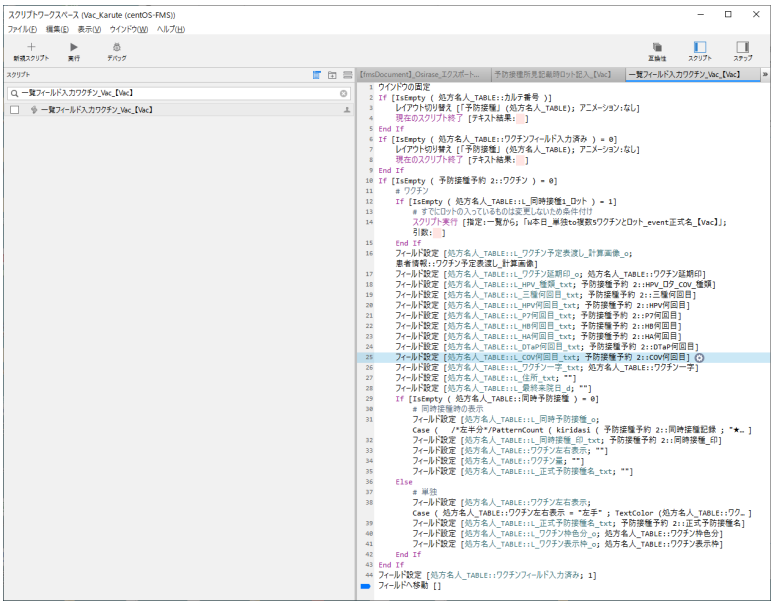
スクリプト下方の部分 5 ヶ所を(図)のように訂正します。ワクチン Pro では 5 個まで同時に出来るようにしています。



一覧フィールド入力ワクチン_Vac_【Vac】の変更　例；COVID-19 追加

Vac_Karute＞スクリプトワークスペース

「**一覧フィールド入力ワクチン_Vac_【Vac】**」をコピーしてスクリプトワークスペースの検索スペースにペーストして見つけます。(図)

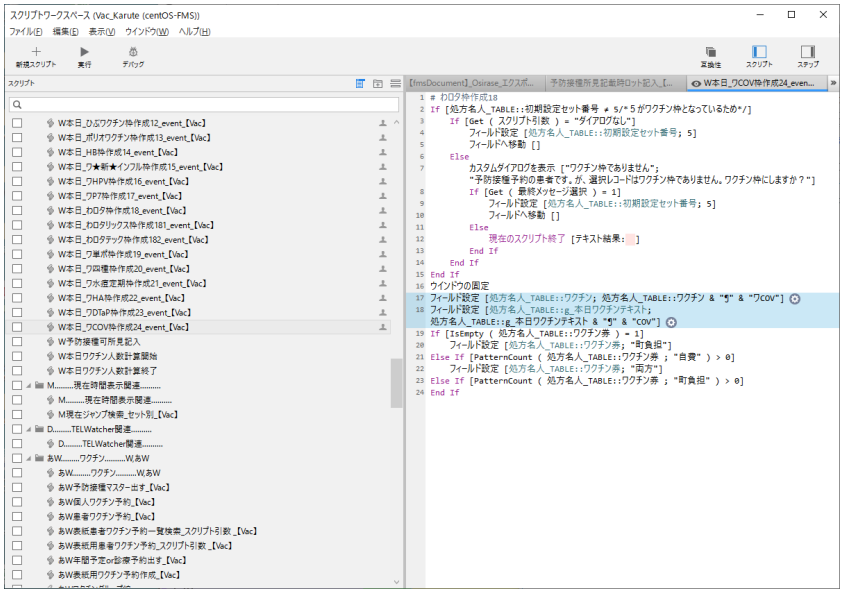


青く表示されたところが今回の見直す部分です。COVID-19 の部分を追加しています。

W 本日_ワ COV 枠作成 24_event_【Vac】の変更　例；COVID-19 追加

Vac_Karute＞スクリプトワークスペース

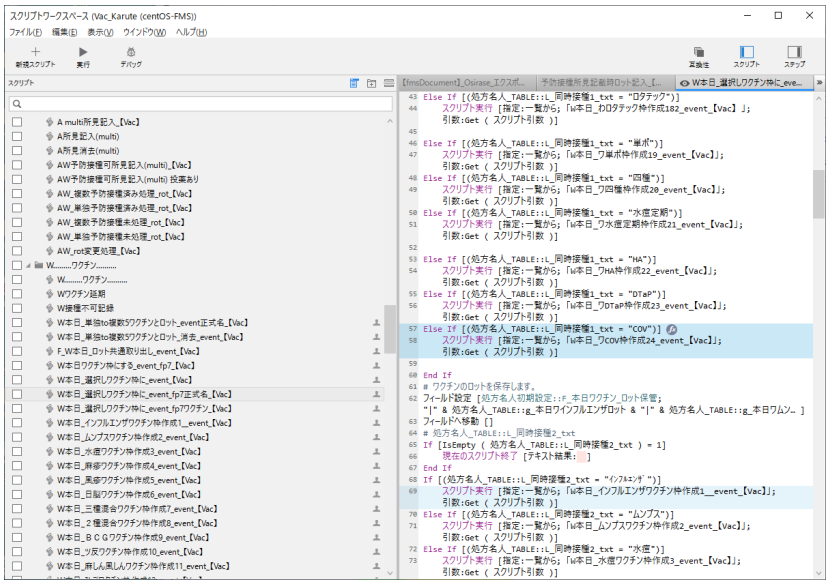
スクリプト　W 本日_ワ DTaP 枠作成 23_event_【Vac】をコピー(ctrl+c)し、貼り付け(ctrl+v)してその名前や内容を COVID-19 用に変更します。(図)



W 本日_選択しワクチン枠に_event_fp7 正式名_【Vac】の変更　例；COVID-19 追加

Vac_Karute＞スクリプトワークスペース

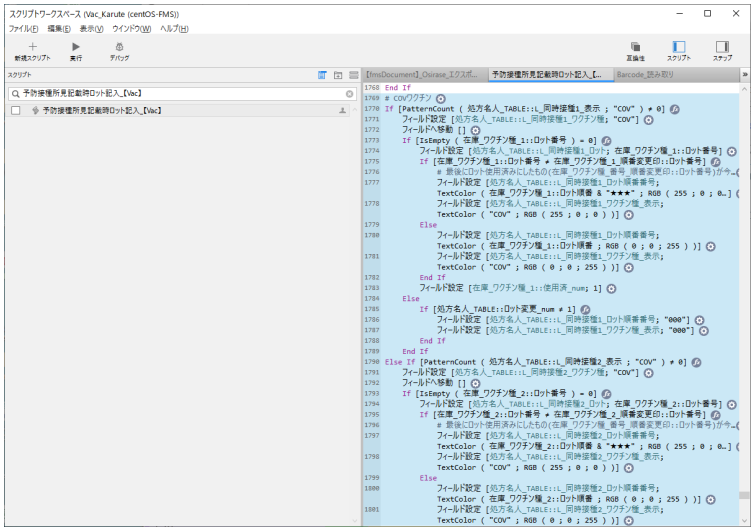
W 本日_選択しワクチン枠に_event_fp7 正式名_【Vac】に(図)のように前項 COVID-19 の「W 本日_ワ COV 枠作成 24_event_【Vac】」を追加します。同時に5までの表示が出来るようになっており、図はワクチン1のものですが、5 までありますので、下に下ろしてあと 4 ヶ所同様に追加します。



予防接種所見記載時ロット記入_【Vac】の変更　例；COVID-19 追加

Vac_Karute＞スクリプトワークスペース

上記の「**予防接種所見記載時ロット記入_【Vac】**」をコピーしてスクリプトワークスペースの検索スペースにペーストして見つけます。(図)

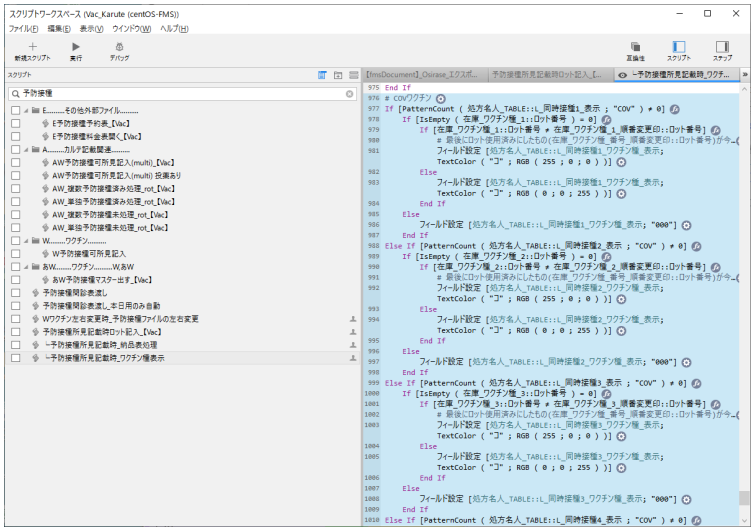


青く表示されたところが今回加える部分です。COVID-19 の上の DTaP をコピー(ctrl+c)し、貼り付け(ctrl+v)して内容を変更しました。

└予防接種所見記載時_ワクチン種表示の変更　例；COVID-19 追加

Vac_Karute＞スクリプトワークスペース

「**予防接種**」をスクリプトワークスペースの検索スペース入力すると(図)のようになります。

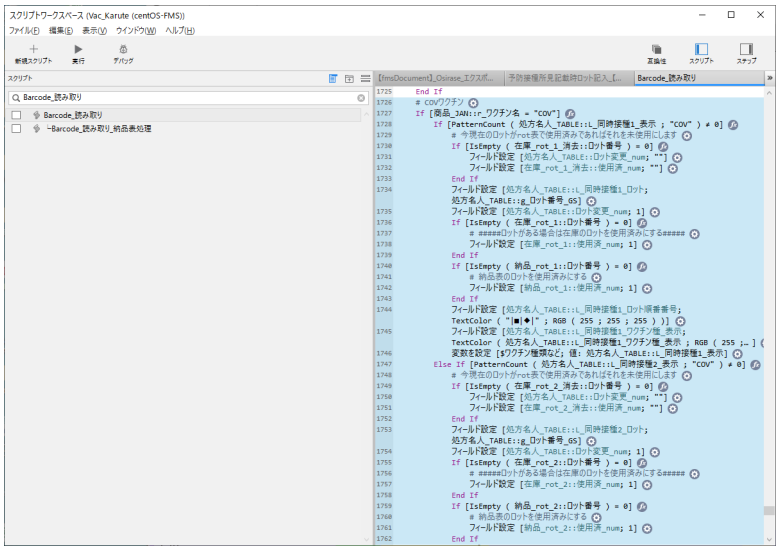


青く表示されたところが今回加える部分です。COVID-19 の上の DTaP の部分をコピー(ctrl+c)し、貼り付け(ctrl+v)して内容を変更しました。

Barcode_読み取りの変更　例；COVID-19 追加

Vac_Karute＞スクリプトワークスペース

上記の「**Barcode_読み取り**」をコピーしてスクリプトワークスペースの検索スペースにペーストして見つけます。（図）

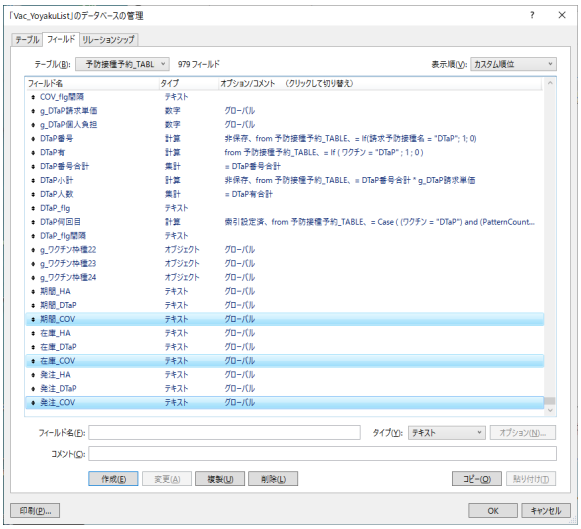


ワクチン在庫計算から発注

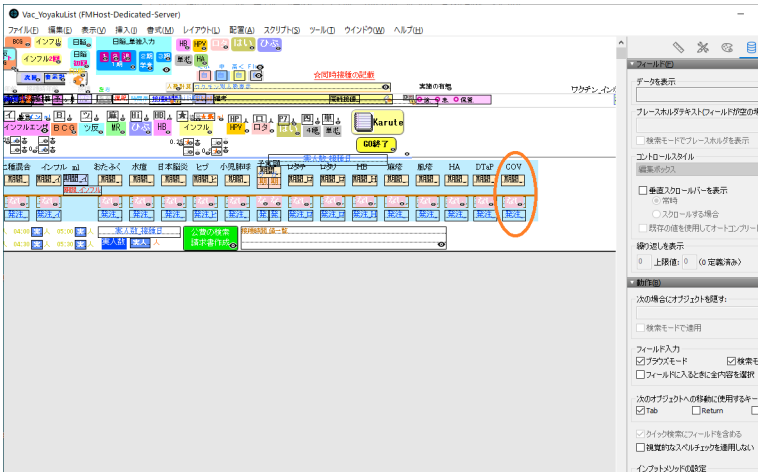
フィールド定義レイアウト変更スクリプト定義）

Vac_YoyakuList のフィールド定義とレイアウトとスクリプトの変更　例；COVID-19 追加

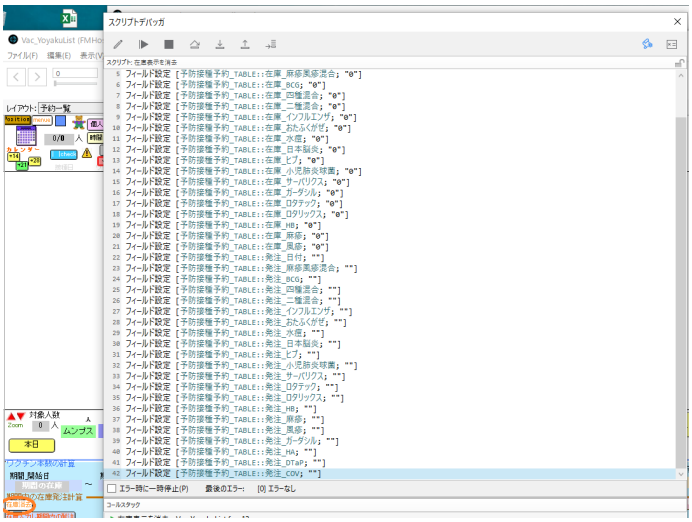
予防接種予約_TABLE で期間_COV、在庫_COV、発注_COV を、(図)のように DTaP 用のものを複製したりして作成します。（テキストグローバル）



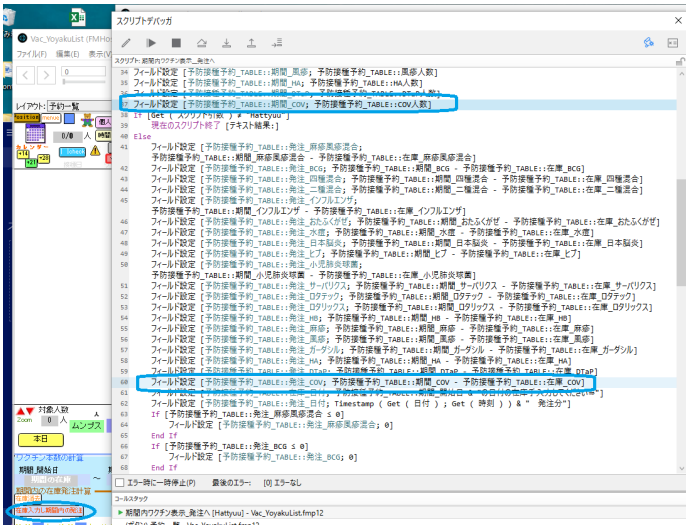
予約一覧レイアウトに加えます。(図)

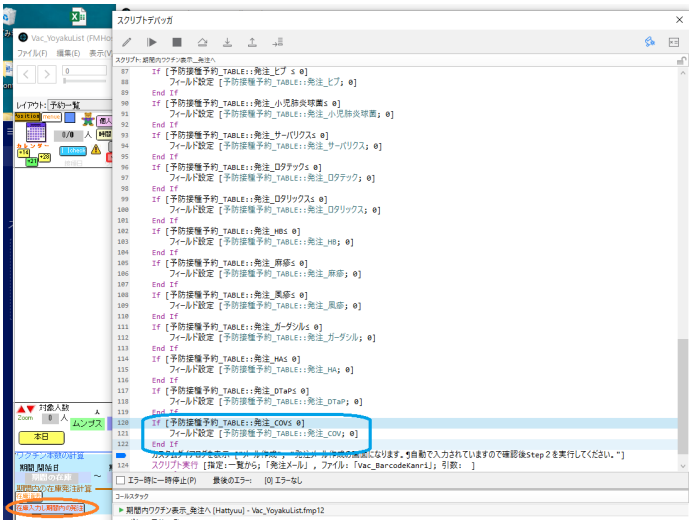


スクリプト「在庫表示を消去」にCOV用を追加。(図)



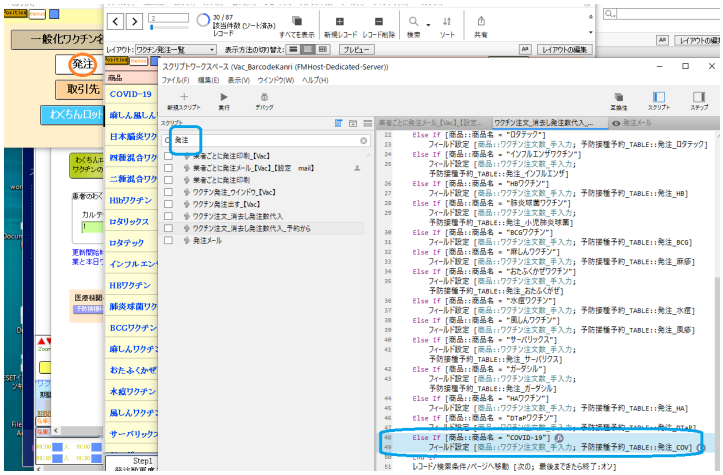
スクリプト「期間内ワクチン表示__発注へ」にCOV用を追加。(図)(図)





ワクチン発注のスクリプトの変更　例；COVID−19 追加

わくちんのメニューの「わくちんロット準備ワクチンの発注など」ボタン＞発注＞スクリプトワークスペースの検索に「発注」を入れて、(図)のようにスクリプト「ワクチン注文_消去し発注数代入_予約から」を追加します。



公費請求するワクチンに必要

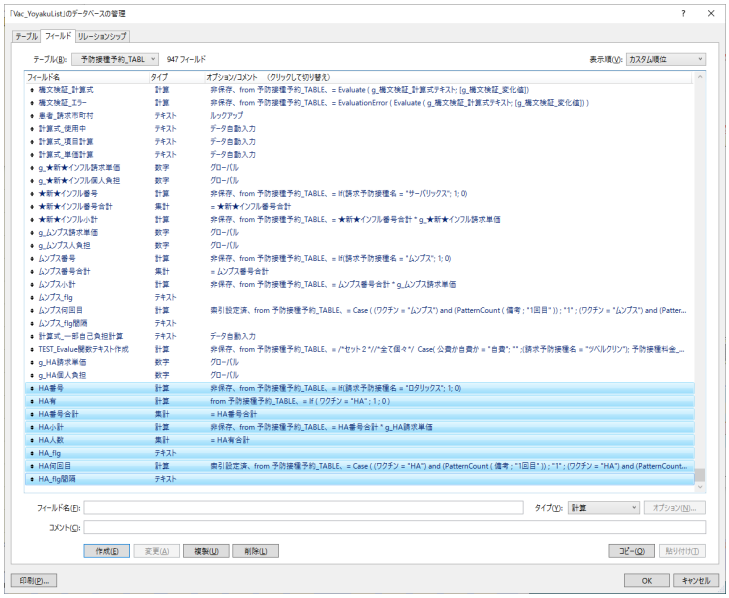
フィールド定義）

Vac_YoyakuLIst ファイルの予防接種予約_TABLE のフィールド定義(1)　例；A 型肝炎ワクチン

【ファイル名　Vac_YoyakuLIst】

（例：A型肝炎ワクチンは「HA」にしています）

フィールド定義(図)__新規に作成



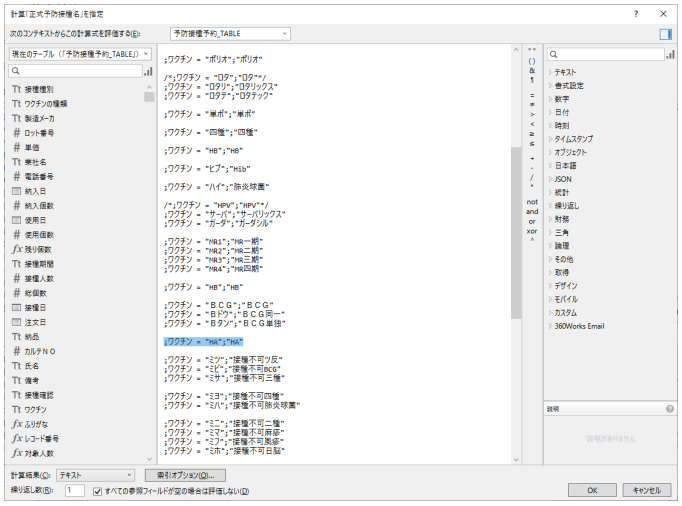
(新わくちん名)請求単価、(新わくちん名)個人負担、(新わくちん名)番号、(新わくちん名)有、(新わくちん名)番号合計、(新わくちん名)小計、(新わくちん名)人数、(新わくちん名)_flg、(新わくちん名)何回目、(新わくちん名)_flg間隔を新規にフィールド定義します。
（例；HAは請求単価、HA個人負担、HA番号、HA有、HA番号合計、HA小計、HA人数、HA_flg、HA何回目、HA_flg間隔とします。あなたが新しいワクチンを追加するときには、このロタリックス関連のフィールド定義を、請求単価からflg間隔までの間を全て選択し「コピー」ボタン、そのまま「貼り付け」ボタンを押すとコピーとして一番最後に追加されます。、その後新しいワクチンの名称で変更してください。

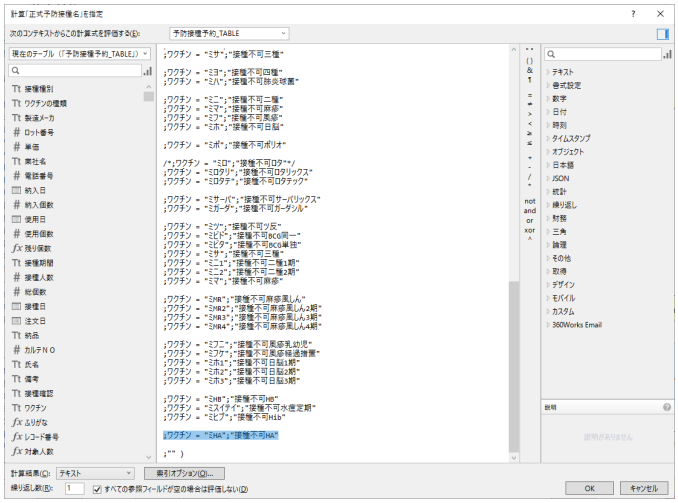
(新わくちん名)有と、(新わくちん名)何回目は計算内容も変えてください。この場合は

HA有ーーIf（ ワクチン ＝″HA″；1；0）

HA何回目ーーCase（（ワクチン ＝″HA″）and（PatternCount（ 備考 ；″1回目″））；
″1″
；（ワクチン ＝″HA″）and（PatternCount（ 備考 ；″2回目″））；
″2″
；（ワクチン ＝″HA″）and（PatternCount（ 備考 ；″3回目″））；
″3″
；″″）

フィールド定義(図)(図)__正式予防接種名の変更

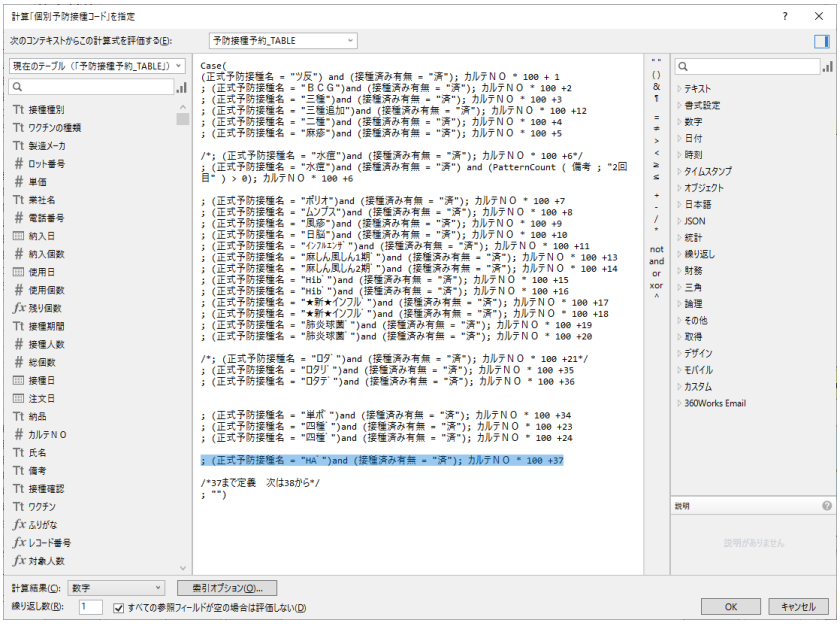




下記のワクチンと正式予防接種名の対応を定義を追加
;ワクチン ＝″HA″;″HA″

;ワクチン ＝″ミHA″;″接種不可HA″
これは公費請求を作成するときにも必要です。

重要フィールド定義(図)__個別予防接種コードの変更 この番号が重要



使用する新しいワクチンの使用コードを作成します。新しい順番を取ります。使用していないコードを使用してください。(HAは37番を取っています)

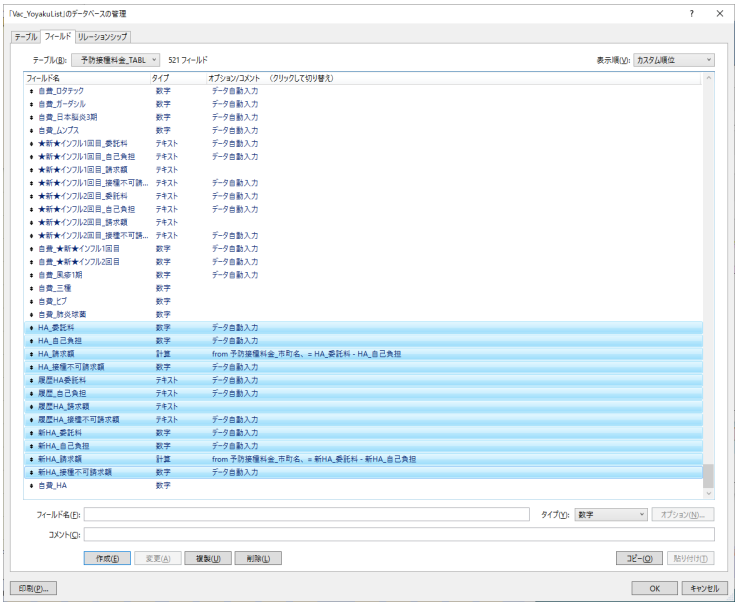
;(正式予防接種名 ＝″HA″)and (接種済み有無 ＝″済″); カルテNO ＊100 +37

料金表の作成 1　例;A 型肝炎ワクチン

わくちんのメニューの「予防接種料金」をクリックします。料金表が現れますのでフィールド定義します。(図)

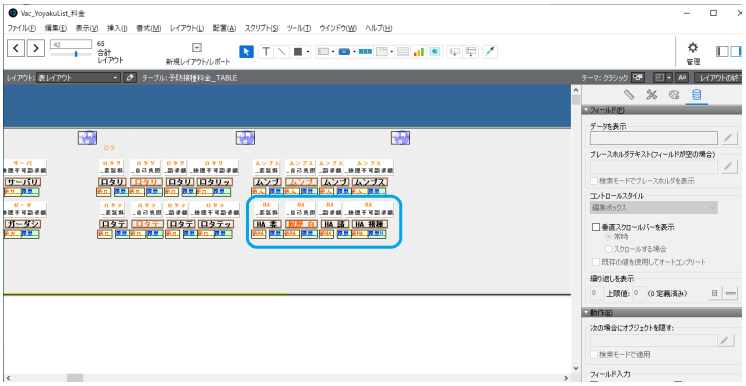
簡単にフィールド定義するためムンプス_委託料から履歴ムンプス_接種不可請求額までを「コ

ピー」(ctrl+c)して「貼り付け」(crrl+v)しました。HA に名前を変更します。(図)

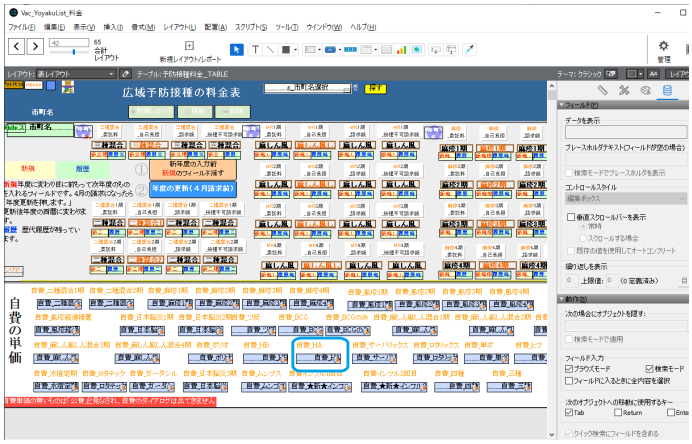


自費_(新しいワクチン)が連続でない場合もありますので作成します。（この場合無いので一番下の自費_HA を追加）

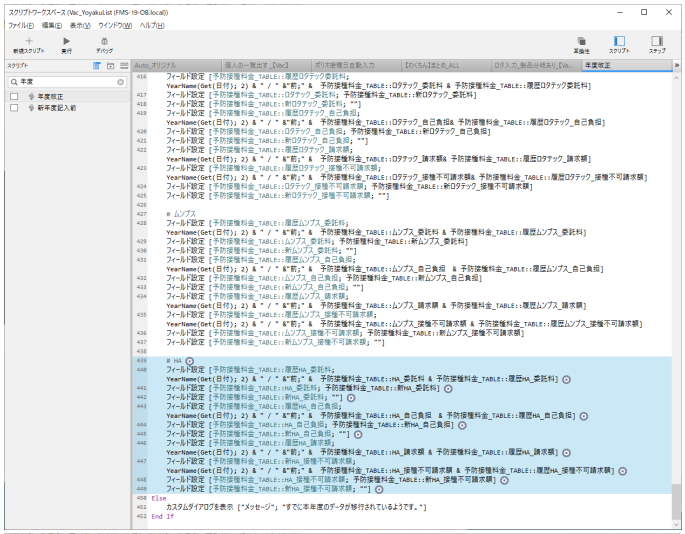
レイアウトモードにして HA の料金を加えます。ムンプス料金の部分をコピーしてその下に加えました。(図)

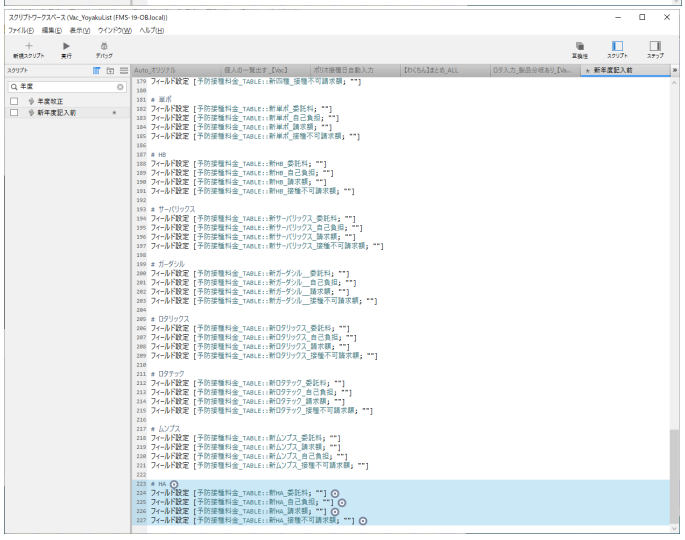


レイアウトモードにして自費部分もを加えます。自費_HA も加わりました。(図 4)



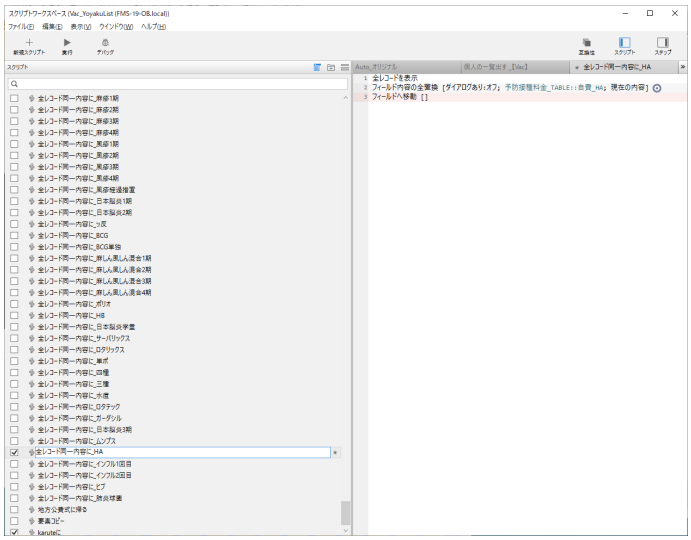
スクリプト「年度改正」と「新年度記入前」に他のワクチン同様に新しいワクチンの部分を既存のワクチン部分をコピーペーストなどをして加えます。(図)(図)





スクリプトトリガー用のスクリプトを準備します。今回は「全レコード同一内容に_ムンプス」をコ

ピーペーストして HA 用にしました。(図)



作ったスクリプトをレイアウトモードにてフィールド　自費_HA にトリガーとして定義しました。(図)

新しいワクチンを公費請求の計算式に加える

Vac_YoyakuList.fmp12 の「公費の検索請求書作成」を押すとアラートが出ますので、「計算式」を押します。(図)

(図)のように全ての式に新しいワクチン関連を加えて計算の確認を取り保存します。

同様に一部自己負担計算、請求単価計算も変更してください。

詳しくは,

ワクチン Pro マニュアル2.x.x「公費請求・編集」を参照してください。さぬき市のものがデフォルトになっていますので他の市区町村を全て削除して、複製し他の市町村用にした方が最も簡単です。

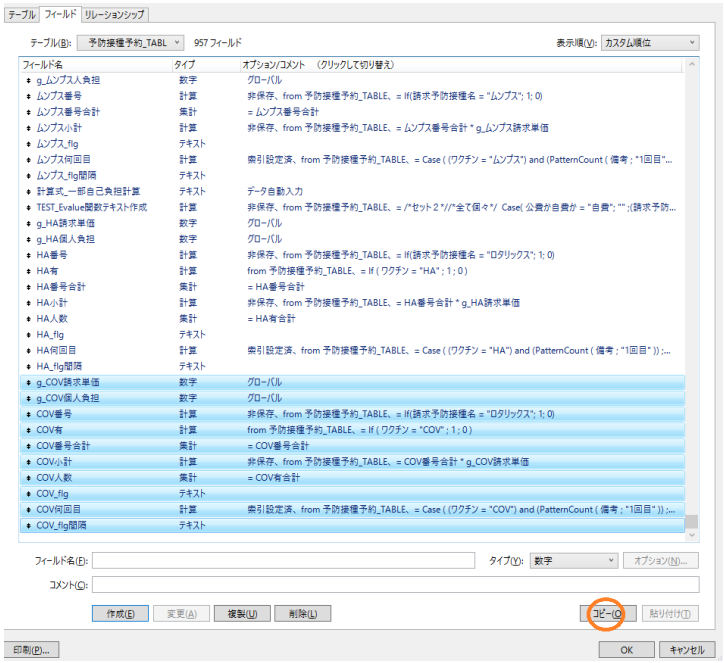
フィールド定義

Vac_YoyakuLIst ファイルの予防接種予約_TABLE のフィールド定義群の追加　例；

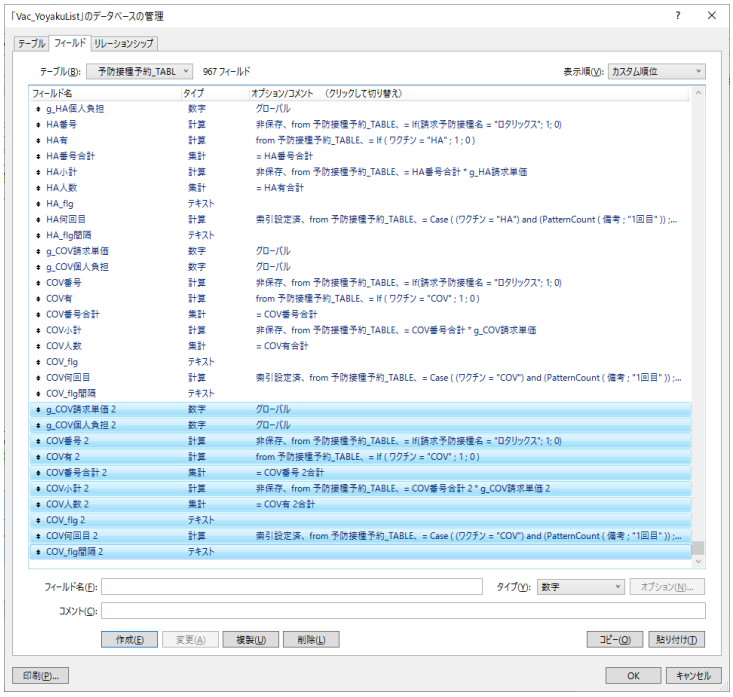
COVID-19　ワクチン Pro マニュアル2.x.x 2 章 4 項同内容

Vac_YoyakuLIst ファイルのファイル＞管理＞データベース(D)...を開きます。(図)

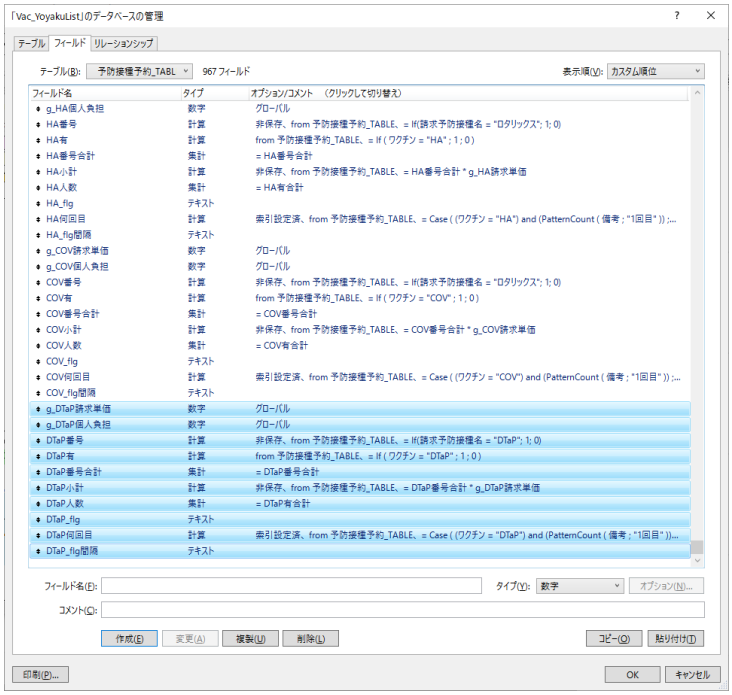
最後(COV)の一連のフィールド定義をShiftキーを使って選択し、赤枠のコピーをクリックします。(図)横の貼り付けをクリックします。



複製されました。(図)



目的のワクチン用に変更します。終了時(図)

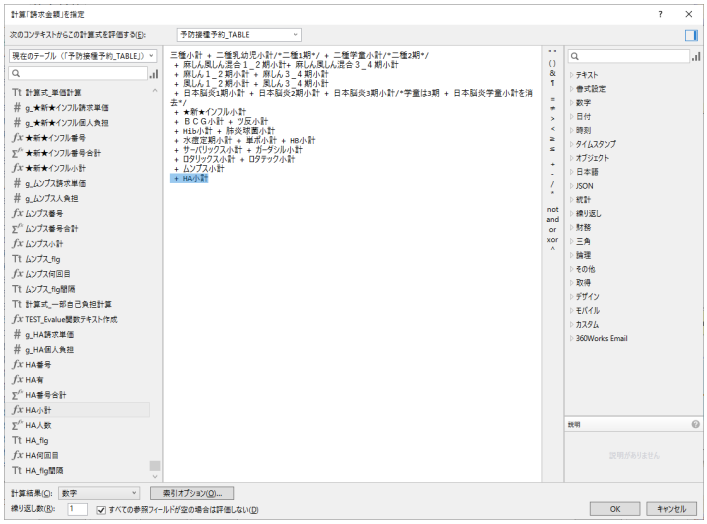


Vac_YoyakuLIstファイルの予防接種予約_TABLEのフィールド定義　例；A型肝炎ワクチン

フィールド定義(図)__**請求金額**の変更

HAの小計を加えています。

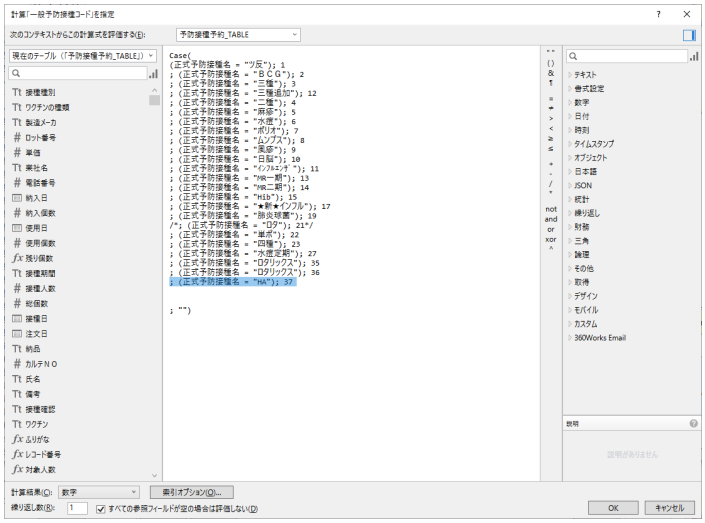
+ HA小計



フィールド定義(図)__一般予防接種コードの変更

HAの定義を加えています。

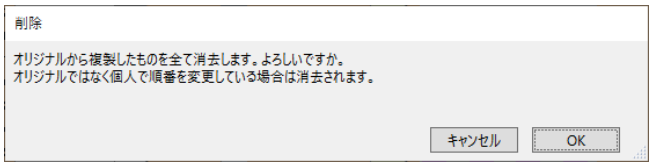
;(正式予防接種名 = ″HA″); 37



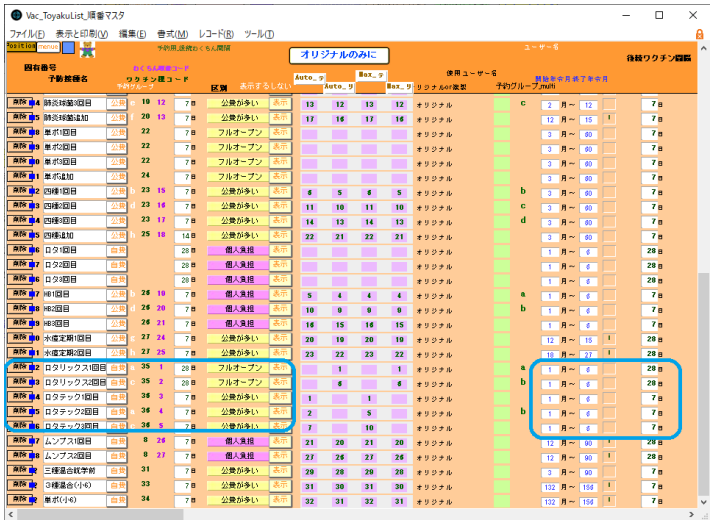
予約Robotに必要

わくちんのメニューから、「予防接種マスタ」ボタンを押します。順番マスタが出ます。オリジナルのみ残してそこにロタわくちんを加えます。(図)「オリジナルのみ」ボタンを押します。

オリジナルのみを残す旨のダイアログが出ます。(図)「OK」を押すと複製は全部消去されます。(クライアントの複製はそのクライアントが予約ROBOTを使った場合に自動で出来ていますが、これが消去されます。)



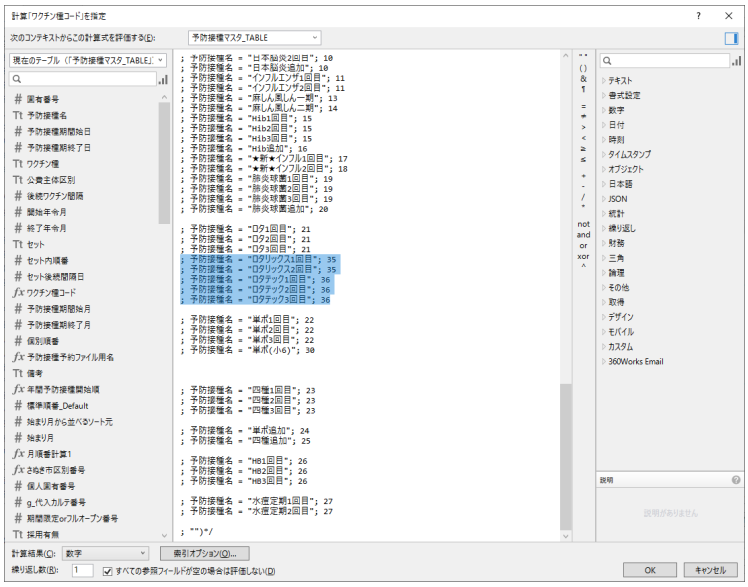
よく似たワクチンを複製して、それをロタウイルスワクチン用にして増やすレコード分増やして名前を自分で付け変更します。後のフィールド定義でこの名前を使用します。(図)



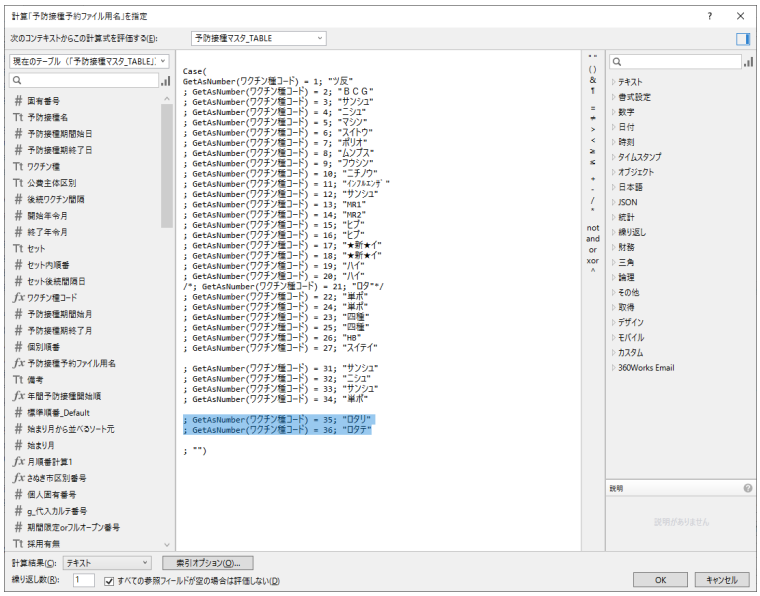
フィールド定義）Vac_YoyakuLIstファイルの予防接種マスター_TABLEのフィールド定義　例；ロタリックス　ロタテック

フィールド定義(図)__ワクチン種コードの変更
予防接種予約_TABLEの個別予防接種コードに対応出来るように同コードで登録します。

; 予防接種名 = "ロタリックス1回目"; 35
; 予防接種名 = "ロタリックス2回目"; 35
; 予防接種名 = "ロタテック1回目"; 36
; 予防接種名 = "ロタテック2回目"; 36
; 予防接種名 = "ロタテック3回目"; 36



フィールド定義(図)__予防接種ファイル用名の変更
予防接種予約_TABLEのワクチンに対応出来るように登録します。
; GetAsNumber(ワクチン種コード) = 35; "ロタリ"
; GetAsNumber(ワクチン種コード) = 36; "ロタテ"

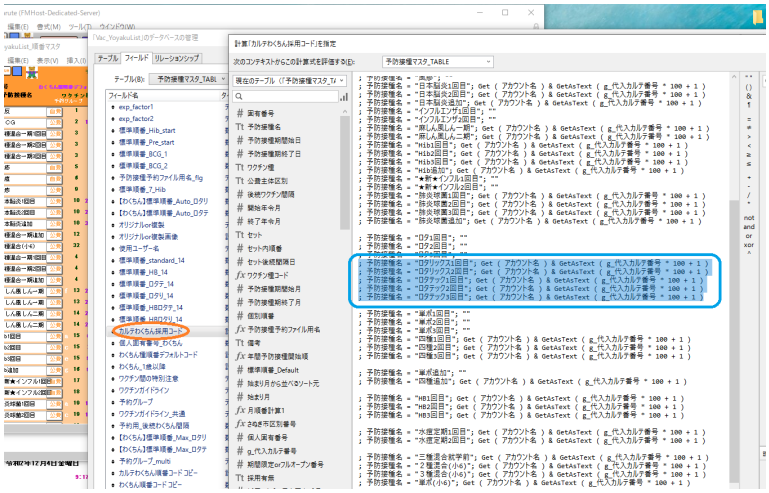


重要

フィールド定義(図)__カルテわくちん採用コード

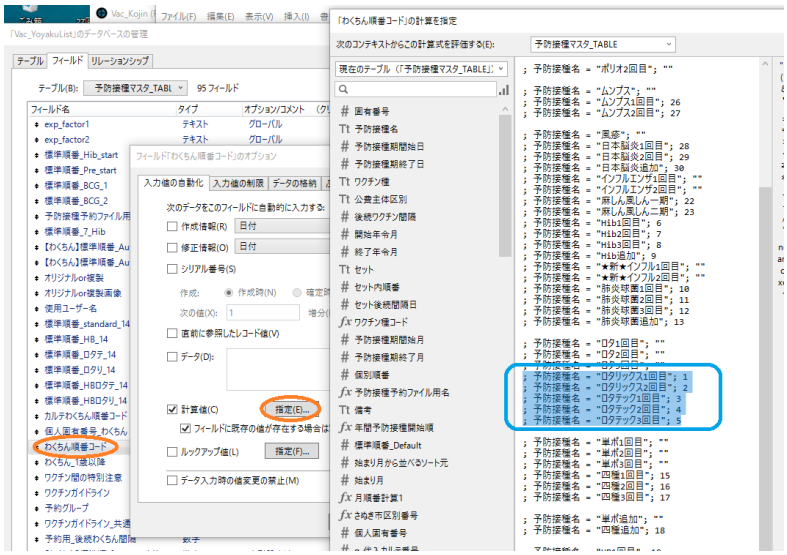
以下はこの予防接種マスター_TABLEから予防接種予約_TABLEにインポートする際に必要です。

; 予防接種名 = "ロタリックス1回目"; Get (アカウント名) & GetAsText (g_代入カルテ番号 * 100 + 1)
; 予防接種名 = "ロタリックス2回目"; Get (アカウント名) & GetAsText (g_代入カルテ番号 * 100 + 1)
; 予防接種名 = "ロタテック1回目"; Get (アカウント名) & GetAsText (g_代入カルテ番号 * 100 + 1)
; 予防接種名 = "ロタテック2回目"; Get (アカウント名) & GetAsText (g_代入カルテ番号 * 100 + 1)
; 予防接種名 = "ロタテック3回目"; Get (アカウント名) & GetAsText (g_代入カルテ番号 * 100 + 1)



フィールド定義(図)__わくちん種順番デフォルトコードの変更　ROBOTの予約作成時にワクチン種としての順番を決めます。そのワクチン種の１回目の順番で決定されますが、そのあとの2回目3回目などは１回目の順番に続くように取ってください。

; 予防接種名　= "ロタリックス1回目"; 1
; 予防接種名　= "ロタリックス2回目"; 2
; 予防接種名　= "ロタテック1回目"; 3
; 予防接種名　= "ロタテック2回目"; 4
; 予防接種名　= "ロタテック3回目"; 5

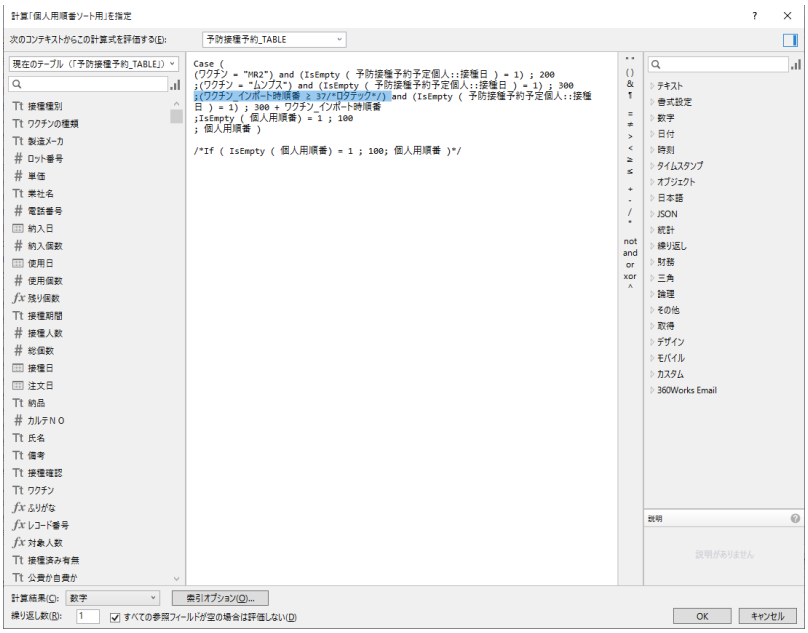


Vac_YoyakuLIstファイルの予防接種予約_TABLEのフィールド定義　ROBOTで上手く予定がはいらない場合のポイント

フィールド定義(図)__**個人用順番ソート用**の変更

Case (
(ワクチン = "MR2") and (IsEmpty (予防接種予約予定個人::接種日) = 1) ; 200
;(ワクチン = "ムンプス") and (IsEmpty (予防接種予約予定個人::接種日) = 1) ; 300
;(**ワクチン_インポート時順番 ≥ 37**/*ロタテック*/) and (IsEmpty (予防接種予約予定個人::接種日) = 1) ; 300 + ワクチン_インポート時順番
;IsEmpty (個人用順番) = 1 ; 100
; 個人用順番)

/*If (IsEmpty (個人用順番) = 1 ; 100; 個人用順番)*/



ROBOTで表示させるときポータル内でのソートの要素です。たとえば通常通りの順番で出すためロタリックスとロタテックを無視するように37以上にしています。

;(ワクチン_インポート時順番 ≥ 37/*ロタテック*/)

ワクチンProで使用されているカスタム関数

カスタム関数　kiridasi

FileMakerのデベロッパーさんが作成した関数です。ワクチンProで至る所で使用させてもらってます。

kiridasi（ テキスト ； 値1 ； 値2 ； 回数 ）

```
/*値1と値2で囲まれた文字列を切り出す関数                    */
/*何回目に登場したものを切り出すかを「回数」で指定する  */
Let ( [
  StartPos = Position ( テキスト ； 値1 ; 1 ； 回数 ) + 1 ;
  EndPos = Position ( テキスト ； 値2 ; StartPos ; 1 ) ];
  Middle ( テキスト ; StartPos ; EndPos − StartPos )
)
```

カスタム関数　workday

デベロッパーさんの作った関数です。Vac_ClinicYotei.fp12で利用させてもらってます。

workday (calcdate ; regularHoliday ; halfHoliday ; halfHoliday1 ; halfHoliday2 ; halfHoliday3 ; halfHoliday4 ; halfHoliday5 ; mayDay ; regularHoliday2)

```
// output encoding="UTF-8"
// line endings="LF(UNIX)"
// http://www.fmpro.jp
//  作者 ： 平松晋介
//  概要 ： 与えられた日が休日であれば  0  を、それ以外ならば  1  を返します。
//  カスタム関数  workday ( calcdate ; regularHoliday ; halfHoliday ; halfHoliday1 ;
halfHoliday2 ; halfHoliday3 ; halfHoliday4 ; halfHoliday5 ; mayDay )
```

```
Case (
        Year ( calcdate ) < 1948 ; "" ;
        Year ( calcdate ) ≥ 2100 ; "" ;

/*元日と3が日*/
        Month ( calcdate ) = 1 and Day ( calcdate ) = 1 ; 0 ;
        Month ( calcdate ) = 1 and Day ( calcdate ) = 2 ; 0 ;
        Month ( calcdate ) = 1 and Day ( calcdate ) = 3 ; 0 ;

/*【当院】12/30,31*/

        Month ( calcdate ) = 12 and Day ( calcdate ) = 30 ; 0 ;
        Month ( calcdate ) = 12 and Day ( calcdate ) = 31 ; 0 ;


/*成人の日*/
        Month ( calcdate ) = 1 and Year ( calcdate ) < 2000 and Day ( calcdate ) =
15 ; 0 ;
        Month ( calcdate ) = 1 and Year ( calcdate ) > 1973 and Year ( calcdate ) <
2000 and Day ( calcdate ) = 16 and DayOfWeek ( calcdate ) = 2 ; 0 ;
        Month ( calcdate ) = 1 and Year ( calcdate ) > 1999 and Int ( ( Day
( calcdate ) + 6 ) / 7 ) = 2 and DayOfWeek ( calcdate ) = 2 ; 0 ;

/*建国記念の日*/
        Month ( calcdate ) = 2 and Day ( calcdate ) = 11 and Year ( calcdate ) ≥
1966 ; 0 ;
        Month ( calcdate ) = 2 and Day ( calcdate ) = 12 and Year ( calcdate ) ≥
1974 and DayOfWeek ( calcdate ) = 2 ; 0 ;

/*昭和天皇の大喪の礼*/
Month ( calcdate ) = 2 and Day ( calcdate ) = 24 and Year ( calcdate ) = 1989 ; 0 ;

/*春分の日*/
        Month ( calcdate ) = 3 and Year ( calcdate ) < 1980 and Day ( calcdate ) =
Int ( 20.8357 + .242194* ( Year ( calcdate ) - 1980 ) - Int ( ( Year ( calcdate ) -
```

1983) / 4)) ; 0 ;
Month (calcdate) = 3 and Year (calcdate) < 1979 and Year (calcdate) > 1973 and Day (calcdate) = Int (20.8357 + .242194* (Year (calcdate) - 1980) - Int ((Year (calcdate) - 1983) / 4)) + 1 and DayOfWeek (calcdate) = 2 ; 0 ;
Month (calcdate) = 3 and Year (calcdate) < 2100 and Day (calcdate) = Int (20.8431 + .242194* (Year (calcdate) - 1980) - Int ((Year (calcdate) - 1980) / 4)) ; 0 ;
Month (calcdate) = 3 and Year (calcdate) < 2100 and Year (calcdate) > 1973 and Day (calcdate) = Int (20.8431 + .242194* (Year (calcdate) - 1980) - Int ((Year (calcdate) - 1980) / 4)) + 1 and DayOfWeek (calcdate) = 2 ; 0 ;

/*天皇誕生日 (昭和天皇)、みどりの日、昭和の日*/
Month (calcdate) = 4 and Day (calcdate) = 29 ; 0 ;
Month (calcdate) = 4 and Day (calcdate) = 30 and Year (calcdate) > 1972 and DayOfWeek (calcdate) = 2 ; 0 ;

/*メーデー*/
Month (calcdate) = 5 and Day (calcdate) = 1 and mayDay = 0 ; 0 ;

/*文化の日*/
Month (calcdate) = 5 and Day (calcdate) = 3 ; 0 ;

/*休日、みどりの日*/
Month (calcdate) = 5 and Year (calcdate) > 1984 and Day (calcdate) = 4 ; 0 ;
Month (calcdate) = 5 and Year (calcdate) > 1972 and Day (calcdate) = 4 and DayOfWeek (calcdate) = 2 ; 0 ;

/*こどもの日*/
Month (calcdate) = 5 and Day (calcdate) = 5 ; 0 ;
Month (calcdate) = 5 and Year (calcdate) > 1972 and Day (calcdate) = 6 and DayOfWeek (calcdate) = 2 ; 0 ;
Month (calcdate) = 5 and Year (calcdate) > 2005 and Day (calcdate) = 6 and DayOfWeek (calcdate) = 3 ; 0 ;
Month (calcdate) = 5 and Year (calcdate) > 2005 and Day (calcdate) = 6 and DayOfWeek (calcdate) = 4 ; 0 ;

/*皇太子徳仁親王の結婚の儀*/

Month (calcdate) = 6 and Day (calcdate) = 9 and Year (calcdate) = 1993 ; 0 ;

/*海の日*/

Month (calcdate) = 7 and Day (calcdate) = 20 and Year (calcdate) ≥ 1996 and Year (calcdate) < 2003 ; 0 ;

Month (calcdate) = 7 and Int ((Day (calcdate) + 6) / 7) = 3 and Year (calcdate) > 2002 and DayOfWeek (calcdate) = 2 ; 0 ;

/*敬老の日*/

Month (calcdate) = 9 and Year (calcdate) < 2003 and Day (calcdate) = 15 ; 0 ;

Month (calcdate) = 9 and Year (calcdate) > 1972 and Year (calcdate) < 2003 and Day (calcdate) = 16 and DayOfWeek (calcdate) = 2 ; 0 ;

Month (calcdate) = 9 and Year (calcdate) > 2002 and Int ((Day (calcdate) + 6) / 7) = 3 and DayOfWeek (calcdate) = 2 ; 0 ;

/*秋分の日*/

Month (calcdate) = 9 and Year (calcdate) < 1980 and Day (calcdate) = Int (23.2588 + .242194* (Year (calcdate) - 1980) - Int ((Year (calcdate) - 1983) / 4)) ; 0 ;

Month (calcdate) = 9 and Year (calcdate) < 1980 and Year (calcdate) > 1973 and Day (calcdate) = Int (23.2588 + .242194* (Year (calcdate) - 1980) - Int ((Year (calcdate) - 1983) / 4)) + 1 and DayOfWeek (calcdate) = 2 ; 0 ;

Month (calcdate) = 9 and Year (calcdate) < 2100 and Day (calcdate) = Int (23.2488 + .242194* (Year (calcdate) - 1980) - Int ((Year (calcdate) - 1980) / 4)) ; 0 ;

Month (calcdate) = 9 and Year (calcdate) < 2100 and Year (calcdate) > 1973 and Day (calcdate) = Int (23.2488 + .242194* (Year (calcdate) - 1980) - Int ((Year (calcdate) - 1980) / 4)) + 1 and DayOfWeek (calcdate) = 2 ; 0 ;

Month (calcdate) = 9 and Year (calcdate) < 2100 and Year (calcdate) > 1984 and Day (calcdate) = Int (23.2488 + .242194* (Year (calcdate) - 1980) - Int ((Year (calcdate) - 1980) / 4)) - 1 and DayOfWeek (calcdate) = 3 ; 0 ;

```
/*体育の日*/
        Month ( calcdate ) = 10 and Year ( calcdate ) < 2000 and Day ( calcdate ) =
10 ; 0 ;
        Month ( calcdate ) = 10 and Year ( calcdate ) > 1972 and Year ( calcdate ) <
2000 and Day ( calcdate ) = 11 and DayOfWeek ( calcdate ) = 2 ; 0 ;
        Month ( calcdate ) = 10 and Year ( calcdate ) > 1999 and Int ( ( Day
( calcdate ) + 6 ) / 7 ) = 2 and DayOfWeek ( calcdate ) = 2 ; 0 ;

/*憲法記念日*/
        Month ( calcdate ) = 11 and Day ( calcdate ) = 3 ; 0 ;
        Month ( calcdate ) = 11 and Year ( calcdate ) > 1972 and Day ( calcdate ) =
4 and DayOfWeek ( calcdate ) = 2 ; 0 ;

/*即位礼正殿の儀*/
        Month ( calcdate ) = 11 and Year ( calcdate ) = 1990 and Day ( calcdate ) =
12 ; 0 ;

/*勤労感謝の日*/
        Month ( calcdate ) = 11 and Day ( calcdate ) = 23 ; 0 ;
        Month ( calcdate ) = 11 and Year ( calcdate ) > 1972 and Day ( calcdate ) =
24 and DayOfWeek ( calcdate ) = 2 ; 0 ;

/*天皇誕生日*/
        Month ( calcdate ) = 12 and Year ( calcdate ) > 1988 and Day ( calcdate ) =
23 ; 0 ;
        Month ( calcdate ) = 12 and Year ( calcdate ) > 1988 and Day ( calcdate ) =
24 and DayOfWeek ( calcdate ) = 2 ; 0 ;

/*年末休暇*/
        /*【当院は営業日】*/Month ( calcdate ) = 12 and Day ( calcdate ) = 29 ;1 ;
        Month ( calcdate ) = 12 and Day ( calcdate ) = 30 ; 0 ;
        Month ( calcdate ) = 12 and Day ( calcdate ) = 31 ; 0 ;

/*定休日*/
        DayOfWeek ( calcdate ) = regularHoliday ; 0 ;
```

```
/*【当院】定休日2*/
        DayOfWeek ( calcdate ) = regularHoliday2 ; 0 ;

/*半ドン*/
        DayOfWeek ( calcdate ) = halfHoliday ;
        Case (
                Day ( calcdate ) ≤  7 ; halfHoliday1 ;
                Day ( calcdate ) ≤  14 ; halfHoliday2 ;
                Day ( calcdate ) ≤  21 ; halfHoliday3 ;
                Day ( calcdate ) ≤  28 ; halfHoliday4 ;
                halfHoliday5
        ) ;
        1
)
```

カスタム関数　HolidayJ

HolidayJ (N_日付)

```
// output encoding="UTF-8"
// line endings="LF(UNIX)"
// http://www.fmpro.jp
//  作者 : 社本修司
//  概要 : 指定するN_日付が日本の祝日の場合、その祝日名を返します。
//  カスタム関数  HolidayJ ( N_日付 )


/*=============================================================

   カスタム関数　HolidayJ ( N_日付 )

   CopyRight©2004,SHAMOTO Syuji
   <http://www.FMPro.jp>
```

指定されたN_日付が祝日の場合、祝日名を得ます。(2005年10月1日現在)

対象範囲
国民の祝日に関する法律(祝日法)の施行された1948(昭和23)年7月20日以降
(振替休日は1973(昭和48)年4月12日制定・施行)
春分・秋分の簡易計算の出来る範囲(1851～2150)
春分・秋分の日の計算は「新こよみの便利帳 」(恒星社厚生閣刊) による
祝日としての春分・秋分の日は閣議で決定され前年の2月の官報に公示される。

このカスタム関数を引用するに当たっては、必ずコメントも一緒に引用してください。

==========*/

```
Let (

	[   // === Let_Start ==================================

	theDate = N_日付 ;
	Y = Year ( theDate ) ;                    // theDate の年

// === HOLIDAYLIST START ============================

	HOLIDAYLIST =

	//
	// 祝日名<改行> N_日付 <改行>の形式のリスト
	// システム書式にとらわれないようにするために、( N_日付 ＊１ ) で N_日付 を
数値化
	//

	 "祝日" & "¶¶" &
	Case (
		Y ≥ 1948 and Y ≤ 2150 ;

	//
	// 元旦(1月1日)1948.07.20制定
```

```
	// New Year's Day (Gantan)
	//

		"元旦¶" &
		Case ( Y ≥ 1949 ; Date ( 1 ; 1 ; Y ) * 1 ) & "¶" &

	//
	// 成人の日(1月15日)1948.07.20制定
	// 成人の日(1月第2月曜)1998.10.21制定 2000.01.01施行
	// Coming of Age Day (Seijin-no-hi)
	//

		"成人の日¶" &
		Case (
			Y ≥ 2000;
				Date ( 1 ; 15 - DayOfWeek ( Date ( 1 ; 6 ; Y ) ) ;
Y ) ;
			Y ≥ 1949;
				Date ( 1 ; 15 ; Y )
		) * 1 & "¶" &

	//
	// 建国記念の日(2月11日)1966.6.25制定
	// National Foundation Day (Kenkoku-kinen-no-hi)
	//

		"建国記念の日¶" &
		Case ( Y ≥ 1967 ; Date ( 2 ; 11 ; Y ) * 1 ) & "¶" &

	//
	// 春分の日(春分日)1948.07.20制定
	// Vernal Equinox (Shunbun-no-hi)
	//

		"春分の日¶" &
		Case (
```

```
                    Y ≤ 1899 and Y ≥ 1851;
                        Date ( 3 ; Int ( 19.8277 + .242194 * ( Y - 1980 ) -
Int ( ( Y - 1983 ) / 4 ) ) ; Y ) ;
                    Y ≤ 1979;
                        Date ( 3 ; Int ( 20.8357 + .242194 * ( Y - 1980 ) -
Int ( ( Y - 1983 ) / 4 ) ) ; Y ) ;
                    Y ≤ 2099;
                        Date ( 3 ; Int ( 20.8431 + .242194 * ( Y - 1980 ) -
Int ( ( Y - 1980 ) / 4 ) ) ; Y ) ;
                    Y ≤ 2150;
                        Date ( 3 ; Int ( 21.851 + .242194 * ( Y - 1980 ) -
Int ( ( Y - 1980 ) / 4 ) ) ; Y )
            ) * 1 & "¶" &

        //
        // 天皇誕生日(4月29日)1948.07.20制定
        // Emperor's Birthday (Tennou-tanjyou-bi)
        //

            "天皇誕生日¶" &
            Case ( Y ≥ 1949 and Y ≤ 1988 ; Date ( 4; 29; Y ) * 1 ) & "¶" &

        //
        // みどりの日(4月29日)1989.02.17制定
        // Greenery Day (Midori-no-hi)
        //

            "みどりの日¶" &
            Case ( Y ≥ 1989 and Y ≤ 2006 ; Date ( 4 ; 29 ; Y ) * 1 ) & "¶" &

        //
        // 昭和の日(4月29日)2005.05.20制定 2007.01.01施行
        // Showa's Day (Showa-no-hi)
        //

            "昭和の日¶" &
```

```
            Case ( Y ≥ 2007 ; Date ( 4 ; 29 ; Y ) * 1 ) & "¶" &

        //
        // 憲法記念日(5月3日)1948.07.20制定
        // Constitution Memorial Day (Kenpou-kinen-bi)
        //

            "憲法記念日¶" &
            Case ( Y ≥ 1949 ; Date ( 5 ; 3 ; Y ) * 1 ) & "¶" &

        //
        // 国民の休日 1985.12.27制定
        // Holiday for a Nation (Kokumin-no-kyujitsu)

        // 国民の祝日に関する法律 第3条3項(1985.12.27施行)
        // その前日及び翌日が「国民の祝日」である日
        // (日曜日にあたる日及び前項に規定する休日にあたる日を除く。)
        // は、休日とする。
        // 「国民の休日」は通称名
        // ※振り替え休日が優先されるので国民の休日が月曜日になることはない。
        //

            "国民の休日¶" &
            Case (
                Y ≥ 1986  and Y ≤ 2006 and DayOfWeek ( Date ( 5 ; 4 ;
Y ) ) ≥ 3 ;
                    Date ( 5 ; 4 ; Y )
            ) * 1 & "¶" &

        //
        // みどりの日(5月4日)2005.05.20制定 2007.01.01施行
        // Greenery Day (Midori-no-hi)
        //

            "みどりの日¶" &
            Case ( Y ≥ 2007; Date ( 5; 4; Y ) * 1 ) & "¶" &
```

```
//
// こどもの日（5月5日）1948.07.20制定
// Children's Day (Kodomo-no-hi)
//

        "こどもの日¶" &
        Case ( Y ≥ 1949; Date ( 5; 5; Y ) * 1 ) & "¶" &

//
// 海の日（7月20日）1995.03.08制定 1996.01.01施行
// 海の日（7月第3月曜日）2001.06.22制定 2003.01.01施行
// Marine Day (Umi-no-hi)
//

        "海の日¶" &
        Case (
                Y ≥ 2003 ;
                        Date ( 7 ; 22 - DayOfWeek ( Date ( 7 ; 6 ; Y ) ) ;
Y ) ;
                Y ≥ 1996 ;
                        Date ( 7 ; 20 ; Y )
        ) * 1 & "¶" &

//
// 敬老の日（9月15日）1966.06.25制定
// 敬老の日（9月第3月曜日）2001.06.22制定 2003.01.01施行
// Respect for the Aged Day (Keirou-no-hi)
//

        "敬老の日¶" &
        Case (
                Y ≥ 2003 ;
                Date ( 9 ; 22 - DayOfWeek ( Date ( 9 ; 6 ; Y ) ) ; Y ) ;
                Y ≥ 1966 ;
                Date ( 9 ; 15 ; Y )
```

```
        ) * 1 & "¶" &

    //
    // 国民の休日
    // Holiday for a Nation (Kokumin-no-kyujitsu)
    // 敬老の日が第3月曜日に改正されたことにより派生
    // 秋分の日が水曜日になる年
    // 2009,2015,2026,2032,2037,2043,2049,2054,2060,2071,2077,
    // 2088,2094,2099,2105,2111,2116,2122,2133,2139,2150...
    //

            "国民の休日¶" &
            Case (
                Y ≥ 2003 and
                Case (
                    Y ≤ 2099 ;
                        Date ( 9 ; Int ( 23.2488 + .242194 * ( Y -
1980 ) - Int ( ( Y - 1980 ) / 4 ) ) ; Y ) ;
                    Y ≤ 2150 ;
                        Date ( 9 ; Int ( 24.2488 + .242194 * ( Y -
1980 ) - Int ( ( Y - 1980 ) / 4 ) ) ; Y )
                ) - 2 = Date ( 9 ; 22 - DayOfWeek ( Date ( 9 ; 6 ; Y ) ) ; Y ) ;
                    Case (
                        Y ≤ 2099 ;
                            Date ( 9 ; Int ( 23.2488 + .242194
* ( Y - 1980 ) - Int ( ( Y - 1980 ) / 4 ) ) ; Y ) ;
                        Y ≤ 2150 ;
                            Date ( 9 ; Int ( 24.2488 + .242194
* ( Y - 1980 ) - Int ( ( Y - 1980 ) / 4 ) ) ; Y )
                    ) -1
            ) * 1 & "¶" &

    //
    // 秋分の日（秋分日）1948.07.20制定
    // Autumnal Equinox (Shuubun-no-hi)
    //
```

```
                "秋分の日¶" &
            Case (
                    Y ≤ 1899 and Y ≥ 1851;
                        Date ( 9; Int ( 22.2588+.242194*( Y - 1980 ) - Int
( ( Y-1983 ) / 4 ) ); Y );
                    Y ≤ 1979;
                        Date ( 9; Int ( 23.2588+.242194*( Y - 1980 ) - Int
( ( Y - 1983 ) / 4 ) ); Y );
                    Y ≤ 2099;
                        Date ( 9; Int ( 23.2488+.242194*( Y - 1980 ) - Int
( ( Y - 1980 ) / 4 ) ); Y );
                    Y ≤ 2150;
                        Date ( 9; Int ( 24.2488+.242194*( Y - 1980 ) - Int
( ( Y - 1980 ) / 4 ) ); Y )
            )＊1 & "¶" &

        //
        // 体育の日(10月10日)1966.06.25制定
        // 体育の日(10月第2月曜日)1998.10.21制定 2000.01.01施行
        // Health and Sports Day (Taiku-no-hi)
        //

                "体育の日¶" &
            Case (
                    Y ≥ 2000 ;
                        Date ( 10 ; 15 - DayOfWeek ( Date ( 10 ; 6 ; Y ) ) ;
Y ) ;
                    Y ≥ 1966;
                        Date ( 10 ; 10 ; Y )
            )＊1 & "¶" &

        //
        // 文化の日(11月3日)1948.07.20制定
        // National Culture Day (Bunka-no-hi)
        //
```

```
        "文化の日¶" &
        Case ( Y ≥ 1948 ; Date ( 11 ; 3 ; Y ) + 1 ) & "¶" &

//
// 勤労感謝の日(11月23日)1948.07.20制定
// Labor Thanksgiving Day (Kinrou-kansha-no-hi)
//

        "勤労感謝の日¶" &
        Case ( Y ≥ 1948 ; Date ( 11 ; 23 ; Y ) + 1 ) & "¶" &

//
// 天皇誕生日(2月23日)2019.05制定
// Emperor's Birthday (Tennou-tanjyou-bi)
//

        "天皇誕生日¶" &
        Case ( Y ≥ 2019 ; Date ( 2 ; 23 ; Y ) + 1 ) & "¶" &

//
// 皇太子明仁親王の結婚の儀(昭和34年4月10日)
// The Rite of Wedding of His Imperial Highness Crown Prince Akihito
// (Koutaishi-AKIHITO-shinno-no-kekkonn-no-gi)
//

        "皇太子明仁親王の結婚の儀¶" &
        Case ( Y = 1959 ; Date ( 4 ; 10 ; Y ) + 1 ) & "¶" &

//
// 昭和天皇の大喪の礼(平成元年2月24日)
// The Funeral Ceremony of Emperor Showa
// (SHOWA-tennou-no-taisou-no-rei)
//

        "昭和天皇の大喪の礼¶" &
```

```
                Case ( Y = 1989 ; Date ( 2 ; 24 ; Y ) * 1 ) & "¶" &

        //
        // 即位礼正殿の儀(平成2年11月12日)
        // The Ceremony of the Enthronement of His Majesty the Emperor (at the
Seiden)
        // (Sokuirei-seiden-no-gi)
        //

                "即位礼正殿の儀¶" &
                Case ( Y = 1990 ; Date ( 11 ; 12 ; Y ) * 1 ) & "¶" &

        //
        // 皇太子徳仁親王の結婚の儀(平成5年6月9日)
        // The Rite of Wedding of His Imperial Highness Crown Prince Naruhito
        // (Koutaishi-NARUHITO-shinno-no-kekkonn-no-gi)
        //

                "皇太子徳仁親王の結婚の儀¶" &
                Case ( Y = 1993 ; Date ( 6 ; 9 ; Y ) * 1 ) & "¶"

        ) ;
// === HOLIDAYLIST END ================================

        //
        // HOLIDAYLIST 内の theDate の位置
        //

        P = Position ( HOLIDAYLIST ; ( theDate * 1 ) ; 1 ; 1 ) ;

        //
        // theDate に対する祝日名の書かれた行が HOLIDAYLIST の何行目か
        //

        LINE = PatternCount ( Left ( HOLIDAYLIST ; P ) ; "¶" )
```

```
        ];  // === Let_End ================================

        //
        // 祝日名( HOLIDAYLIST の LINE( theDate の一つ前)の行 )
        //

        Case (
                not IsEmpty ( theDate ) ;
                        Case (
                                // theDate が HOLIDAYLIST に含まれる場合
                                PatternCount ( HOLIDAYLIST ; ( theDate * 1 ) ) ;
                                        Substitute ( MiddleValues
( HOLIDAYLIST ; LINE ; 1 ) ; "¶" ; "" ) ;

                                //
                                // 2005.05.20 の法改正で
                                // 「国民の祝日」が日曜日に当たるときは、その日後
においてその日に最も近い「国民の祝日」でない日を休日とする。
                                // と振替休日の定義が変更になっている
                                //

                                // 5月3日、5月4日が日曜日の場合は5月6日は振替
休日

                                theDate ≥ Date ( 1 ; 1 ; 2007 ) and
                                theDate  = Date ( 5 ; 6 ; Y ) and
                                (
                                        DayOfWeek ( Date ( 5 ; 3 ; Y ) ) = 1 or
                                        DayOfWeek ( Date ( 5 ; 4 ; Y ) ) = 1
                                ) ;
                                        "振替休日" ;

                                //
                                // 振り替え休日(1973(昭和48)年4月12日施行)
                                // Substitute Holiday
                                //
```

```
                                        // theDate が月曜日でかつ theDate の前日が
HOLIDAYLIST に含まれる場合
                                        theDate ≥ Date ( 4 ; 12 ; 1973 ) and
                                        DayOfWeek ( theDate ) = 2 and
                                        PatternCount ( HOLIDAYLIST; ( theDate * 1 ) -
1 ) ;
                                                "振替休日"
                    )
          )
)
// ©2004-2005,SHAMOTO Syuji http://www.FMPro.jp
```

カスタム関数　gakunenrei gakunenrei_接種日

デベロッパーさんの作った関数です。Vac_PtData.fp12、Vac_YoyakuList.fp12、Vac_Kojin.fp12で利用させてもらってます。

gakunenrei (生年月日)

```
Let([
    誕生学年=Year(生年月日+274)-1;
    本日学年=Year(Get(日付)+274)-1;
    学年齢=本日学年 - 誕生学年
  ];
    学年齢
  )
```

上記利用して

gakunenrei_接種日 (生年月日 ; 接種日)

```
Let([
    誕生学年=Year(生年月日+274)-1;
```

```
    本日学年=Year(接種日+274)-1;
    学年齢=本日学年 - 誕生学年
  ];
    学年齢
  )
```

カスタム関数　nendorei　nendorei_接種日

デベロッパーさんの作った関数です。Vac_PtData.fp12、Vac_YoyakuList.fp12、Vac_Kojin.fp12で利用させてもらってます。

nendorei（ 生年月日 ）

```
Let([
    誕生年度=Year(生年月日+275)-1;
    本日年度=Year(Get(日付)+275)-1;
    年度年齢=本日年度 - 誕生年度
  ];
    年度年齢
  )
```

上記利用して

nendorei_接種日（ 生年月日 ； 接種日 ）

```
Let([nendorei_接種日（ 生年月日 ； 接種日 ）
    誕生年度=Year(生年月日+275)-1;
    本日年度=Year(接種日+275)-1;
    年度年齢=本日年度 - 誕生年度
  ];
    年度年齢
  )
```

カスタム関数　IsValidEmail

デベロッパーさんの作った関数です。Vac_PtData.fp12、Vac_WD_public.fp12、
Vac_WD_private.fp12で利用させてもらってます。

IsValidEmail (メールアドレス ; オプション)

```
// output encoding="UTF-8"
// line endings="LF(UNIX)"
// http://www.fmpro.jp
//  作者 : 社本修司
//  概要 : ＜メールアドレス＞が妥当な場合に真「1」を返します。妥当ではない場合、偽「0」を返します。
//  カスタム関数  IsValidEmail ( メールアドレス ; オプション )


/*=============================================================
   カスタム関数  IsValidEmail ( メールアドレス ; オプション )

＜メールアドレス＞が妥当な場合に真「1」を返します。妥当ではない場合、偽「0」を返します。
オプションに「1」を指定することで偽の場合にその理由を返し、真の場合はnull値を返します。
（オプション「0」の場合の結果は「1」か「0」）

# トップレベルドメインは2006年4月現在申請中も含む
<http://www.nic.ad.jp/ja/dom/types.html> 参照
# ドメイン名のルールは JPNIC の「ドメイン名のしくみ」を参照
<http://www.nic.ad.jp/ja/dom/system.html>
# ドメイン名のラベルは第6レベルまで判定
# ローカルパートで利用出来る文字列は RFC2822 3.2.4. Atom の atext による
# 日本語訳 <http://www.puni.net/~mimori/rfc/rfc2822.txt> 参照
# ()によるコメントや、""によるローカルパート表記、日本語ドメインを含む国際化ドメイン名は考慮していません
```

```
  このカスタム関数を引用するに当たっては、必ずコメントも一緒に引用してください。
==========================================================*/

Let (
        [
        ADDR_SPEC = Lower ( メールアドレス ) ;
        OPTION = オプション ;

        TLD_LIST =
                "com¶net¶org¶edu¶gov¶mil¶int¶info¶biz¶name¶pro¶museum¶aero¶
coop¶jobs¶travel¶mobi¶cat¶post¶tel¶xxx¶asia¶arpa¶ac¶ad¶ae¶af¶ag¶ai¶al¶am¶an¶
ao¶aq¶ar¶as¶at¶au¶aw¶ax¶az¶ba¶bb¶bd¶be¶bf¶bg¶bh¶bi¶bj¶bm¶bn¶bo¶br¶bs¶bt¶
bv¶bw¶by¶bz¶ca¶cc¶cd¶cf¶cg¶ch¶ci¶ck¶cl¶cm¶cn¶co¶cr¶cs¶cu¶cv¶cx¶cy¶cz¶de¶
dj¶dk¶dm¶do¶dz¶ec¶ee¶eg¶eh¶er¶es¶et¶eu¶fi¶fj¶fk¶fm¶fo¶fr¶ga¶gb¶gd¶ge¶gf¶gg¶
gh¶gi¶gl¶gm¶gn¶gp¶gq¶gr¶gs¶gt¶gu¶gw¶gy¶hk¶hm¶hn¶hr¶ht¶hu¶id¶ie¶il¶im¶in¶io¶iq
¶ir¶is¶it¶je¶jm¶jo¶jp¶ke¶kg¶kh¶ki¶km¶kn¶kp¶kr¶kw¶ky¶kz¶la¶lb¶lc¶li¶lk¶lr¶ls¶lt¶lu¶
lv¶ly¶ma¶mc¶md¶mg¶mh¶mk¶ml¶mm¶mn¶mo¶mp¶mq¶mr¶ms¶mt¶mu¶mv¶mw¶mx¶
my¶mz¶na¶nc¶ne¶nf¶ng¶ni¶nl¶no¶np¶nr¶nu¶nz¶om¶pa¶pe¶pf¶pg¶ph¶pk¶pl¶pm¶pn¶
pr¶ps¶pt¶pw¶py¶qa¶re¶ro¶ru¶rw¶sa¶sb¶sc¶sd¶se¶sg¶sh¶si¶sj¶sk¶sl¶sm¶sn¶so¶sr¶
st¶sv¶sy¶sz¶tc¶td¶tf¶tg¶th¶tj¶tk¶tl¶tm¶tn¶to¶tp¶tr¶tt¶tv¶tw¶tz¶ua¶ug¶uk¶um¶us¶
uy¶uz¶va¶vc¶ve¶vg¶vi¶vn¶vu¶wf¶ws¶ye¶yt¶yu¶za¶zm¶zw¶su¶" ;
                // トップレベルドメインのリスト
<http://www.nic.ad.jp/ja/dom/types.html> 参照
                // su 旧ソビエト連邦 wikipediaによると未だに使われているらしい
        JP_LIST = "ac¶co¶go¶or¶ad¶ne¶gr¶ed¶lg¶" ;       // 属性型(組織種別型)
JPドメイン名の属性リスト

        P_SEPARATOR = Position ( ADDR_SPEC ; "@" ; 1 ; PatternCount
( ADDR_SPEC ; "@" ) ) ;             // @が2回以上書かれた場合を考慮して最後の@をド
メインとの区切りとする
        LOCAL_PART = Left ( ADDR_SPEC ; P_SEPARATOR - 1 ) ;          // ローカ
ルパート(アカウント名)
        DOMAIN = Right ( ADDR_SPEC ; Length ( ADDR_SPEC ) -
P_SEPARATOR ) ;          // ドメイン名

        P_LAST_DOT = Position ( DOMAIN ; "." ; 1 ; PatternCount ( DOMAIN ;
```

```
"." ) ) ;
        P_LAVEL2_DOT = Position ( DOMAIN ; "." ; 1 ; PatternCount ( DOMAIN ;
"." ) - 1 ) ;
        P_LAVEL3_DOT = Position ( DOMAIN ; "." ; 1 ; PatternCount ( DOMAIN ;
"." ) - 2 ) ;
        P_LAVEL4_DOT = Position ( DOMAIN ; "." ; 1 ; PatternCount ( DOMAIN ;
"." ) - 3 ) ;
        P_LAVEL5_DOT = Position ( DOMAIN ; "." ; 1 ; PatternCount ( DOMAIN ;
"." ) - 4 ) ;
        P_LAVEL6_DOT = Position ( DOMAIN ; "." ; 1 ; PatternCount ( DOMAIN ;
"." ) - 5 ) ;

        TOP_LEVEL_DOMAIN = Middle ( DOMAIN ; P_LAST_DOT + 1 ; Length
( DOMAIN ) - P_LAST_DOT ) ;    // トップレベルドメイン
        LAVEL2 = Middle ( DOMAIN ; P_LAVEL2_DOT + 1 ; P_LAST_DOT -
P_LAVEL2_DOT - 1 ) ;    // 第2レベルドメイン
        LAVEL3 = Middle ( DOMAIN ; P_LAVEL3_DOT + 1 ; P_LAVEL2_DOT -
P_LAVEL3_DOT - 1 ) ;    // 第3レベルドメイン
        LAVEL4 = Middle ( DOMAIN ; P_LAVEL4_DOT + 1 ; P_LAVEL3_DOT -
P_LAVEL4_DOT - 1 ) ;    // 第4レベルドメイン
        LAVEL5 = Middle ( DOMAIN ; P_LAVEL5_DOT + 1 ; P_LAVEL4_DOT -
P_LAVEL5_DOT - 1 ) ;    // 第5レベルドメイン
        LAVEL6 = Middle ( DOMAIN ; P_LAVEL6_DOT + 1 ; P_LAVEL5_DOT -
P_LAVEL6_DOT - 1 ) ;    // 第6レベルドメイン

        DOCOMO = Exact ( DOMAIN ; "docomo.ne.jp" ) or Exact ( DOMAIN ;
"ezweb.ne.jp" ) or PatternCount ( DOMAIN ; ".ezweb.ne.jp" ) ;       // 携帯アドレス
特別ルール判定用
        JP = Exact ( TOP_LEVEL_DOMAIN ; "jp" ) and Length ( LAVEL2 ) ≤ 2 ;
        // 属性型(組織種別型)JPドメイン判定用
        JO = Exact ( TOP_LEVEL_DOMAIN ; "jo" ) and Length ( LAVEL2 ) ≤ 2 ;
        // JP のtypo対策(ヨルダンドメインの属性は3文字)

        FILTERE_LOCAL_PART =
                Filter (
                        LOCAL_PART ;
```

```
                    "abcdefghijklmnopqrstuvwxyz" &
                    "ABCDEFGHIJKLMNOPQRSTUVWXYZ" &
                    "0123456789" &
                    "!#$%&'*+-/=?^_`{|}~" &
                    "."
            );
        FILTERE_DOMAIN =
            Filter (
                    DOMAIN ;
                    "abcdefghijklmnopqrstuvwxyz" &
                    "ABCDEFGHIJKLMNOPQRSTUVWXYZ" &
                    "0123456789-."
            );

        RESULT =
            Case (
                    IsEmpty ( ADDR_SPEC ) ;
                        "メールアドレスが空です" ;

                    PatternCount ( ADDR_SPEC ; "@" ) = 0 or
                    IsEmpty ( DOMAIN ) ;
                        "ドメイン(@の右側)が空です" ;
                    PatternCount ( DOMAIN ; "." ) = 0 ;
                        "無効なドメイン(@の右側)です" ;
                    Length ( DOMAIN ) > 255 ;
                        "ドメイン(@の右側)が長すぎます" ;          // ドメイ
ン名全体の長さは255文字以下
                    Length ( LAVEL2 ) > 63 or
                    Length ( LAVEL3 ) > 63 or
                    Length ( LAVEL4 ) > 63 or
                    Length ( LAVEL5 ) > 63 or
                    Length ( LAVEL6 ) > 63 ;
                        "無効なドメイン(@の右側)です" ; // 各ラベルの長さ
は63文字以下
                    Left ( DOMAIN ; 1 ) = "-" ;
                        "ドメインの最初を¥"-¥"(ハイフン)にすることは出来ま
```

せん″ ;

Left (LAVEL2 ; 1) = ″−″ or Right (LAVEL2 ; 1) = ″−″ or
Left (LAVEL3 ; 1) = ″−″ or Right (LAVEL3 ; 1) = ″−″ or
Left (LAVEL4 ; 1) = ″−″ or Right (LAVEL4 ; 1) = ″−″ or
Left (LAVEL5 ; 1) = ″−″ or Right (LAVEL5 ; 1) = ″−″ or
Left (LAVEL6 ; 1) = ″−″ or Right (LAVEL6 ; 1) = ″−″ ;

″無効なドメインです(¥″.¥″(ドット)の直前または直後を¥″−¥″(ハイフン)にすることは出来ません)″ ; // ラベルの先頭と最後にハイフンは使用できない

not Exact (DOMAIN ; FILTERE_DOMAIN) and PatternCount (DOMAIN ; ″¶″) ;

″ドメイン(@の右側)に無効な文字列(改行)が含まれています″ ;

not Exact (DOMAIN ; FILTERE_DOMAIN) and (PatternCount (DOMAIN ; ″ ″) or PatternCount (DOMAIN ; ″ ″)) ;

″ドメイン(@の右側)に無効な文字列(¥″ ¥″スペース)が含まれています″ ;

not Exact (DOMAIN ; FILTERE_DOMAIN) ;

″ドメイン(@の右側)に無効な文字列が含まれています″ ;

Left (DOMAIN ; 1) = ″.″ ;

″ドメイン(@の右側)の最初を¥″.¥″(ドット)にすることは出来ません″ ;

Right (DOMAIN ; 1) = ″.″ ;

″ドメイン(@の右側)の最後を¥″.¥″(ドット)にすることは出来ません″ ;

PatternCount (DOMAIN ; ″..″) ≥ 1 ;

″ドメイン(@の右側)に¥″..¥″を含める(ドットを二つ続ける)ことは出来ません″ ;

IsEmpty (FilterValues (TOP_LEVEL_DOMAIN ; TLD_LIST)) ;

″トップレベルドメイン(最後の¥″.jp¥″など)が無効です″ ;

JP and IsEmpty (FilterValues (LAVEL2 ; JP_LIST)) ;

″ドメインの属性(¥″.co¥″や¥″.ne¥″など)が無効です″ ; // 属性型(組織種別型)JPドメインの場合は属性もチェック

```
                    JO and not IsEmpty ( FilterValues ( LAVEL2 ; JP_LIST ) ) ;
                        "ヨルダン(Jordan)ドメインの属性は
¥".com, .org, .edu, .mil, .net, .gov¥" のいずれかです" ;

                    not Exact ( LOCAL_PART ; FILTERE_LOCAL_PART ) and
( PatternCount ( LOCAL_PART ; " " ) or PatternCount ( LOCAL_PART ; "  " ) ) ;
                        "アカウント(@の左側)に無効な文字列(¥" ¥"スペー
ス)が含まれています" ;
                    not Exact ( LOCAL_PART ; FILTERE_LOCAL_PART ) ;
                        "アカウント(@の左側)に無効な文字列が含まれていま
す" ;
                    Left ( LOCAL_PART ; 1 ) = "." ;
                        "アカウントの最初を¥".¥"(ドット)にすることは出来ませ
ん" ;

                    not DOCOMO and Right ( LOCAL_PART ; 1 ) = "." ;
                        "アカウント(@の左側)の最後を¥".¥"(ドット)にすること
は出来ません" ;
                    not DOCOMO and PatternCount ( LOCAL_PART ; ".." ) ≥
1 ;
                        "アカウント(@の左側)に¥"..¥"を含める(ドットを二つ続
ける)ことは出来ません" ;
                        // docomo.ne.jp ないしは ezweb.ne.jp の場合、.の
扱いが RFC2822 で認められていない使い方ができるため
                )
        ] ;

        Case (
                OPTION ;
                    RESULT ;
                    IsEmpty ( RESULT )
        )

)
// ©2006,SHAMOTO Syuji http://www.FMPro.jp
```

カスタム関数　FilterEmail

デベロッパーさんの作った関数です。Vac_PtData.fp12、Vac_Mail_SyoMe.fp12で利用させてもらってます。

FilterEmail（ フィルタするメールアドレス ）

```
// output encoding="UTF-8"
// line endings="LF(UNIX)"
// http://www.fmpro.jp
//  作者 : 社本修司
//  概要 : ＜フィルタするメールアドレス＞からメールアドレスで使用できる文字のみを返します。
//  カスタム関数  FilterEmail（ フィルタするメールアドレス ）


/*
FilterEmail（ フィルタするメールアドレス ）
＜フィルタするメールアドレス＞からメールアドレスで使用できる文字のみを返します。
*/

Filter（
        フィルタするメールアドレス ;
        "abcdefghijklmnopqrstuvwxyz" &
        "ABCDEFGHIJKLMNOPQRSTUVWXYZ" &
        "0123456789" &
        "!#$%&'*+-/=?^_`{|}~" &
        ".@"
）
```

カスタム関数　Pref

デベロッパーさんの作った関数です。Vac_PtData.fp12で利用させてもらってます。

Pref (住所)

```
// output encoding="UTF-8"
// line endings="LF(UNIX)"
// http://www.fmpro.jp
// 作者 : 社本修司
// 概要 : 住所から都道府県名を返します。
// カスタム関数 Pref ( 住所 )


/*=============================================================

   カスタム関数　Pref ( 住所 )

   CopyRight©2004-2006,SHAMOTO Syuji
   <http://www.FMPro.jp>

  住所から都道府県名を返します。

  このカスタム関数を引用するに当たっては、必ずコメントも一緒に引用してください。

=============================================================*/

Case (
        PatternCount ( "北海道 青森県 岩手県 宮城県 秋田県 山形県 福島県 茨城県 栃木県 群馬県 埼玉県 千葉県 東京都 新潟県 富山県 石川県 福井県 山梨県 長野県 岐阜県 静岡県 愛知県 " & "三重県 滋賀県 京都府 大阪府 兵庫県 奈良県 鳥取県 島根県 岡山県 広島県 山口県 徳島県 香川県 愛媛県 高知県 福岡県 佐賀県 長崎県 熊本県 大分県 宮崎県 沖縄県 " ; Left ( 住所 ; 3 ) ) = 1;
                Left ( 住所 ; 3 ) ;

        PatternCount ( "神奈川県 和歌山県 鹿児島県 " ; Left ( 住所 ; 4 ) ) = 1 ;
                Left ( 住所 ; 4 )
```

)

カスタム関数　City

デベロッパーさんの作った関数です。Vac_PtData.fp12で利用させてもらってます。

City (住所)

```
// output encoding="UTF-8"
// line endings="LF(UNIX)"
// http://www.fmpro.jp
//  作者 : 社本修司
//  概要 : 住所から市区郡町村名を返します。
//  カスタム関数  City ( 住所 )


/*=============================================================

  カスタム関数　　City ( 住所 )

    CopyRight©2004-2006,SHAMOTO Syuji
    <http://www.FMPro.jp>

  住所から市区郡町村名を返します。
  町村の場合は、郡を含む町または村まで。政令指定都市の場合は区までを市区郡町村名とします。
  2006.04.01  現在
=============================================================*/

Let (
          [
          PREF = Pref ( 住所 ) ;
          ADDRESS = Substitute ( 住所 ; PREF ; "" ) ;
          C = Position ( ADDRESS ; "市" ; 1 ; 1 ) ;
```

```
T = Position ( ADDRESS ; "町" ; 1 ; 1 ) ;
V = Position ( ADDRESS ; "村" ; 1 ; 1 ) ;
K = Position ( ADDRESS ; "区" ; 1 ; 1 ) ;
G = Position ( ADDRESS ; "郡" ; 1 ; 1 )
] ;

Case (
// 市名に「市」が含まれる
        (
                PatternCount ( 住所 ; "市川市" ) and
                (
                        Middle ( 住所 ; Position ( 住所 ; "市川市" ; 1 ;
1 ) - 1 ; 1 ) = "県" or
                        Position ( 住所 ; "市川市" ; 1 ; 1 ) = 1
                )
        ) or    // ○○市川市町という形で町名が川市ということもあるので場
所も判断
        (
                PatternCount ( 住所 ; "市原市" ) and
                (
                        Middle ( 住所 ; Position ( 住所 ; "市原市" ; 1 ;
1 ) - 1 ; 1 ) = "県" or
                        Position ( 住所 ; "市原市" ; 1 ; 1 ) = 1
                )
        ) or    // ○○市原市町という形で町名が原市ということもあるので場
所も判断
        PatternCount ( 住所 ; "今市市" ) or     // (現・日光市)
        PatternCount ( 住所 ; "八日市場市" ) or         // (現・匝瑳市)
        PatternCount ( 住所 ; "四日市市" ) or
        PatternCount ( 住所 ; "八日市市" ) or
        PatternCount ( 住所 ; "廿日市市" ) ;

                Left (
                        ADDRESS ;
                        Position ( ADDRESS ; "市" ; 1 ; 2 )
                ) ;
```

```
		// 町名に「市」が含まれる（旧住所の地名を含め他府県にも同じ町名が存在する場合は都道府県も考慮する）
		// 町名に「村」が含まれる
		PatternCount ( 住所 ; "余市郡余市町" ) or		// 北海道余市郡余市町
		PatternCount ( 住所 ; "北海道余市町" ) or
		(
			PatternCount ( 住所 ; "余市町" ) and
			Position ( 住所 ; "余市町" ; 1 ; 1 ) = 1
		) or
		PatternCount ( 住所 ; "九戸郡種市町" ) or		// 岩手県九戸郡種市町
		PatternCount ( 住所 ; "岩手県種市町" ) or
		(
			PatternCount ( 住所 ; "種市町" ) and
			Position ( 住所 ; "種市町" ; 1 ; 1 ) = 1
		) or
		PatternCount ( 住所 ; "芳賀郡市貝町" ) or		// 栃木県芳賀郡市貝町
		PatternCount ( 住所 ; "栃木県市貝町" ) or
		(
			PatternCount ( 住所 ; "市貝町" ) and
			Position ( 住所 ; "市貝町" ; 1 ; 1 ) = 1
		) or
		PatternCount ( 住所 ; "中新川郡上市町" ) or	// 富山県中新川郡上市町
		PatternCount ( 住所 ; "富山県上市町" ) or
		(
			PatternCount ( 住所 ; "上市町" ) and
			Position ( 住所 ; "上市町" ; 1 ; 1 ) = 1
		) or
		PatternCount ( 住所 ; "石川郡野々市町" ) or	// 石川県石川郡野々市町
		PatternCount ( 住所 ; "石川県野々市町" ) or
		(
```

```
                PatternCount ( 住所 ; "野々市町" ) and
                Position ( 住所 ; "野々市町" ; 1 ; 1 ) = 1
        ) or
        PatternCount ( 住所 ; "西八代郡市川大門町" ) or        // 山梨県西八代郡市川大門町(現・市川三郷町)
        PatternCount ( 住所 ; "山梨県市川大門町" ) or
        (
                PatternCount ( 住所 ; "市川大門町" ) and
                Position ( 住所 ; "市川大門町" ; 1 ; 1 ) = 1
        ) or
        PatternCount ( 住所 ; "西八代郡市川三郷町" ) or        // 山梨県西八代郡市川三郷町
        PatternCount ( 住所 ; "山梨県市川三郷町" ) or
        (
                PatternCount ( 住所 ; "市川三郷町" ) and
                Position ( 住所 ; "市川三郷町" ; 1 ; 1 ) = 1
        ) or
        PatternCount ( 住所 ; "芦品郡新市町" ) or       // 広島県芦品郡新市町(現・福山市)
        PatternCount ( 住所 ; "広島県新市町" ) or
        (
                PatternCount ( 住所 ; "新市町" ) and
                Position ( 住所 ; "新市町" ; 1 ; 1 ) = 1
        ) or
        PatternCount ( 住所 ; "香美郡野市町" ) or       // 高知県香美郡野市町
        PatternCount ( 住所 ; "高知県野市町" ) or
        (
                PatternCount ( 住所 ; "野市町" ) and
                Position ( 住所 ; "野市町" ; 1 ; 1 ) = 1
        ) or
        PatternCount ( 住所 ; "日置郡東市来町" ) or     // 鹿児島県日置郡東市来町(現・日置市)
        PatternCount ( 住所 ; "鹿児島県東市来町" ) or
        (
                PatternCount ( 住所 ; "東市来町" ) and
```

```
                Position ( 住所 ; "東市来町" ; 1 ; 1 ) = 1
        ) or
        PatternCount ( 住所 ; "日置郡市来町" ) or       // 鹿児島県日置郡市来町(現・いちき串木野市)
        PatternCount ( 住所 ; "鹿児島県市来町" ) or
        (
                PatternCount ( 住所 ; "市来町" ) and
                Position ( 住所 ; "市来町" ; 1 ; 1 ) = 1
        ) or
        PatternCount ( 住所 ; "吉野郡下市町" ) or       // 奈良県吉野郡下市町
        PatternCount ( 住所 ; "奈良県下市町" ) or
        (
                PatternCount ( 住所 ; "下市町" ) and
                Position ( 住所 ; "下市町" ; 1 ; 1 ) = 1
        ) or
        PatternCount ( 住所 ; "神崎郡市川町" ) or       // 兵庫県神崎郡市川町
        PatternCount ( 住所 ; "兵庫県市川町" ) or
        (
                PatternCount ( 住所 ; "市川町" ) and
                Position ( 住所 ; "市川町" ; 1 ; 1 ) = 1
        ) or
        PatternCount ( 住所 ; "氷上郡市島町" ) or       // 兵庫県氷上郡市島町(現・丹波市)
        PatternCount ( 住所 ; "兵庫県市島町" ) or
        (
                PatternCount ( 住所 ; "上市町" ) and
                Position ( 住所 ; "上市町" ; 1 ; 1 ) = 1
        ) or
        PatternCount ( 住所 ; "鹿足郡六日市町" ) or     // 島根県鹿足郡六日市町(現・吉賀町)
        PatternCount ( 住所 ; "島根県六日市町" ) or
        (
                PatternCount ( 住所 ; "六日市町" ) and
                Position ( 住所 ; "六日市町" ; 1 ; 1 ) = 1
```

```
        ) or
        PatternCount ( 住所 ; "阿波郡市場町" ) or      // 徳島県阿波郡市場町(現・阿波市)
        PatternCount ( 住所 ; "徳島県市場町" ) or
        (
            PatternCount ( 住所 ; "市場町" ) and
            Position ( 住所 ; "市場町" ; 1 ; 1 ) = 1
        ) or
        PatternCount ( 住所 ; "香美郡野市町" ) or      // 高知県香美郡野市町
        PatternCount ( 住所 ; "高知県野市町" ) or
        (
            PatternCount ( 住所 ; "野市町" ) and
            Position ( 住所 ; "野市町" ; 1 ; 1 ) = 1
        ) or
        PatternCount ( 住所 ; "西多摩郡五日市町" ) or          // 東京都西多摩郡五日市町(現・あきる野市)
        PatternCount ( 住所 ; "東京都五日市町" ) or
        (
            PatternCount ( 住所 ; "五日市町" ) and
            Position ( 住所 ; "五日市町" ; 1 ; 1 ) = 1
        ) or
        PatternCount ( 住所 ; "佐伯郡廿日市町" ) or    // 広島県佐伯郡廿日市町(現・廿日市市)
        PatternCount ( 住所 ; "広島県廿日市町" ) or
        (
            PatternCount ( 住所 ; "廿日市町" ) and
            Position ( 住所 ; "廿日市町" ; 1 ; 1 ) = 1
        ) or


        PatternCount ( 住所 ; "柴田郡村田町" ) or      // 宮城県柴田郡村田町
        PatternCount ( 住所 ; "宮城県村田町" ) or
        (
            PatternCount ( 住所 ; "村田町" ) and
```

```
                Position ( 住所 ; "村田町" ; 1 ; 1 ) = 1
        ) or
        PatternCount ( 住所 ; "佐波郡玉村町" ) or       // 群馬県佐波郡玉村町
        PatternCount ( 住所 ; "群馬県玉村町" ) or
        (
                PatternCount ( 住所 ; "玉村町" ) and
                Position ( 住所 ; "玉村町" ; 1 ; 1 ) = 1
        ) or
        PatternCount ( 住所 ; "中蒲原郡村松町" ) or    // 新潟県中蒲原郡村松町
        PatternCount ( 住所 ; "新潟県村松町" ) or
        (
                PatternCount ( 住所 ; "村松町" ) and
                Position ( 住所 ; "村松町" ; 1 ; 1 ) = 1
        ) or
        PatternCount ( 住所 ; "恵那郡岩村町" ) or       // 岐阜県恵那郡岩村町(現・恵那市)
        PatternCount ( 住所 ; "岐阜県岩村町" ) or
        (
                PatternCount ( 住所 ; "岩村町" ) and
                Position ( 住所 ; "岩村町" ; 1 ; 1 ) = 1
        ) or
        PatternCount ( 住所 ; "美方郡村岡町" ) or       // 兵庫県美方郡村岡町(現・美方郡香美町)
        PatternCount ( 住所 ; "兵庫県村岡町" ) or
        (
                PatternCount ( 住所 ; "村岡町" ) and
                Position ( 住所 ; "村岡町" ; 1 ; 1 ) = 1
        ) or
        PatternCount ( 住所 ; "東宇和郡野村町" ) or    // 愛媛県東宇和郡野村町(現・西予市)
        PatternCount ( 住所 ; "愛媛県野村町" ) or
        (
                PatternCount ( 住所 ; "野村町" ) and
                Position ( 住所 ; "野村町" ; 1 ; 1 ) = 1
```

```
        ) or

        PatternCount ( 住所 ; "富良野町" ) ;    //

                Left ( ADDRESS ; T ) ;

        // 村名に「市」が含まれる
        PatternCount ( 住所 ; "北津軽郡市浦村" ) or    // 青森県北津
軽郡市浦村(現・五所川原市)
        PatternCount ( 住所 ; "青森県市浦村" ) or
        (
                PatternCount ( 住所 ; "市浦村" ) and
                Position ( 住所 ; "市浦村" ; 1 ; 1 ) = 1
        ) ;

                Left ( ADDRESS ; V ) ;

        // 町名に「町」が含まれる
        PatternCount ( 住所 ; "杵島郡大町町" ) or      // 佐賀県杵島
郡大町町
        PatternCount ( 住所 ; "佐賀県大町町" ) or
        (
                PatternCount ( 住所 ; "大町町" ) and
                Position ( 住所 ; "大町町" ; 1 ; 1 ) = 1
        ) or
        PatternCount ( 住所 ; "北松浦郡鹿町町" ) or    // 長崎県北松
浦郡鹿町町
        PatternCount ( 住所 ; "長崎県鹿町町" ) or
        (
                PatternCount ( 住所 ; "鹿町町" ) and
                Position ( 住所 ; "鹿町町" ; 1 ; 1 ) = 1
        ) ;

                Left (
                        ADDRESS ;
                        Position ( ADDRESS ; "町" ; 1 ; 2 )
```

```
        ) ;

        // 郡名に「市」が含まれる町（北海道余市郡・奈良県高市郡）
        PatternCount ( 住所 ; "余市郡仁木町" ) or       // 北海道余市郡仁木町
        PatternCount ( 住所 ; "高市郡高取町" ) ;        // 奈良県高市郡高取町

            Left ( ADDRESS ; T ) ;

        // 郡名に「市」が含まれる村（北海道余市郡・奈良県高市郡）
        PatternCount ( 住所 ; "余市郡赤井川村" ) or     // 北海道余市郡赤井川村
        PatternCount ( 住所 ; "高市郡明日香村" ) ;      // 奈良県高市郡明日香村

            Left ( ADDRESS ; V ) ;

        // 郡名に「村」が含まれる村（福島県田村郡）
        PatternCount ( 住所 ; "田村郡都路村" ) ;

            Left (
                ADDRESS ;
                Position ( ADDRESS ; "村" ; 1 ; 2 )
            ) ;

        // 郡名に「村」が含まれる町（山形県東村山郡・山形県西村山郡・山形県北村山郡・福島県田村郡）
        PatternCount ( 住所 ; "東村山郡" ) or
        PatternCount ( 住所 ; "西村山郡" ) or
        PatternCount ( 住所 ; "北村山郡" ) or
        PatternCount ( 住所 ; "田村郡" ) ;

            Left ( ADDRESS ; T ) ;
```

```
                // 市を優先させるもの
                PatternCount ( 住所 ; "村山市" ) or      // 山形県
                PatternCount ( 住所 ; "原町市" ) or      // 福島県(現・南相馬
市)
                PatternCount ( 住所 ; "田村市" ) or      // 福島県
                PatternCount ( 住所 ; "町田市" ) or      // 東京都
                PatternCount ( 住所 ; "東村山市" ) or    // 東京都
                PatternCount ( 住所 ; "武蔵村山市" ) or          // 東京都
                PatternCount ( 住所 ; "羽村市" ) or      // 東京都
                PatternCount ( 住所 ; "十日町市" ) or    // 新潟県
                PatternCount ( 住所 ; "村上市" ) or      // 新潟県
                PatternCount ( 住所 ; "大町市" ) or      // 長野県
                PatternCount ( 住所 ; "中村市" ) or      // 高知県(現・四万十
市)
                PatternCount ( 住所 ; "大村市" ) or      // 長崎県
                PatternCount ( 住所 ; "郡山市" ) or      // 福島県
                PatternCount ( 住所 ; "郡上市" ) or      // 岐阜県
                PatternCount ( 住所 ; "蒲郡市" ) or      // 愛知県
                PatternCount ( 住所 ; "大和郡山市" ) or          // 奈良県
                PatternCount ( 住所 ; "小郡市" ) or      // 福岡県
                PatternCount ( 住所 ; "向日町市" ) ; // 京都府向日市の誤り

                        Left ( ADDRESS ; C ) ;


                // 区を優先させるもの
                ( PatternCount ( 住所 ; "札幌市" ) and PatternCount ( 住所 ; "区
" ) ) or
                ( PatternCount ( 住所 ; "仙台市" ) and PatternCount ( 住所 ; "区
" ) ) or
                ( PatternCount ( 住所 ; "横浜市" ) and PatternCount ( 住所 ; "区
" ) ) or
                ( PatternCount ( 住所 ; "川崎市" ) and PatternCount ( 住所 ; "区
" ) ) or
                ( PatternCount ( 住所 ; "名古屋市" ) and PatternCount ( 住所 ; "
区" ) ) or
```

```
                ( PatternCount ( 住所 ; "中村区" ) ) or  // 村・町のつく区
                ( PatternCount ( 住所 ; "京都市" ) and PatternCount ( 住所 ; "区
" ) ) or
                ( PatternCount ( 住所 ; "大阪市" ) and PatternCount ( 住所 ; "区
" ) ) or
                ( PatternCount ( 住所 ; "神戸市" ) and PatternCount ( 住所 ; "区
" ) ) or
                ( PatternCount ( 住所 ; "広島市" ) and PatternCount ( 住所 ; "区
" ) ) or
                ( PatternCount ( 住所 ; "北九州市" ) and PatternCount ( 住所 ; "
区" ) ) or
                ( PatternCount ( 住所 ; "福岡市" ) and PatternCount ( 住所 ; "区
" ) ) or
                ( PatternCount ( 住所 ; "千葉市" ) and PatternCount ( 住所 ; "区
" ) ) or
                ( PatternCount ( 住所 ; "さいたま市" ) and PatternCount ( 住所 ; "
区" ) ) or
                ( PatternCount ( 住所 ; "静岡市" ) and PatternCount ( 住所 ; "区
" ) ) or
                ( PatternCount ( 住所 ; "堺市" ) and PatternCount ( 住所 ; "区
" ) ) or          // 2006.04.01
                ( PatternCount ( 住所 ; "新潟市" ) and PatternCount ( 住所 ; "区
" ) ) or          // 2007.04.01
                ( PatternCount ( 住所 ; "浜松市" ) and PatternCount ( 住所 ; "区
" ) ) or          // 2007.04.01

                ( PREF = "東京都" and PatternCount ( 住所 ; "区" ) ) or
                Left ( 住所 ; 4 ) = "千代田区" or
                Left ( 住所 ; 3 ) = "中央区" or
                Left ( 住所 ; 2 ) = "港区" or
                Left ( 住所 ; 3 ) = "新宿区" or
                Left ( 住所 ; 3 ) = "文京区" or
                Left ( 住所 ; 3 ) = "台東区" or
                Left ( 住所 ; 3 ) = "墨田区" or
                Left ( 住所 ; 3 ) = "江東区" or
                Left ( 住所 ; 3 ) = "品川区" or
```

```
Left ( 住所 ; 3 ) = "目黒区" or
Left ( 住所 ; 3 ) = "大田区" or
Left ( 住所 ; 4 ) = "世田谷区" or
Left ( 住所 ; 3 ) = "渋谷区" or
Left ( 住所 ; 3 ) = "中野区" or
Left ( 住所 ; 3 ) = "杉並区" or
Left ( 住所 ; 3 ) = "豊島区" or
Left ( 住所 ; 2 ) = "北区" or
Left ( 住所 ; 3 ) = "荒川区" or
Left ( 住所 ; 3 ) = "板橋区" or
Left ( 住所 ; 3 ) = "練馬区" or
Left ( 住所 ; 3 ) = "足立区" or
Left ( 住所 ; 3 ) = "葛飾区" or
Left ( 住所 ; 4 ) = "江戸川区" ;

        Left ( ADDRESS ; K ) ;

// 市の後に町がくる場合、市を優先
PatternCount ( 住所 ; "市" ) and
PatternCount ( 住所 ; "町" ) and
C < T ;
        Left ( ADDRESS ; C ) ;

// 町の後に市がくる場合、町を優先
PatternCount ( 住所 ; "町" ) and
PatternCount ( 住所 ; "市" ) and
T < C ;
        Left ( ADDRESS ; T ) ;

// 市の後に村がくる場合、市を優先
PatternCount ( 住所 ; "市" ) and
PatternCount ( 住所 ; "村" ) and
C < V ;
        Left ( ADDRESS ; C ) ;

// 村の後に市がくる場合、村を優先
```

```
PatternCount ( 住所 ; "村" ) and
PatternCount ( 住所 ; "市" ) and
V < C ;
        Left ( ADDRESS ; V ) ;

// 町の後に村がくる場合、町を優先
PatternCount ( 住所 ; "町" ) and
PatternCount ( 住所 ; "村" ) and
T < V ;
        Left ( ADDRESS ; T ) ;

// 村の後に町がくる場合、村を優先
PatternCount ( 住所 ; "村" ) and
PatternCount ( 住所 ; "町" ) and
V < T ;
        Left ( ADDRESS ; V ) ;

// 市がある場合
PatternCount ( 住所 ; "市" ) ;
        Left ( ADDRESS ; C ) ;

// 町がある場合
PatternCount ( 住所 ; "町" ) ;
        Left ( ADDRESS ; T ) ;

// 村がある場合
PatternCount ( 住所 ; "村" ) ;
        Left (   ADDRESS ; V ) ;

// 区がある場合
PatternCount ( 住所 ; "区" ) ;
        Left (   ADDRESS ; K ) ;

// "町"・"村"が抜けてしまい"郡"だけがある場合
PatternCount ( 住所 ; "郡" ) ;
        Left (   ADDRESS ; G ) ;
```

```
        )
)
```

カスタム関数　YearNameToDate

デベロッパーさんの作った関数です。Vac_WD_public.fp12で利用させてもらってます。

YearNameToDate（ 日付テキスト ）

```
/*=============================================================

   カスタム関数　　YearNameToDate（ 日付テキスト ）

   ″平成16年1月1日″という元号で書かれたテキストから日付を得ます。
   ″H16.1.1″というテキストなら日付として解釈するファイルメーカーの機能を
   そのまま利用しています。
   （平成十六年一月一日のように漢数字が使われている場合は変換できません。）
=============================================================*/

GetAsDate（
    Substitute（
        日付テキスト ;
        [ ″西暦″ ; ″″ ] ;
        [ ″明治″ ; ″M″ ] ; [ ″明″ ; ″M″ ] ;
        [ ″大正″ ; ″T″ ] ; [ ″大″ ; ″T″ ] ;
        [ ″昭和″ ; ″S″ ] ; [ ″昭″ ; ″S″ ] ;
        [ ″平成″ ; ″H″ ] ; [ ″平″ ; ″H″ ] ;
        [ ″令和″ ; ″R″ ] ; [ ″令″ ; ″R″ ] ;
        [ ″元″ ; ″1″ ] ;
        [ ″年″ ; ″.″ ] ; [ ″月″ ; ″.″ ] ; [ ″日″ ; ″″ ]
    ）
）
// ©2004,SHAMOTO Syuji http://www.FMPro.jp
```

カスタム関数　IsValidTel

デベロッパーさんの作った関数です。Vac_WD_public.fp12で利用させてもらってます。

IsValidTel（ 電話番号 ； オプション ）

```
/*================================================================
カスタム関数　IsValidTel（ 電話番号 ； オプション ）

＜電話番号＞が妥当な場合に真「1」を返します。妥当ではない場合、偽「0」を返します。
オプションに「1」を指定することで偽の場合にその理由を返し、真の場合はnull値を返します。
（オプション「0」の場合の結果は「1」か「0」）

================================================================*/

Let（
    [
    OPTION = オプション ；
    TEL = Filter（ RomanHankaku（ 電話番号 ）；"0123456789"）；
    LEN = Length（ TEL ）；
    ONE = Left（ RomanHankaku（ 電話番号 ）；1 ）；
    THR = Left（ TEL ; 3 ）；
    FOU = Left（ TEL ; 4 ）；
    RESULT =
        Case（
            IsEmpty（ TEL ）；
                "電話番号が空です"；
            ONE = "+"；
                ""；        // 海外の電話番号としてスルー
            ONE ≠ "0"；
                "海外の電話番号の場合+ではじめてください"；        // 国内プレフィックスの0がない場合海外番号とする
```

```
FOU = "0570" and LEN ≠ 10 ;
    "統一番号(ナビダイヤル)は10桁です" ;
FOU = "0990" and LEN ≠ 10 ;
    "情報料代理徴収用電話番号は10桁です" ;
FOU = "0120" and LEN ≠ 10 ;
    "フリーダイヤル番号は10桁です" ;
FOU = "0800" and LEN ≠ 10 ;
    "フリーダイヤル番号は10桁です" ;

THR = "020" and LEN ≠ 11 ;
    "ポケベル番号は11桁です" ;
THR = "050" and LEN ≠ 11 ;
    "IP電話番号は11桁です" ;
THR = "070" and LEN ≠ 11 ;
    "PHS/携帯電話番号は11桁です" ;
THR = "080" and not (FOU = "0800") and LEN ≠ 11 ;
    "携帯番号は11桁です" ;
THR = "090" and LEN ≠ 11 ;
    "携帯番号は11桁です" ;

THR = "010" ;
    "010は現在使われておりません" ;
THR = "030" ;
    "030は現在使われておりません" ;
THR = "040" ;
    "040は現在使われておりません" ;
THR = "060" ;
    "060は現在使われておりません" ;       // FMC番号

not (
    THR = "020" or THR = "050" or THR = "070" or THR = "080" or
THR = "090"
) and
LEN ≠ 10 ;
    "電話番号は10桁です"      // 神岡MA(2007年2月1日より),小田
```

```
原MA（2007年2月25日より） の一部の9桁番号も現在は10桁
            )
        ];
        Case (
            OPTION ;
                RESULT ;
                IsEmpty ( RESULT )
        )
)
```