ORCA (Online Research Control system Architecture)

ORCA Script ガイド

ORCA Script Ver1.0

もくじ

ORCA Script とは

ORCA Script とは、**ORCA** 内で動作するプログラミング言語です。

ORCA Script を使用することで、**Web** アンケート実行時の操作に柔軟で複雑な操作を与えることができます。また、集計では、集計する回答データのフィルタリングや、回答データから新たな設問の作成といった制御を行うことができます。

ORCA Script を使用することで、アンケートの制御からデータの加工まで、幅広い操作を可能にします。

本ドキュメントでは、**ORCA Script** の言語説明と、**ORCA** 上でどのように **ORCA Script** が使用するのかを簡単に説明しています。

ORCA Script の基本

記述

構造

ORCA Script は以下のような構造を持ちます。

```
programall
  :  program EOF!
  ;

program
  :  ( statements )*
  ;

statements
  :  structuredStatement
  |  statement ';'
  ;

statement
  :  TK_BREAK
  |  TK_CONTINUE
  |  returnStatement
  |  declarationStatement
  |  expression
  ;

returnStatement
```

```
    :  'return' ( expression )?
    ;

declarationStatement
    :  'var' declarationStatementArg ( ',' declarationStatementArg )*
    ;

declarationStatementArg
    :  '$' IDENT ( = expression )?
    ;

structuredStatement
    :  compoundStatement
    |  ifStatement
    |  forStatement
    |  whileStatement
    |  doStatement
    |  funcStatement
    ;

compoundStatement
    :  '{' program '}'
    ;

ifStatement
    :  'if' '(' expression ')' compoundStatement ( ifStatementElsif )*
( ifStatementElse )?
    ;

ifStatementElsif
    :  'elsif' '(' expression ')' compoundStatement
    ;
```

```
    ifStatementElse
      :  'else' compoundStatement
      ;

    forStatement
      :  'for' '(' ( expression )? ';' ( expression )? ';' ( expression )?
')' compoundStatement
      ;

    whileStatement
      :  'while' '(' expression  ')' compoundStatement
      ;

    doStatement
      :  'do' compoundStatement 'while' '(' expression ')'
      ;

    funcStatement
      :  'function' IDENT '(' ( '$' IDENT ( ',' '$' IDENT )* )* ')'
compoundStatement
```

プログラムの流れ

ORCA Script では、記述された ORCA Script が上から順に実行されます。単純なスクリプトであれば、順次実行だけですべてのスクリプトを記述することが出来ます。

■順次実行

```
文1;
文2;
 :
 :
文3;
```

■反復実行

for 文や while 文を記述することで、条件をみたすまで実行を反復することが出来ます。

■分岐実行

if 文を記述することで、条件をみたすことで実行順を分岐することが出来ます。

単純なスクリプト

文の終端には、セミコロン(；)が必要です。セミコロンが無い場合は文法エラーとなります。

■1 を返す ORCA Script

```
1;
```

■Q1000 の選択肢１の選択状態を返す ORCA Script

```
Q1000.C[1];
```

条件文(分岐実行)

条件に合致した場合、ブロック内に記述された処理を行います。

■if

・条件式が真の場合処理を実行する

```
if ( 条件式 ) { 処理 }
```

■if-lse

・条件式が真の場合は処理１を実行し、偽の場合は処理２を実行する

```
if ( 条件式 )
{
        /* 処理１ */
}
else
{
        /* 処理２ */
}
```

■ **if -elsif**

・条件式１が真の場合は処理１を実行する

・条件式１が偽で条件式２が真の場合は処理２を実行する

・条件式１、条件式２がともに偽の場合は処理３を実行する

```
if ( 条件式１ )
{
        /* 処理１ */
}
elsif( 条件式２ )
{
        /* 処理２ */
}
else
{
        /* 処理３ */
}
```

ループ文(反復実行)

条件に合致するまで、ブロック内に記述された処理を繰り返し行います。

■ for

for 文は条件式が **true** を返すまで処理を繰り返します。

for 文の動作は以下のようになります。

・初期化式は、ループの最初に一回だけ実行する

・処理の最後に更新式を実行し条件式を評価する

・条件式が真の場合、再度処理を実行する

・条件式が偽の場合、ループを終了する

```
for ( $i = 0; $i < 10; $i += 1 )
{
        /* 処理 */
}
```

■ while

処理を実行後、条件式を評価します。条件式が **true** の場合は、再度処理を実行し、**false** の場合はループを終了します。

処理が必ず一度は実行されます。

```
while ( $i < 10 )
{
        /* 処理 */
}
```

■ **do-while**

処理を実行後、条件式を評価します。条件式が **true** の場合は、再度処理を実行し、**false** の場合はループを終了します。

処理が必ず一度は実行されます。

```
do
{
        /* 処理 */
}
while ( $i < 10 )
```

ループ制御

ループ内で、条件に合致した場合にループを制御します。

■ break

ループ内で使用し、ループを終了させます。

```
for ( $i =0; $i < 10; $i += 1 )
{
        /* 処理1 */
        if ( 条件式 ) { break; }
        /* 処理2 */
}
```

■ continue

ループ内で使用し、この文以降の処理を行わず、次のループに
移行し評価を行います。
for の場合、continue 文の実行後、更新式を実行し、評価式を評価します。

```
for ( $i =0; $i < 10; $i += 1 )
{
        /* 処理1 */
        if ( 条件式 ) { continue; }
        /* 処理2 */
}
```

trueとfalse(真偽値)

ORCA Scriptでは、式や関数の結果が成立した場合にtrue(真)を、成立しなかった場合にはfalse(偽)を返します。

trueを返すとは、数字の1を返すことを示します。またfalseを返すとは、1以外を返すこと示します(空値、文字列などを含む)。

ORCA Scriptのtrueは数字の１と同値です。例えばtrue、on、yesなどの定数は常に1を返しますので、結果として真を返すことになります。

以下の表記は見かけ上違いますが全て同じ意味になります。

■true(真)を返す

```
retrun 1;
return true;
return on;
return yes;
```

■Q1100の選択肢１を選択

```
Q1100.C[1] = 1;
Q1100.C[1] = on;
```

同じように1以外を常に返すようにすると、常にfalseを返すことになります。

■false(真)を返す

```
retrun 0;
return false;
return off;
return no;
```

result ()

result()は、最後に評価された値を格納する変数です。

ORCA Scriptの記述欄に明示的に返す値が指定されていない場合は、最後に格納された値(=result)が返されます。

変数

変数は、値を遷移させるための容れ物のようなものです。

使用する場合は、関数で用いる場合以外は必ず宣言**(var)**が必要です。また、変数は必ず接頭に「**$**」を伴います。また、変数の値を参照するには、必ず事前に宣言していなければなりません。

スカラー値

単一の数値、文字列を格納している変数です。

■**$scalar** はスカラー値

```
var $scalar = 1;
var $scalar = "ABC";
```

配列

スカラー値の連続を格納している変数です。

■**$array** は配列

```
var $array = [1, 2, 3];
var $array = ["A", "B", "C"];
```

リスト

一時的なスカラー値の配列を意味します。

■選択肢 **1,2,3** のリスト

```
Q1000.C[1,2,3];
```

定数

定数とは、**ORCA Script** に最初から登録されている文字列のことです。

これらの文字列は、**ORCA Script** 上で特別な意味を持ちます。

整数

ORCA Scriptn では、記述された数値は **10** 進数表記の数値として扱われます。

```
1;
10;
-10;
```

16 進数

16 進数を表現するには、接頭に「**0X**」と記述することで **16** 進数として扱われます。

```
0x1234
0xabcd
0ABcD
```

小数

小数を表現するには、ピリオドなどの小数表記で小数として扱われます。

```
1.2
1.5e2
1.5e-2
-1.2
1.2e2 等
```

配列定数

[]内に囲まれた値（文字列、数値）は配列として扱われます。連続する値はカンマ（**,**）で区切ります。

```
[, , ... ]
```

```
[ 1, 2, 3, 4, 5 ]
[ "aaa", "bbb", "ccc" ]
```

特殊表記

通常入力の出来ない、または通常表記では表現できない特殊な文字列の表記です。

接頭にエスケープ文字列として「****」を記述します。

■\n

```
改行
```

■\r

```
改行
```

■\t

```
改行
```

■\"

```
ダブルクォーテーション
```

■\'

```
シングルクォーテーション
```

■

```
円マーク
```

■\uFFFF

```
16 進表記(¥u0123、¥uabcd、¥uABCD 等 16 進数 4 桁で表記)
```

■\000

```
8 進数表記（¥012、¥47 等 8 進数で表記）
¥0～¥377 まで(0～255)
```

定義済み文字列

ORCA Script に最初から定義された文字列です。常に決まった値を返します。

■skip

常に-2 を返す

■error

常に-1 を返す

■true, ok, yes, on

常に 1 を返す

■false, ng, no, off

常に 0 を返す

回答情報関数と設問タイプ

スペック内で設問を取り扱う **ORCA Script** を使用する際には、設問タイプによって得られる値や使用できる関数が異なることを意識しなければなりません。

以下にまとめられた設問タイプごとの回答情報関数は、設問タイプに使用することを前提とされた関数ですが、設問タイプが異なっていても特にエラーにはなりません。ただし、予期せぬ動作をまねく原因になりますので、記述に際しては注意が必要です。

数値回答

設問から取得できる値、設定する値ともに数値のスカラー値です。

■回答情報関数(数値回答)

関数名	説明
number, N	入力欄の数値
min, MI	設定された入力欄の最小値
max, MX	設定された入力欄の最大値

文字回答

設問から取得できる値、設定する値ともにスカラー値です。

■回答情報関数(文字回答)

関数名	説明
value, V	入力欄の値
number, N	数値として扱った入力欄の値

単一回答

設問から取得できる値、設定する値ともに数値のスカラー値です。

■回答情報関数(単一回答)

関数名	説明
multiple	設問タイプが複数回答かの真偽値
choice, C	設問または選択肢が選択されているかの真偽値
choicelist, CL	選択肢番号のリスト
choicetext, TX	選択肢名称
choicevalue, CV	追加入力欄の値
valuecount, VC	追加入力欄の個数

複数回答

設問から取得できる値、設定する値は数値のスカラー値、もしくは配列です。

■回答情報関数(複数回答)

関数名	説明
multiple	設問タイプが複数回答かの真偽値
limitupper, LU	選択できる選択肢の数
limitlower, LL	選択できる選択肢の数
choice, C	設問または選択肢が選択されているかの真偽値
choiceall, CA	選択肢がすべて選択されているかの真偽値
count, CT	選択された選択肢の個数
choicelist, CL	設定されている選択肢番号のリスト
selection, S	選択された選択肢番号のリスト
unselection, US	選択されていない選択肢番号のリスト
excludes, E	排他選択肢番号のリスト
choicetext, TX	選択肢名称
choicevalue, CV	追加入力欄の値
valuecount, VC	追加入力欄の個数

その他

その他の回答情報関数は、どの設問タイプにも共通して使用できます。

ただし、1つだけ例外として **currentstep** 関数があります。この関数はステップ設問のステップ数を返す関数で、上記のどの設問タイプとも異なる制御設問です。この関数は、ステップ設問にのみ有効です。

コメント

ORCA Script では**/* */**で囲まれた記述は、コメントとして機能し、**Script** として解釈されません。
また、**//**もしくは**#**で始まった行の文字列も同様にコメントとして機能します。

■ブロックコメント

```
/*
        コメント
*/
```

※ブロックコメントはネスト（入れ子）にすることができません

■一行コメント

```
//      コメント
```

```
#       コメント
```

設問の表現

設問の記述

ORCA Script 上で設問を表現する場合は、接頭の「Q」に続いて数値(設問番号)を記述します。
例えば設問番号 1000 の設問を表す場合は、「Q1000」というように記述します。

■設問番号 1000 の設問

```
Q1000
```

回答情報関数の記述

設問情報を記述した場合、必ず続けて回答情報関数を記述します。
回答情報関数は、設問の後に「.」(ピリオド)で区切って関数を記述します。

■設問番号 1000 の設問に choice 関数を使用する
例えば設問番号 1000 の設問で関数「choice」を使用する場合は、「Q1000.choice」と記述します。
※choice を使用する場合の設問番号 1000 はプリコード(選択肢)設問でなければなりません

```
Q1000.choice;
```

■設問番号 1000 の設問に choice 関数で、選択肢番号 1 を指定する
また、選択肢の設定や指定をともなう関数では、選択肢を[]内に記述をします。例えば、設問番号 1000 の選択肢番号１の選択を場合は、「Q1000.choice[1]」のように記述します。

```
Q1000.choice[1];
```

■設問番号 1000 の設問に choice 関数で、選択肢番号 1～4 を指定する

複数の選択肢を選択する場合は、配列と同様の記述します。

```
Q1000.choice[1, 2, 3, 4];
```

```
Q1000.choice[1~4];
```

マトリクス設問の記述

ORCA Script 上でマトリクス設問の従属設問を表現する場合には、接頭に「P」と記述し、続けて[ ]内に選択肢を記述します。

「P」のことを特に階層修飾子と呼びます。

■マトリクス設問の記述

```
P[マトリクス設問の選択肢番号].Q設問番号
```

マトリクス設問

	選択肢A	選択肢B	選択肢C
設問１	□	□	□
設問２	□	□	□
設問２	□	□	□

上図のようなマトリクス設問を例にとって説明します。

・表側（選択肢１～３）がマトリクス設問(マトリクス子設問)で、設問番号が 1000

・表頭（A～C）が従属設問(子設問選択肢)で、設問番号が 1100

■マトリクスの親設問

子設問を従える設問のことを特に親設問と呼びます。該当するのは、バンク設問、ステップ設問、マトリクス設問があてはまります。

マトリクス設問の場合、分析軸となるマトリクス子設問（設問の種類がマトリクス設問）のことを親設問と呼びます。

上図では、表側（選択肢１～３）が親設問のに該当します。

■マトリクスの従属設問

親設問にぶら下がった設問を従属設問と呼びます。

上図では、表頭（選択肢１～選択肢３）が従属設問(子設問選択肢)に該当します。

■マトリクス設問の選択

親設問の選択肢２(マトリクス子設問)のA、B、Cの選択を取得する場合は、

```
P[2].Q1100.S;
```

のように記述します。

記述内容は、親設問（=P)の選択肢 B（=[2]）に登録された Q1000 の selection で取得した値という意味になります。

下図の赤で囲まれた領域が、上記の記述で指定された範囲になります。

	選択肢１	選択肢２	選択肢３
A	☐	☐	☐
B	☐	☐	☐
C	☐	☐	☐

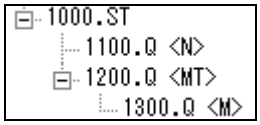
ステップ設問の記述

マトリクス設問と同様に、親設問をともなうステップ設問も同様に接頭に「P」と記述し、続けて[ ]内に選択肢を記述します。

■ステップ設問の記述

P[選択肢番号].Q 設問番号

ステップ設問



上図のようなステップ設問を例にとって説明します。

・ステップ設問（**1000.ST)**が親設問で、選択肢（１～３）が登録されている

・従属設問は **2** 問で、数値回答（**1100.Q**）とマトリクス設問**(1200.Q-1300.Q)**

■ステップ設問の親設問

従属設問を従える設問のことを特に親設問と呼びます。作成されたステップ設問は自動的に親設問となります。

上図では **1000** 番のステップ設問が親設問になります。

また、マトリクス設問の従属設問の場合、親設問が二つあることになります。詳しくは以下で説明します。

■ステップ設問の従属設問

親設問にぶら下がった設問を従属設問と呼びます。

上図では、**1100** 番の数値回答と、**1200** 番と **1300** 番のマトリクス設問が従属設問に該当します。

■ステップ設問内の設問の選択

親設問の選択肢２(ループ回数 2 回目)で、数値回答（1100 番）の値を取得する場合は、

```
P[2].Q1100.N;
```

のように記述します。

記述内容は、親設問（=P)のループ２回目の（=[2]）Q1100 の numeric の指定した値という意味になります。

また、ステップ設問内の Q1100 をすべて取得したい場合は、

```
P[1~3].Q1100.N;
```

のように記述します。

この場合、取得できる値は配列になります。

取得できる値は以下のようになります。

```
var $array = P[1~3].Q1100.N;
----------------------------------------
[結果] $array[0] はステップ設問選択肢１の Q1100 の回答 です。
       $array[1] はステップ設問選択肢２の Q1100 の回答 です。
       $array[2] はステップ設問選択肢３の Q1100 の回答 です。
```

また、ステップ設問内で設問を指定する場合は、Ｐの記述を省略することができます。

その場合はステップ設問の選択肢（ループ順番）を指定することができません。選択肢は、ステップ設問で実行中の選択肢番号を使用します。

■ステップ設問内のマトリクス設問の選択

親設問の選択肢２(ループ回数 2 回目)で、マトリクス設問（1200 番）の選択肢１の選択内容を取得する場合は、

```
P[2].P[1].Q1300.S;
```

のように記述します。

記述内容は、親設問（**=P)**のループ２回目の（**=[2]**)、マトリクス設問 **Q1200** の選択肢２の **selection** で指定した値という意味になります。

このように複数の親を持つ従属設問を直接指定する場合は、**P** の階層修飾子を連続して記述します。

演算子

算術演算子

■算術演算子

演算子	説明
＋	加算
-	減算
*	乗算
/	除算
%	剰余算

例）**$a** と **10** の積に１を加えたものを**$num** に代入する

```
$num = $a * 10 + 1;
```

例）**$a** を３で割った余りを**$num** に代入

```
$num = $a % 3;
```

数値比較演算子

■数値比較演算子

演算子	説明
==	数値一致
!=	数値不一致
<>	数値不一致
<	数値小なり
<=	数値小なり、または一致
>	数値大なり
>=	数値大なり、または一致

例）$i が 10 より小さければ **true(真)**、10 以上ならば **false(偽)**

```
$i < 10;
```

例）$num が 0 でなければ **true(真)**、0 ならば **false(偽)**

```
$num != 0;
```

文字列比較演算子

■文字列比較演算子

演算子	説明
eq	文字列一致
ne	文字列不一致
lt	文字列小なり
le	文字列小なり、または一致
gt	文字列大なり
ge	文字列大なり、または一致

例）$char が「ABC」ならば **true(真)**、「ABC」でなけれ **false(偽)**

```
$char eq "ABC";
```

例）$char が「ABC」でなければ **true(真)**、「ABC」ならば **false(偽)**

```
$char ne "ABC";
```

論理演算子

■論理演算子

演算子	説明
|	論理和
||	論理和
or	論理和
&	論理積
&&	論理積
and	論理積
!	論理否定

例）$num が数値の１もしくは$char が「ABC」ならば **true(真)**、
　　$num が数値の１もしくは$char が「ABC」でなければ **false(偽)**

```
$num == 1 | $char eq "ABC";
```

上記と同意

```
$num == 1 || $char eq "ABC";
```

例）$num が数値の１かつ$char が「ABC」ならば **true(真)**、
　　$num が数値の１かつ$char が「ABC」でなければ **false(偽)**

```
$num == 1 & $char eq "ABC";
```

代入演算子

■代入演算子

演算子	説明
=	代入
+=	加算代入
-=	減算代入
*=	乗算代入
/=	除算代入
%=	剰余算代入
<<=	文字結合代入

例）$num に「3」を代入

```
$num += 3;
```

例）$char が「ABC」でなければ **true(真)**、「ABC」ならば **false(偽)**

```
$char <<= "ABC";
```

演算子の優位順位と結合規則

ORCA Script の演算子には優位順位があります。

例えば、四則演算では乗算と除算が加算と減算に対して優先されるように、演算子にも同様に優先順位があります。

$a = $b < 0;というような記述があった場合、 先に$b < 0 が評価され、その結果が$a に代入されます。

優先順位は以下の表に従います。

優先度	演算子	結合規則
HIGH ↑	() [] .	左
	!	右
	* / %	左
	+ - <<	左
	<= >= < > lt gt le ge	左
	== != eq ne	左
	& && || and or	左
	=	右
LOW ↓	,	左

関数の作成

ORCA Script では、関数を作成することが出来ます。
関数とは、指定された処理を行い、その結果を返す、一群の処理に名前をつけたものです。
定義された関数は、同一入力欄で呼び出すことが可能です。
ただし、呼び出す関数は、必ず呼び出す位置より前に作成されていなければなりません。

関数定義

■関数なしの関数定義と呼び出し

```
function callfunc () // 関数の定義
{
      処理
};

callfunc()// 関数の呼び出し
```

■引数つきの関数定義

```
function callfunc( $a, $b )
{
      処理
};

callfunc( 1, "ABC" );
```

サンプル

■消費税(5%)をもとめる関数

```
/*
        元値から消費税(5%)を算出する
        (0 以下の金額の場合はエラー値を返す)
*/
function tax( $price )
{
        if( $price < 1 ) // $price が 1 円未満の場合エラー値を返す
        {
                return -1;
        }
        else            // $price が 1 円未満でなければ 1.05 倍する
        {
                return $price * 1.05;
        }
};

tax( Q1100.N );  // 作成した関数 tax に、引数として Q1100 の回答を渡す
```

組み込み関数

ORCA Script には、使用頻度の高そうな関数や便利な関数が最初から組み込んであります。

これらの関数の詳しい使用方法については、別紙 **ORCA Script** リファレンスを参照ください。

実際の使用

ORCA Script は ORCA の様々な場所で使用できます。

スペック上では、分岐をはじめとして、文字列や回答データの操作などアンケートを表現力豊富かつ高度な制御を提供することができます。また、集計で使用する場合は、回答データのフィルタリングをはじめとして、集計したデータからさらに回答データを作成するといった回収した情報の加工などに使用します。

これら様々な場所で使用できる ORCA Script は、どの場面でも同一の書式で表現可能です。しかし、使用箇所によって使用意図などが異なる場合や、使用できる ORCA Script にも制限が出てきます。特にスペック上で記述する場合に、どの個所で書くのが適切なのかで迷ってしまうことがあるかもしれません。

本章ではこの点をふまえ、ORCA Script の使用箇所や場面などでの ORCA Script の簡単な例と使用用途などを説明していきます。

ORCA Editor での記述(スペックファイル)

記述する場所

特定の動作（たとえばジャンプや自動回答など）を行わせるには当然、そのアクション部位に記述する必要がありますが、必要な値の設定や代入などはスペックの状態にあわせて好きな場所に記述してかまいません。

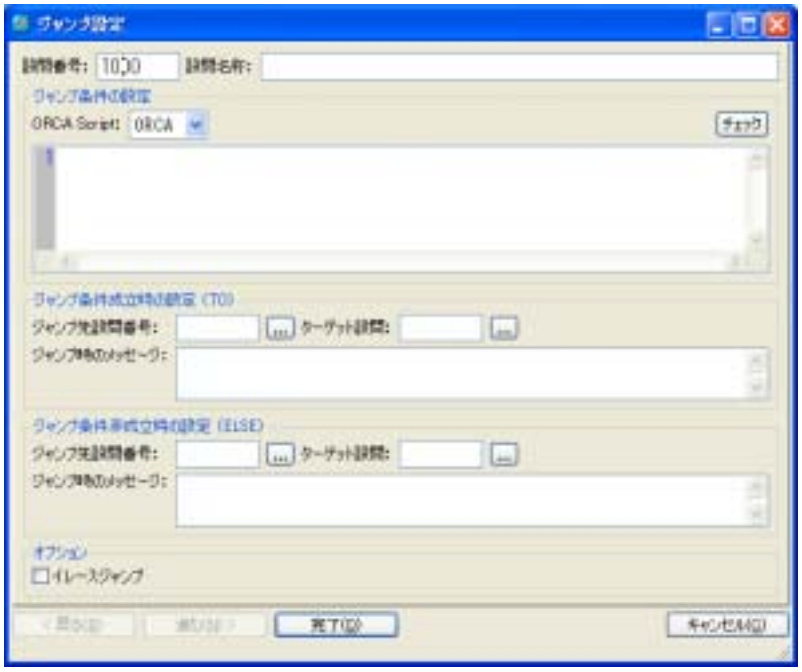
当然、記述や記述箇所によっては値がうまく反映できないこともありますので、その点を把握して適した場所に記述するのが望ましいことです。

このドキュメントでは、特に動作や設定に関する記述は、ジャンプに記述することを前提として進めていきます（詳しくはジャンプの「ジャンプを目的としないジャンプ設問」で説明します)。もちろん、この記述以外の方法でも同じ動作を行うことは可能です。

ジャンプ

ジャンプは、記述された **ORCA Script** の結果から **true** もしくは **false** を判定し、設問先を分岐する動作を行う制御設問です。

■ジャンプ設定画面



上のイメージは、**ORCA Editor** でのジャンプ設問の設定画面です。

ORCA Script はジャンプ条件の設定に記述します。

ORCA Script を記述した結果が **true** となった場合、ジャンプ条件成立時の設定にあるジャンプ先設問番号にジャンプします。

同様に **false** となった場合、ジャンプ条件非成立時の設定にあるジャンプ先設問番号にジャンプします。

サンプルスペック

以上のような設問のスペックを作成して、ジャンプの実際の動作を確認します。

- **Q1000** 単一回答　選択肢１～３
- **Q1100** ジャンプ
- **Q1200** 複数回答　選択肢１～３
- **Q1300** 文字回答

何もしないジャンプ設問

まずはジャンプしないジャンプ設問を作成します。
ジャンプ先設問番号に **1200** を指定し、もっともシンプルな **ORCA Script** を記述します。

```
1;
```

ORCA Script では、数値の **1** が **true**(真)となります。ですので、数値の **1** 以外は常に **false**(偽)です。

この記述だと常に **1** が返りますので、結果は常に **true** になります。下記の記述も同様の意味を持ちます。

```
true;
```

この記述の結果、このジャンプ設問を通っても常に条件が成立し、次の **1200** 番の設問に移動することになります。

同じように以下の記述だと条件は常に非成立になります。

```
0;
```

ちなみに、何もしないジャンプとしましたが本当に何もしていないわけではありません。実際には、

ORCA Script の「1;」という式を評価して、真偽を判定してジャンプの処理を行っています。
その結果、このジャンプ設問は回答者にとっては何もしていないことになります。

選択肢を判定してジャンプする

次に関数を使用した ORCA Script を作成します。
Q1000(単一回答)の選択肢１を選んだ場合に 1300 番にジャンプする ORCA Script を記述します。

```
Q1000.C[1];
```

上の記述で Q1000 の選択肢 1 番の選択状態を表現しています。
回答者が選択肢１を選択した場合、結果は **true** となり条件が成立します。また、選択肢１を選択しなかった場合は、結果が **false** となり条件が非成立となり、1200 番の設問に移動します（ジャンプ先の設問が設定していない場合は、設問番号順に推移します)。

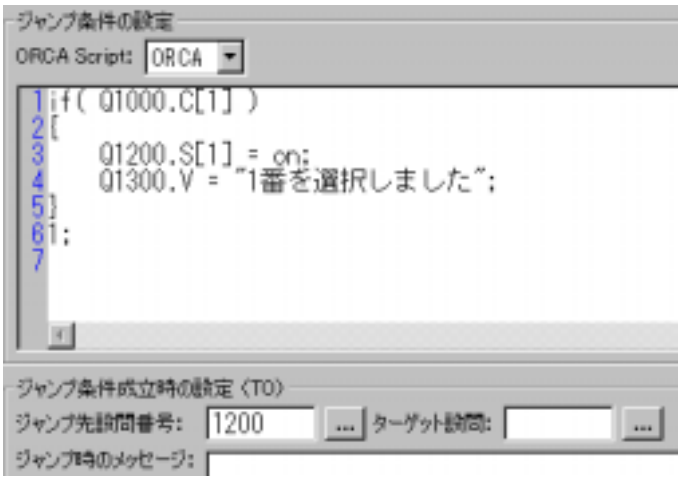
```
if( Q1000.C[1] )
{
      return true;
}
```

ジャンプは一般的なプログラム言語の **if** 文に相当するものです。あえて書けば「Q1000.C[1];」は下記の記述と同様のスクリプトが記述してあることになります。
どちらの記述でも結果は変わりません。

ジャンプを目的としないジャンプ設問

ジャンプ設問に記述する ORCA Script は、必ず特定の判定をしなければならないわけではありません。上述の「何もしないジャンプ設問」のように常に、1 を返すことで、他の処理を行う ORCA Script を記述することもできます。例えば、回答結果を受けての設問の設定や、回答の計算結果などを他の設問に事前に入力することが可能です。

◇Q1000 の選択肢１を選択した場合(成立時のジャンプ先を 1200 番に設定)



Q1200(複数回答)の選択肢１をチェックし、Q1300(文字回答)に 1 番を回答欄に入力する

```
if( Q1000.C[1] )
{
	Q1200.S[1] = on;
	Q1300.V = "1 番を選択しました";
}
1;
```

この ORCA Script では、Q1000 の選択肢１が選択された場合の処理を記述しています。
選択肢１が選択されて、このジャンプを通過した場合、まず Q1200 の選択肢１をチェックします。次に Q1300 番に"１番を選択しました"という文字列を代入します。
最後に、条件を成立させるために 1 を記述しています。if 文で囲まれた ORCA Script とは無関係に常に条件が成立します。

選択肢の表示

選択肢の表示、非表示を **ORCA Script** で制御することが出来ます。

ORCA Script は、選択肢設定の **ORCA Script** の欄に **ORCA Script** を記述します。

ジャンプ設問と同様に **ORCA Script** の結果から **true** か **false** を判定し、**true** の場合は該当選択肢を表示し、**false** の場合は非表示となります。

ジャンプのサンプルスペックと同様のスペックで、**Q1000(**単一回答**)**の選択肢を選択した時の **Q1200(**複数回答**)**の表示設定を記述してみます。

■**Q1000** の選択した選択肢から **Q1200** の選択肢の表示・非表示を設定する

◇**Q1300** の **ORCA Script**

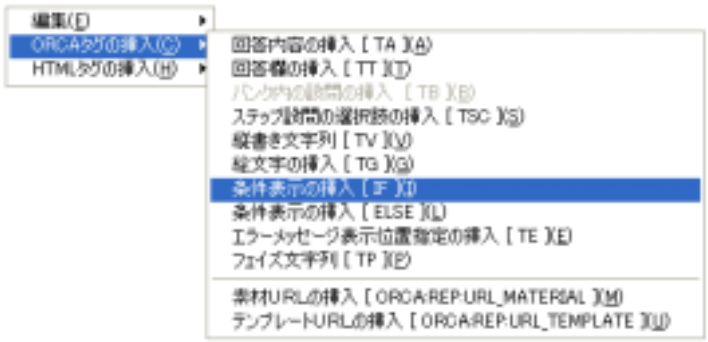
選択肢	排他	ORCA Script	画像/ボタン名
	☐	Q1000.C[1];	
	☐	Q1000.C[2];	
	☐	Q1000.C[3];	

上記の **ORCA Script** では、**Q1000** の選択肢１が選択された場合、**Q1200** の **Q1000.C[1]**の条件が成立し、他の条件は非成立となるため **Q1000.C[1]**の選択肢のみが表示されます。**Q1000** の選択肢 **2** と **3** も同様の動作で表示・非表示が決定されます。

条件表示の挿入（ORCA タグ）

ORCA では設問画面で条件に応じて文面を切り替えることが出来ます。

ORCA Editor の「設問文章」画面で右クリックすると表示される「ORCA タグの挿入」の中にある「条件表示の挿入」でその機能を提供します。この「条件表示の挿入」にも ORCA Script が必要になります。タグに表示される<EXP>に ORCA Scritp を差し込みます。記述の際には、;(セミコロン)も必要です。



◇条件表示の挿入を使用した例

```
<IF><EXP>Q1000.C[1];</EXP>
表示文字列
</IF>
```

「IF」と「ELSE」

条件表示の挿入には「IF」と「ELSE」の 2 種類があり、それぞれ挿入したときの内容が異なります。

◇条件表示の挿入(IF)を挿入した場合

```
<IF><EXP></EXP>


</IF>
```

◇条件表示の挿入(ELSE)を挿入した場合

```
<ELSE/><EXP></EXP>
```

ELSE は ELSE 単体で用いることは出来ません(エラーになります)。必ず、<IF>～</IF>の中におさまっていなければなりません。

実際に ELSE を使う場合は、以下のようになります。

◇ELSE をともなった条件表示

```
<IF><EXP></EXP>

<ELSE/><EXP></EXP>
</IF>
```

条件表示の動作

条件表示(IF)は、言葉の示すとおり特定の条件が成立した場合のみ、任意の文字列を表示することが出来ます。この際、条件に該当する部分に ORCA Script を使用します。条件の ORCA Script の結果から **true** か **false** を判定し、**true** だった場合は設定した文字列を表示します。**false** の場合は何も表示されません。

◇条件表示(IF)の書式

```
<IF><EXP>ORCA Script</EXP>
表示したい文字列
</IF>
```

条件表示(ELSE)は、IF の ORCA Script が非成立だった場合に判定されます。ELSE に条件となる ORCA Script を記述した場合は、ORCA Script の結果から **true** か **false** を判定し、**ture** だった場合は設定した文字列を表示し、**false** の場合は何も表示しません。ただし、ELSE の条件が記述されていなかった場合は、事前の IF または ELSE の条件がすべて非成立の場合に条件の記述されていない ELSE の文字列が表示されます。

まとめると以下のようになります。

■条件表示の書式

```
<IF><EXP>ORCA Script1</EXP>
ORCA Script1 の条件が成立した時の文章
<ELSE/><EXP>ORCA Script2</EXP>
```

```
ORCA Script1 が成立せず、ORCA Script2 が成立した時の文章
<ELSE/><EXP/>
ORCA Script1 と ORCA Script2 がともに成立しなかった時に表示する文章
</IF>
```

■サンプル

・Q1000(単一回答)の選択肢１を選択時は「選択肢１を選択された方へ」と表示

・Q1000(単一回答)の選択肢 2 を選択時は「選択肢２を選択された方へ」と表示

・Q1000(単一回答)の選択肢 1,2 以外を選択時は「選択肢 1,2 以外を選択された方へ」と表示

```
<IF><EXP>Q1000.C[1];</EXP>
選択肢１を選択された方へ
<ELSE/><EXP>Q1000.C[2];</EXP>
選択肢２を選択された方へ
<ELSE/><EXP/>
選択肢１,２以外を選択された方へ
</IF>
```

自動回答設問

自動回答設問とは、該当する自動回答の設問番号を回答者が通過したときに自動的に値を取得、または設定を行うといった動作を行った上で、その結果を回答データに落とす設問のことです。また回答者には設問画面が表示されません。

自動回答でも **ORCA Script** を使用できる設問が用意されています。

自動回答設問の種類と設問タイプ

自動回答設問には様々な種類があります。これらの種類はそれぞれ異なる設問タイプを有しています。この設問タイプを意識せずに使うと期待していた回答データが得られないケースが発生します。

例えば **ORCA Script(**数字回答**)**の **ORCA Script** に「**return "A";**」と記述しても回答データに文字列の **A** にはなりません。これは、数字回答の設問タイプでは、必ず数値のスカラー値が回答となるという仕様に則したものです。（このケースだと数値の１が回答となります）

また、**ORCA Script** で回答データを取得する際も設問タイプが分かっていないと適切にデータを処理することができません。

■自動回答の種類と設問タイプの一覧

種類	**設問タイプ**
ORCA Script(単一回答**)**	単一回答
ORCA Script(複数回答**)**	複数回答
ORCA Script(数字**)**	数値回答
ORCA Script(文字**)**	文字回答
カウンタ	数値回答
選択肢の単一回答	単一回答
選択肢の複数回答	複数回答
クォータシグナル	数値回答
クォータ回収値	数値回答
ランダム値	数値回答
開始時刻	数値回答
経過時間	数値回答
クライアントＩＰ	文字回答
クライアントＵＡ	文字回答

クライアントリンク元	文字回答
アクセスパラメータ（数値）	数値回答
アクセスパラメータ（文字）	文字回答
アクセスシーケンス	数値回答
対象者情報：テーブルＩＤ	文字回答
対象者情報：回答者ＩＤ	文字回答
対象者情報：カテゴリＩＤ	文字回答

ORCA Script の使用できる自動回答の種類

■**ORCA Script**(単一回答)

プリコード設問の回答タイプが単一回答の種類に相当する自動回答設問です。

ORCA Script を使用して単一回答の回答データが可能です。

■**ORCA Script**(複数回答)

プリコード設問の複数回答に相当する自動回答設問です。

ORCA Script を使用して単一回答の回答データが設定可能です。

■**ORCA Script**(数値)

数値入力回答設問に相当する自動回答設問です。

ORCA Script を使用して単一回答の回答データが設定可能です。

■**ORCA Script**(文字)

文字回答入力設問に相当する自動回答です。

ORCA Script を使用して文字回答の回答データが設定可能です。

■選択肢の単一回答

プリコード設問の単一回答に相当する自動回答設問です。

ORCA Script を使用して選択肢の操作を行った単一回答のデータの格納・取得が可能です。

■選択肢の複数回答

プリコード設問の複数回答に相当する自動回答設問です。

ORCA Script を使用して選択肢の操作を行った複数回答のデータの格納・取得が可能です。

これらの種類の中でも特に、**ORCA Script** は **ORCA Script** のみで機能するように作成された自動回

答です。設問の制御、回答データの操作・設定などを行う場合は、設問タイプにあった **ORCA Script** を利用するとよいでしょう。

ORCA Script の記述がない自動回答

自動回答の **ORCA Script**(単一回答、複数回答、数値、文字)の **ORCA Script** を記述していない場合、この設問を通過した回答結果はすべて **1** となります。

ORCA Script を使用して、事前に自動回答に値を入力している場合にも、**ORCA Script** を記述しない場合はすべて **1** で上書きされてしまいます。事前に設定した値を自動回答に保持したい場合は、該当する自動回答をジャンプする必要があります。

ORCA Core での記述

ORCA Core で使用する ORCA Script

ORCA Core と ORCA Editor では、取り扱う対象の情報が異なります。

ORCA Editor で使用するスクリプトは、Web アンケート上で回答中の回答者(個人)に対しての情報の制御や取得を目的としています。

一方 ORCA Core では、回答の終了した回答者の情報を対象としています。ORCA Core で扱う情報は常に真偽値の結果から得られる回答者の集合となります。

例えば、Q1000.C[1]というような記述があった場合、Q1000 の選択肢１の選択状況をさします。

ORCA Editor では回答中の回答者が Q1000 の選択肢１を選択している状態が成立した場合、回答者個人の成立・不成立の情報のみが返ります。

一方、ORCA Core では回答したデータ中から、Q1000 の選択肢１を選択している状態の回答者すべての回答者を結果として返します。

サンプル除去設定

サンプル除去設定とは、集計した回答データから条件に該当したサンプルを除去し、集計の母数を制御します。ORCA Script を使用する場合は、「ORCA Script を使用する」を選択して ORAC Script を記述します。

条件の優先度は上から作成された設定のリストの順番に従います。下の条件は、上の条件で絞られたサンプルに対しての条件となります。

デフォルトでは、「終了 ID 100 以上」が登録されています。このスクリプトは、回答者が終了番号 100 番以上で終わった場合は、集計から除くことを示しています。終了 ID100 以上とは、デフォルトではカテゴリクォータ達成(101)、多重回答(102)をサンプルから除去していることを示しています。

```
last_id() >= 100;
```

last_id()は ORCA Core でしか使用できない関数です。この関数は、回答者の終了番号をしめしているので回答が完了した後でしか使用できないからです。

新アイテム作成

新アイテム作成は、集計用に作成する設問です。設問は全て **ORCA Script** で作成します。

例えば、以下のようなスペックでアンケートを実施したとします。

・**Q1000(**単一回答**)** １**.**男性 **2.**女性

・**Q1100(**数値回答**)** 年齢

このような条件で集計された回答データをもとに、成人男性・成人女性・未成人男性・未成人女性と分類して集計を行いたいとします。

その場合、新規設問で単一回答を選択して、以下のような **ORCA Script** を記述することで条件を満たす設問を作成することが出来ます。

■成人・未成人分類

CNo.1 成人男性

```
Q1000.C[1] & 20 <= Q1100.N;
```

CNo.2 成人女性

```
Q1000.C[2] & 20 <= Q1100.N;
```

CNo.3 未成人男性

```
Q1000.C[1] & 20 > Q1100.N;
```

CNo.4 未成人女性

```
Q1000.C[2] & 20 > Q1100.N;
```

絞り込み設定

絞り込み設定は、サンプル**(**サンプル除去済みの回答者データ**=**集計母数**)**に対して絞り込みを行う場合に使用します。条件は全て **ORCA Script** で記述します。

また、絞り込み設定では、アクションの「含める」「除外」を選択することで、**ORCA Script** の条件が成立した回答データを除外するか含めるかを設定します。

例えば、以下のようなスペックでアンケートを実施したとします。

・**Q1100(**数値回答**)** 年齢

成人のデータのみに絞り込みを行いたい場合には、以下のように記述します。

■集計を成人のみに絞り込み

No.1 未成人を除外()

```
20 > Q1100.N;
```

この ORCA Script を記述の上、アクションを「除外する」、有効/無効を「有効」に設定します。

スポット集計

スポット集計は、任意に複数の条件のともなう回収状況や結果を確認したい場合に作成します。
ORCA Script を記述することで、さらに条件を絞った状況を確認することが可能です。

例えば、以下のようなスペックでアンケートを実施したとします。

・Q1000(複数回答) １.男性 2.女性

・Q1100(数値回答) 年齢

・Q1200(説明文)

Q1100 で女性と答えて、Q1100 で 30 歳以上と答えた、Q1200 を通過したスポット集計を行いたい場合は以下のように記述します。

■説明文を読んだ 30 歳以上の女性の回答者

```
Q1000.C[2] & Q1100.N >= 30 & Q1200.PS;
```

クォータ編集

クォータ編集は、条件に合致した回答数が設定したクォータ(閾値)に達した時点でクォータアクションを実行する機能を編集する画面です。

クォータ編集の設問クォータの条件設定は ORCA Script を記述することで行います。

例えば、以下のようなスペックでアンケートを実施したとします。

・Q1000(複数回答) １.男性 2.女性

・Q1100(数値回答) 年齢

成人の数が 100 人になった時点でアンケートを終了したい場合には以下のように記述します。

■成人の数が 100 人でアンケートを締め切るクォータ設問

・クォータ数を 100 に設定

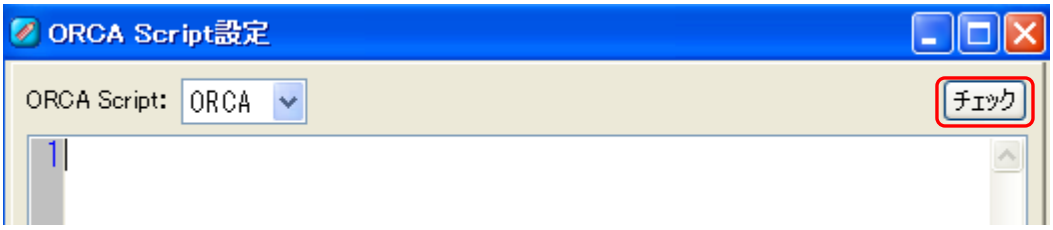
```
Q1100.N >= 20;
```

ORCA Script のエラーチェック

記述した ORCA Script が文法的に間違っていた場合、スペックがアップロードできなかったり、設定が保存ができないといったことになります。

たとえばセミコロン（；）のつけ忘れや、文法ミスなど様々なケースが考えられます。これらを手軽にチェックできるように、ORCA Editor と ORCA Core には ORCA Script の記述欄の近くにチェックボタンがあります。

■ORCA Editor のチェックボタン



■ORCA Core のチェックボタン

このチェックボタンを押すことで、記述された ORCA Script に記述エラーがないかを確認します。設定を保存する前に一度チェックをすると、ORCA Script を記述する作業効率があがることでしょう。

ただし、このチェックボタンはあくまでも簡単な文法のエラーを探し出すものです。記述自体の矛盾を指摘できるものではありません。

索引

ORCA Script ガイド

2008 年 6 月 2 日　初版発行

2008 年 9 月 2 日　第二版発行

発行者　株式会社サイズ

発行日　2008 年 6 月 2 日

連絡先　株式会社サイズ

〒150-0043

東京都渋谷区道玄坂 1-18-1　渋谷 INCS　7F

電話　03-5459-3817

URL　http://www.cyze.jp/

E-mail　info@cyze.jp