4D Japan Pack 6.8

リファレンス

Windows® and Mac™ OS

4D

4D Japan Pack 6.8 リファレンス
Windows® and Mac™ OS

このマニュアルに記載されている事項は、将来予告なしに変更されることがあり、いかなる変更に関しても4D SAは一切の責任を負いかねます。このマニュアルで説明されるソフトウェアは、本製品に同梱のLicense Agreement（使用許諾契約書）のもとでのみ使用することができます。

注意

このソフトウェアの使用に際し、本製品に同梱のLicense Agreement（使用許諾契約書）に同意する必要があります。ソフトウェアを使用する前に、License Agreementを注意深くお読みください。

目次

文字コードの変換（AJ_Pack : String Japan）

この章では文字コード変換コマンドについて説明します。

注：ルーチンのカテゴリで、String Japan は、4th Dimension バージョン6.0 でも使うことができます。

AJP Nkf

AJP Nkf (変換コード;変換元文字列;変換後文字列)→エラー

引数	タイプ		説明
変換コード	文字列	→	”-s” ：SJIS コードへの変換 ”-j” ：JIS コードへの変換 ”-e” ：EUC コードへの変換 オプションの先頭2バイト引数 ”-S” ：SJIS コードからの変換 ”-J” ：JIS コードからの変換 ”-E” ：EUC コードからの変換 特別 ”-mB” ：MIME 文字列から SJIS コードへの変換
変換元文字列	テキスト	→	変換する文字列
変換後文字列	テキスト	←	指定されたコードで変換された文字列
戻り値	倍長整数	←	エラーコード

説明

AJP Nkf は変換元文字列に渡された文字列を、変換コードで指定されたコードに変換し、その文字列を変換後文字列に返します。

第1引数に（”-s”や”-j”、”-e”などの）小文字のみの変換コードを渡した場合、**AJP Nkf** は変換元文字列の文字コードを自動的に判別します。

変換元文字列が変換コードに指定されたコードの文字列である場合、変換後文字列には変換元文字列がそのまま渡され、エラーは発生しません。

オプションの先頭2バイト引数を加えることで変換元文字列の文字コードを明示し、**AJP Nkf**による自動判別を省略することができます。これにより自動判別にかかる時間を節約することができます。

変換元の文字コードを間違えて指定した場合でも、**AJP Nkf**は正しい変換元文字コードを選択し、正しい変換を行います。

特別な変換コード”-mB”は、変換元文字列をMIME文字列として、それをSJISコードに変換します。

変換元文字列にはフィールドや変数を渡すことができます。リテラル文字列を渡すことはできません。

変換後文字列には変数を渡します。フィールドを渡すことはできません。

エラーコード

現時点では結果にかかわらず返り値に0が返るようになっています。これは将来の使用のために予約されているものです。

例：

次のコードは変換元文字列の文字コードを自動判別し、変換元文字列をJISコードに変換します。

```
$Error:=AJP Nkf (“-j”;[Mail]Body;Encoded_Strings)
```

[Mail]Bodyの内容をJISに変換し、変数Encoded_Stringsに返します。

次のコードは変換元文字列の文字コードをJISと指定し、それをSJISコードに変換しています。

```
$Error:=AJP Nkf (“-J-s”;Mail_Body;Decoded_Strings)
```

JISコードで記述された変数Mail_Bodyの内容をSJISに変換し、Decoded_Stringsに返します。

次のコードはMIME変換された文字列をSJISコードに変換します。

```
$Error:=AJP Nkf (“-mB”;Mime_Encoded_Strings;Decoded_Strings)
```

MIMEコードで記述された変数Mime_Encoded_Stringsの内容をSJISに変換し、Decoded_Stringsに返します。

参照

AJP mime Encode

AJP sjis to unicode

AJP sjis to unicode (変換元文字列;変換後文字列) → エラー

引数	タイプ		説明
変換元文字列	テキスト	→	SJIS コードで記述された変換元文字列
変換後文字列	テキスト	←	UNICODE に変換された文字列
戻り値	倍長整数	←	エラーコード

説明

AJP sjis to unicode はSJIS コードで記述された変換元文字列を、UNICODE に変換し、その文字列を変換後文字列に返します。

変換元文字列にはフィールドや変数を渡すことができます。リテラル文字列を渡すことはできません。

変換後文字列には変数を渡します。フィールドを渡すことはできません。

エラーコード

現時点では結果にかかわらず返り値に0が返るようになっています。これは将来の使用のために予約されているものです。

例：

```
$Error:=AJP sjis to unicode (SJIS_Strings;Unicode_Strings)
```

SJIS_Strings の内容を Unicode に変換し、変数 Unicode_Strings に返します。

参照

AJP unicode to sjis

AJP unicode to sjis

AJP unicode to sjis (変換元文字列;変換後文字列) → エラー

引数	タイプ		説明
変換元文字列	テキスト	→	UNICODEで記述された変換元文字列
変換後文字列	テキスト	←	SJISコードに変換された文字列
戻り値	倍長整数	←	エラーコード

説明

AJP unicode to sjisはUNICODEで記述された変換元文字列を、SJISコードに変換し、その文字列を変換後文字列に返します。

変換元文字列にはフィールドや変数を渡すことができます。リテラル文字列を渡すことはできません。

変換後文字列には変数をします。フィールドを渡すことはできません。

エラーコード

現時点では結果にかかわらず返り値に0が返るようになっています。これは将来の使用のために予約されているものです。

例：

```
$Error:=AJP unicode to sjis (Unicode_Strings;SJIS_Strings)
```

Unicode_Stringsの内容をSJISコードに変換し、変数SJIS_Stringsに返します。

参照

AJP sjis to unicode

AJP mime Decode

AJP mime Decode (変換元文字列;変換後文字列) → エラー

引数	タイプ		説明
変換元文字列	テキスト	→	MIMEエンコードされた変換元文字列
変換後文字列	テキスト	←	SJISコードに変換された文字列
戻り値	倍長整数	←	エラーコード

説明

AJP mime DecodeはMIMEエンコードされた変換元文字列を、SJISコードに変換し、その文字列を変換後文字列に返します。

変換元文字列にはフィールドや変数を渡すことができます。リテラル文字列を渡すことはできません。

変換後文字列には変数を渡します。フィールドを渡すことはできません。

エラーコード

現時点では結果にかかわらず返り値に0が返るようになっています。これは将来の使用のために予約されているものです。

例：

$Error:=**AJP mime Decode** (Mime_Strings;SJIS_Strings)

Mime_Stringsの内容をSJISコードに変換し、変数SJIS_Stringsに返します。

参照

なし

AJP mime Encode

AJP mime Encode (変換元文字列;変換後文字列) → エラー

引数	タイプ		説明
変換元文字列	テキスト	→	SJIS コードで記述された変換元文字列
変換後文字列	テキスト	←	MIME エンコードされた文字列
戻り値	倍長整数	←	エラーコード

説明

AJP mime Encode は SJIS コードで記述された変換元文字列を、MIME 変換し、その文字列を変換後文字列に返します。

変換元文字列にはフィールドや変数を渡すことができます。リテラル文字列を渡すことはできません。

変換後文字列には変数を渡します。フィールドを渡すことはできません。

エラーコード

現時点では結果にかかわらず返り値に 0 が返るようになっています。これは将来の使用のために予約されているものです。

例：

```
$Error:=AJP mime Encode (SJIS_Strings;Encoded_Strings)
```

SJIS_Strings の内容を MIME 変換し、変数 Encoded_Strings に返します。

参照

AJP Nkf

ウインドウユーティリティ（AJ_Pack : WinUtils）

この章ではWindows版の4Dで使用することができる、ウインドウユーティリティについて説明します。

注：ルーチンのカテゴリで、WinUtilsは、4th Dimensionバージョン6.0でも使うことができます。

AJP FRAME ENABLE

AJP FRAME ENABLE (フレームサイズ変更)

引数	タイプ		説明
フレームサイズ変更	整数	→	0：フレームサイズ変更不可 1：フレームサイズ変更可

説明

AJP FRAME ENABLEはWindowsのアプリケーションウインドウサイズをユーザーのマウス操作による変更可または不可に設定します。

このコマンドでサイズ変更不可に設定した場合も、**AJP SET FRAME WINDOW SIZE**コマンドを使用してアプリケーションウインドウのサイズを変更できます。

参照

AJP GET FRAME WINDOW SIZE、AJP SET FRAME WINDOW SIZE

AJP Get display height

AJP Get display height → ディスプレイの縦表示サイズ

引数	タイプ		説明
この関数には、引数はありません。			
戻り値	倍長整数	←	ディスプレイの縦表示サイズ

説明

AJP Get display Heightは現在のディスプレイ縦表示サイズをピクセル単位で返します。

この結果は「コントロールパネル」-「画面」の「画面領域」で設定された値と一致します。

参照

AJP Get display Width

AJP Get display width

AJP Get display width → ディスプレイの横表示サイズ

引数	タイプ		説明
この関数には、引数はありません。			
戻り値	整数	←	ディスプレイの横表示サイズ

説明

AJP Get display Width は現在のディスプレイ横表示サイズをピクセル単位で返します。

この結果は「コントロールパネル」-「画面」の「画面領域」で設定された値と一致します。

参照

AJP Get display height

AJP GET FRAME WINDOW SIZE

AJP GET FRAME WINDOW SIZE (左座標;上座標;右座標;下座標)

引数	タイプ		説明
左座標	倍長整数	←	ウインドウ左端の画面上の位置（ピクセル）
上座標	倍長整数	←	ウインドウ上端の画面上の位置（ピクセル）
右座標	倍長整数	←	ウインドウ右端の画面上の位置（ピクセル）
下座標	倍長整数	←	ウインドウ下端の画面上の位置（ピクセル）

説明

AJP GET FRAME WINDOW SIZEはアプリケーションウインドウのスクリーンに対する現在の位置をピクセル単位で取得し、引数に返します。

注：ウインドウを最小化した後にこのコマンドを呼び出した場合、**AJP GET FRAME WINDOW SIZE**は意味のない数値を返します。

参照

AJP SET FRAME WINDOW SIZE

AJP SET FRAME WINDOW SIZE

AJP SET FRAME WINDOW SIZE (左座標;上座標;右座標;下座標)

引数	タイプ	説明	
左座標	倍長整数	→	ウインドウ左端の画面上の位置（ピクセル）
上座標	倍長整数	→	ウインドウ上端の画面上の位置（ピクセル）
右座標	倍長整数	→	ウインドウ右端の画面上の位置（ピクセル）
下座標	倍長整数	→	ウインドウ下端の画面上の位置（ピクセル）

説明

AJP SET FRAME WINDOW SIZEはスクリーンに対する、引数に指定された座標にアプリケーションウインドウを移動します。

このコマンドはすでに開かれたアプリケーションウインドウにたいして有効です。例えば最小化されたアプリケーションウインドウを開くためにこのコマンドを使用することはできません。

参照

AJP GET FRAME WINDOW SIZE

AJP SET FRAME WINDOW STATUS

AJP SET FRAME WINDOW STATUS (ウインドウサイズ)

引数	タイプ		説明
ウインドウサイズ	整数	→	0：通常サイズのウインドウ 1：ウインドウの最大化 2：ウインドウの最小化

説明

AJP SET FRAME WINDOW STATUSはアプリケーションウインドウサイズを引数で指定されたサイズに設定します。

このコマンドを呼び出すことは、それぞれ対応するウインドウの最大化ボタン、最小化ボタン、あるいはウインドウサイズを元に戻すボタンをクリックすることと同じです。

参照

なし

AJP SHOW FRAME WINDOW TITLE

AJP SHOW FRAME WINDOW TITLE (タイトル表示)

引数	タイプ		説明
タイトル表示	整数	→	0：タイトルメニュー非表示 1：タイトルメニュー表示

説明

AJP SHOW FRAME WINDOW TITLEはWindowsのアプリケーションウインドウのタイトルを表示または非表示に設定します。

参照

なし

入力メソッド切換（AJ_Pack : IME&FEP）

この章では入力メソッド切り替えコマンドについて説明します。

AJP Get key input mode

AJP Get key input mode → 入力メソッドの状態

引数	タイプ		説明
この関数には、引数がありません。			
戻り値	倍長整数	←	Macintosh 0：入力メソッドオフ（英語モード） 1：入力メソッドオン（日本語モード） Windows 0：入力メソッドオフ 1：全角ひらがな 2：全角カタカナ 3：全角英数 4：半角カタカナ 5：半角英数 上記以外：ロック

説明

AJP Get key input modeは現在の入力メソッドの状態を返します。この関数は現在の入力モードを取得したいときに使用します。

Macintoshの場合、入力メソッドのオン・オフは通常日本語入力か英語入力かを表しますが、入力メソッド内部の設定状態をこの関数で取得することはできません。例えば入力メソッドがオン（返値=1）の場合でも、それが英数入力モードか、かな入力モードかを取得することはできません。

参照

AJP SET KEY INPUT MODE

AJP SET KEY INPUT MODE

AJP SET KEY INPUT MODE (入力モード)

引数	タイプ		説明
入力モード	倍長整数	→	0：入力メソッドオフ（英語モード） 1：入力メソッドオン（日本語モード） Windows 0: 入力メソッドオフ（IMEを通さない） 1: 全角ひらがな 2: 全角カタカナ 3: 全角英数 4: 半角カタカナ 5: 半角英数 -1: IMEをロック（IMEを閉じる） 10: 閉じているIMEウインドウを開く

説明

AJP SET KEY INPUT MODEは入力メソッドの状態を引数で指定したものに設定します。

引数に0を指定した場合、FEPを通さずに入力するモードとなります。コードの入力時など、引数に4を指定して英数字モードにするよりも、FEPを通さない方をユーザが好む場合があります。

また、-1がIME MODEのロック(IMEを閉じる)、10がクローズしたIMEの再表示となります。

Macintoshの場合

Macintoshでは通常入力メソッドのオン・オフは日本語入力と英語入力の切り替えを意味します。このコマンドを使用して入力モードを日本語と英語の間で切り替えることができます。

ただしこのコマンドは入力メソッド内部の設定を行いません。例えば入力メソッドをオン（日本語モード）にした時に英数入力モードとなるか、かな入力モードとなるかを設定することはできません。これは以前の設定が採用されます。

Windowsの場合

Windowsではこのコマンドにより、入力モードを細かく設定することが可能です。

参照

AJP Get key input mode

文字列操作ユーティリティ（AJ_Pack : Charbase String）

文字列操作ユーティリティは4Dのなかで２バイト文字を含んだ文字列操作を可能にします。

『4th Dimensionランゲージリファレンス』の文字列関数の章で説明されている関数では、文字の位置や文字数を引数に渡す際にバイトを単位とする必要があります。

この章で説明する関数を使用することにより文字単位の文字列操作が可能となり、２バイト文字を含む文字列へのアクセスが容易になります。

注：文字列操作のルーチンは4th Dimensionバージョン6.5以降となります。

AJP sub characters

AJP sub characters (文字列;先頭文字位置;文字数;部分文字列) → エラー

引数	タイプ		説明
文字列	テキスト	→	この文字列から部分文字列を得ます
先頭文字位置	整数	→	取り出す文字列の最初の文字の位置
文字数	整数	→	取り出す文字列の長さ
部分文字列	テキスト	←	文字列の部分文字列
戻り値	倍長整数	←	エラーコード

説明

AJP sub charactersは4D関数の**Substring**と同じ働きをするものです。この関数は文字列の先頭文字位置から文字数分の文字を取り出し、部分文字列に返します。

文字列には部分文字列を得るための元の文字列を渡します。文字列にはフィールドや変数を渡すことができます。リテラル文字列を渡すことはできません。

先頭文字位置には取り出す文字列の最初の文字の位置を文字数で数えた整数値で渡します。文字数には取り出す文字列の長さ文字数で数えた整数値で渡します。つまり1バイト文字も2バイト文字もそれぞれ1文字として数えます。

部分文字列には変数を渡します。フィールドを渡すことはできません。

エラーコード

文字列取出が成功した場合はエラーコードに0が返ります。

注：**Substring**コマンドと違い、文字数引数を省略することはできません。

例：

次のコードでは、変数Base_Stringに「今日は天気です。」と代入されています。

この戻り値、変数Result_Stringには「は天気で」が返されます。

```
$Error:=AJP sub characters (Base_String;3;4;Result_String)
```

参照

なし

AJP characters length

AJP characters length (文字列) → 整数

引数	タイプ		説明
文字列	テキスト	→	この文字列から部分文字列を得ます
戻り値	倍長整数	←	文字列の長さ

説明

AJP characters length は4D関数の**Length**と同じ働きをするものです。この関数は文字列の長さを、文字数単位で返します。

文字列には文字数を得るための文字列を渡します。文字列にはフィールドや変数を渡すことができます。リテラル文字列を渡すことはできません。

文字列の長さには文字列の長さが文字数単位で返ります。つまり1バイト文字も2バイト文字もそれぞれ1文字として数えます。

例：

次のコードでは、変数Base_Stringに「今日は天気です。」と代入されています。

この戻り値は8となります。

```
$String_Length:=AJP characters length (Base_String)
```

参照

なし

AJP Change characters

AJP Change characters(元の文字列;修正文字列;修正位置;結果文字列)→エラー

引数	タイプ		説明
元の文字列	テキスト	→	元の文字列
修正文字列	テキスト	→	新しい文字列
修正位置	整数	→	修正開始位置
結果文字列	テキスト	←	結果文字列
戻り値	倍長整数	←	エラーコード

説明

AJP Change characters は4D関数の**Change string**と同じ働きをするものです。この関数は元の文字列の中の文字グループを修正したものを返します。修正位置で指定された位置から、修正文字列で元の文字列を上書きします。

文字数のカウントはすべて文字単位で行われます。例を参照してください。

エラーコード

文字列修正が成功した場合はエラーコードに0が返ります。

例：

次のコードでは、変数Base_Stringに「今日は天気です。」と代入されています。

AJP Change characters はBase_Stringの4文字目、”a”からはじめて、”とっても”に該当する文字数分”abcd”を”とっても”と置き換えます。結果vTextには「今日とっても天気です。」が返されます。

```
$Error:=AJP Change characters (Base_String;”とっても”;4;vText)
```

参照

なし

AJP Insert characters

AJP Insert characters (元の文字列;挿入文字列;挿入位置;結果文字列) → エラー

引数	タイプ		説明
元の文字列	テキスト	→	元の文字列
挿入文字列	テキスト	→	新しい文字列
挿入位置	整数	→	挿入開始位置
結果文字列	テキスト	←	結果文字列
戻り値	倍長整数	←	エラーコード

説明

AJP Insert charactersは4D関数の**Insert string**と同じ働きをするものです。この関数は元の文字列に文字列を挿入したものを返します。挿入位置で指定された位置に、挿入文字列を挿入します。

元の文字列と挿入文字列にはフィールドや変数を渡すことができます。リテラル文字列を渡すことはできません。結果文字列には変数を渡します。フィールドを渡すことはできません。文字数のカウントはすべて文字単位で行われます。

エラーコード

文字列修正が成功した場合はエラーコードに0が返ります。

例：

次のコードでは、変数Base_Stringに「今日は天気です。」と代入されています。

AJP Insert charactersはBase_Stringの4文字目に”とっても”を挿入します。結果vTextには「今日とっても天気です。」が返されます。

```
$Error:=AJP Insert characters (Base_String;”とっても”;4;vText)
```

参照

なし

AJP Delete characters

AJP Delete characters (元の文字列;削除位置;文字数;結果文字列) → エラー

引数	タイプ		説明
元の文字列	テキスト	→	元の文字列
削除位置	整数	→	削除を開始する文字の位置
文字数	整数	→	削除する文字数
結果文字列	テキスト	←	結果文字列
戻り値	倍長整数	←	エラーコード

説明

AJP Delete charactersは4D関数の**Delete string**と同じ働きをするものです。この関数は元の文字列より、削除位置から文字数分の文字を削除した文字列を返します。

元の文字列にはフィールドや変数を渡すことができます。リテラル文字列を渡すことはできません。

結果文字列には変数を渡します。フィールドを渡すことはできません。

文字数のカウントはすべて文字単位で行われます。

エラーコード

文字列削除が成功した場合はエラーコードに0が返ります。

例：

次のコードでは、変数Base_Stringに「今日は天気です。」と代入されています。

AJP Delete charactersはBase_Stringの4文字目から4文字を削除します。結果vTextには「今日天気です。」が返されます。

```
$Error:=AJP Delete characters (Base_String;4;4;vText)
```

参照

なし

AJP Characters Position

AJP Characters Position (元の文字列;検索文字列) → 結果

引数	タイプ		説明
元の文字列	テキスト	→	元の文字列
検索文字列	テキスト	→	探す文字列
戻り値	倍長整数	←	最初の発見位置

説明

AJP Characters Positionは4D関数の**Position**と同じ働きをするものです。この関数は元の文字列の中で検索文字列が最初に表示される位置を返します。

元の文字列や探す文字列にはフィールドや変数を渡すことができます。リテラル文字列を渡すことはできません。文字数のカウントはすべて文字単位で行われます。

検索文字列が見つからない場合、戻り値には0（ゼロ）を返します。処理を実行するために必要なメモリが確保できなかった場合や引数が異常（典型的なケースとしては引数が空文字列）であった場合には-1を返します。

“@”の扱い

AJP Characters Position関数は”@”を一文字のワイルドカードと見なします。

注：**Position**関数とは引数リストの順序が逆です。第1引数が元の文字列で第2引数が検索文字列となっています。

例：

次のコードでは、変数Base_Stringに「今日は天気です。」と代入されています。

AJP Characters PositionはBase_Stringの中から「とっても」が初めに表示される位置を文字数で返します。結果$Posには4が返ります。

```
$Pos:=AJP Characters Position (Base_String;4;4;vText)
```

注：

Position関数とは引数リストの順序が逆です。第1引数が元の文字列で第2引数が検索文字列となっています。

また、検索文字列が見つからない場合の戻り値は、0だけでなく-1も考慮する必要があります。したがって、文字検索が失敗したかどうかの検出は以下のように1未満の戻り値を失敗と判定するように記述する必要があります。

```
If ( AJP Characters Position ( $元の文字列 ; $検索文字列 ) < 1 )
        ` 見つからない
Else
        ` 見つかった
End if
```

参照

なし

コマンド索引

S

U