Cisco Unity Express Script Editor の概要

Cisco Unity Express Script Editor は、自動受付アプリケーション スクリプトを作成するための視覚的なプログラミング環境です。Cisco Unity Express Script Editor は、Microsoft Windows ソフトウェアが動作している任意の PC で使用できます。

（注） 以前に Cisco Customer Response Applications Developer（CRA）Editor アプリケーションをインストールしたハードウェアに、Cisco Unity Express Script Editor アプリケーションをインストールしないでください。これらのアプリケーションはレジストリ ファイルを共有するため、同一のハードウェアにインストールすると機能しません。

この章では、Cisco Unity Express Script Editor アプリケーションの概要（ウィンドウやペインなど）について説明し、スクリプト要素の使用方法に関する情報を提供します。次の各トピックについて説明します。

Cisco Unity Express スクリプトの作成

Cisco Unity Express Script Editor を使用すると、自動受付アプリケーションに到達する通話を処理するスクリプトの作成と検証ができます。スクリプトはステップと変数で構成されます。

ステップは、応対処理ロジックの Java ベース ブロックです。Cisco Unity Express スクリプトを作成するために、Java プログラミングを理解する必要はありません。

各ステップには、プロンプトの簡単な増分、生成、再生や、ユーザ入力の取得など、独特な固有の機能があります。

ステップは個別に動作するか、または出力ブランチと呼ばれるサブプロシージャとしてグループ化して動作できます。出力ブランチはメイン プロシージャに値を返します。

機能を完成するために変数が必要なステップも、追加情報が必要ないステップもあります。

変数は、スクリプトの処理に影響を与えるデータを提供します。データ ソースはユーザ入力か、ほかのスクリプトのステップやプロシージャで計算された値になります。

スクリプトの作成には、次の作業が含まれます。

- ステップを適切な順序に構成する。

  ワークスペースの左ペインにあるパレットからワークスペースの右ペインにある設計領域に、ステップ アイコンをドラッグします。必要に応じてステップの追加、削除、並べ替えができます。

  Cisco Unity Express Script Editor は、ステップの連結に必要なコードを提供します。
- ステップに必要な変数定義とその他のパラメータを指定する。
- 完成したスクリプトを Cisco Unity Express Script Editor で直接検証する。

Cisco Unity Express Script Editor ウィンドウの使用方法

Cisco Unity Express Script Editor を起動するには、次のパスを使用します。

Start > Programs > Cisco CUE Developer > Cisco CUE Editor

図 1 に示すように、Cisco Unity Express Script Editor ウィンドウが表示されます。

図 1　　Cisco Unity Express Script Editor ウィンドウ

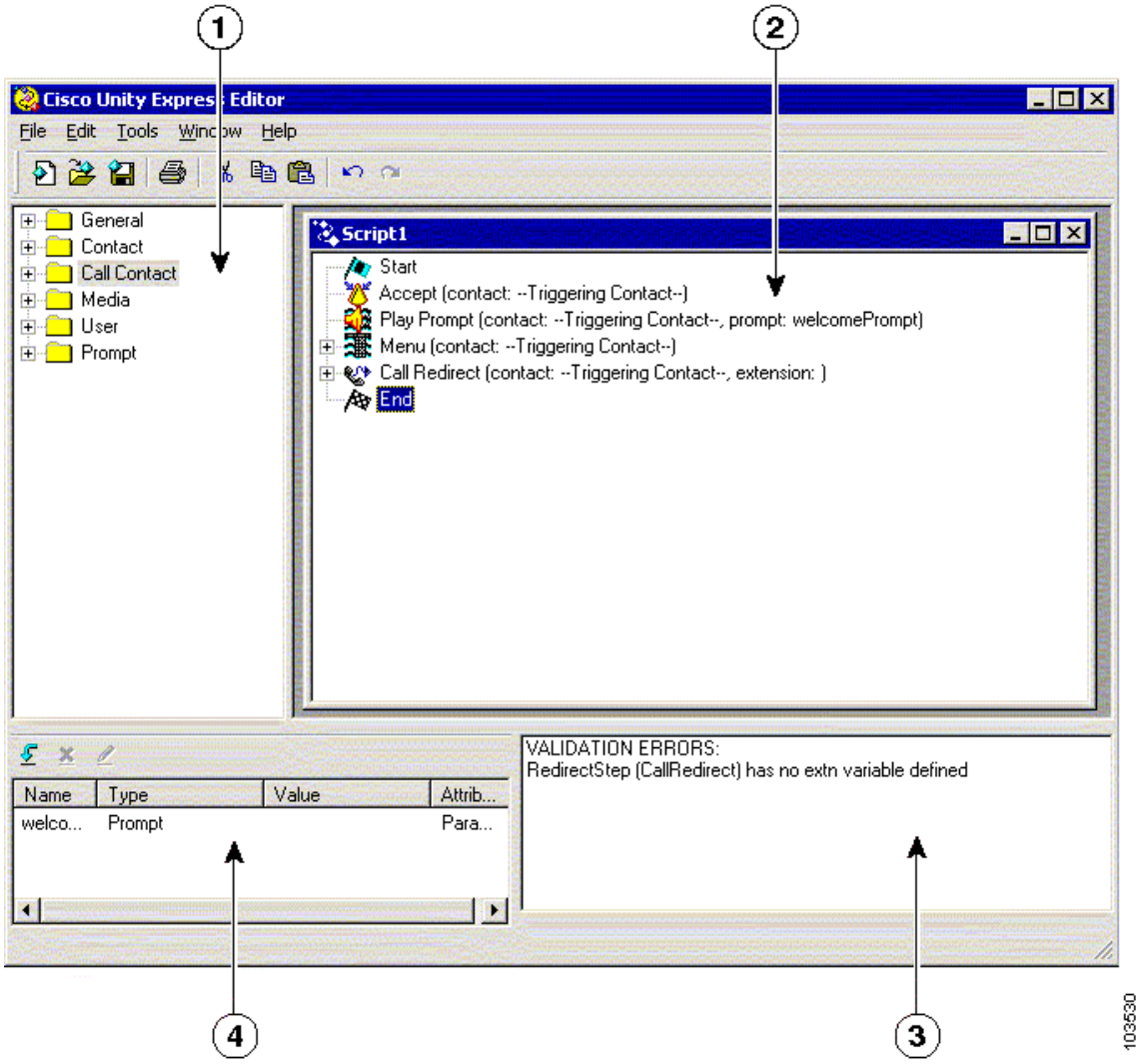


表 1 に、Cisco Unity Express Script Editor ウィンドウの 4 つのペインを示します。

表 1　Cisco Unity Express Script Editor ウィンドウ

ペイン番号	ペイン名	説明
1	Palette ペイン	**Palette** ペインは、スクリプトの作成に必要なステップを選択する場合に使用します。詳細については、P.11 の「Palette ペインの使用方法」を参照してください。
2	Design ペイン	**Design** ペインは、スクリプトを作成する場合に使用します。詳細については、P.11 の「Design ペインの使用方法」を参照してください。
3	Debug ペイン	**Debug** ペインは、スクリプト検証時にメッセージを表示する場合に使用します。詳細については、P.20 の「Debug ペインの使用方法」を参照してください。
4	Variable ペイン	**Variable** ペインは、スクリプトの変数を作成、修正、および表示する場合に使用します。詳細については、P.15 の「Variable ペインの使用方法」を参照してください。

メニュー バーの使用方法

次の各項では、Cisco Unity Express Script Editor ウィンドウのメニュー バー オプションについて説明します。

File メニュー オプション

File メニュー オプションは、表 2 に示すように、ファイルに関連するさまざまな操作を実行する場合に使用します。

表 2　File メニュー オプション

オプション	説明
New	新しいスクリプトを作成し、Design ペインに Start ステップを配置します。 **Start** ステップはすべての新規スクリプトの最初のステップです。
Open	既存のスクリプト（.aef）ファイルを選択して開くことができる、標準の Open ウィンドウを表示します。 **Open** オプションは、既存のスクリプトを修正する場合に使用します。
Close	現在のスクリプト ファイルを閉じます。
Save	現在のスクリプト ファイルを保存します。
Save As	標準の Save As ウィンドウを開きます。このウィンドウで、ファイル名と拡張子 .aef を入力して、現在のスクリプトを保存できます。
Print	現在のファイルを印刷します。

表 2　File メニュー オプション（続き）

オプション	説明
Properties	次の 2 つのタブがあります。 • **General**：開いているファイルのタイプ、場所、およびサイズを示します。 • **Summary**：開いているファイルの説明情報を入力できるフィールドを示します。

Edit メニュー オプション

Edit メニュー オプションは、表 3 に示すように、さまざまな編集操作を実行する場合に使用します。

表 3　Edit メニュー オプション

オプション	説明
Undo	最後の操作を元に戻します。
Redo	最後の操作をやり直します。
Cut	選択した項目を切り取ります。
Copy	選択した項目をコピーします。
Paste	選択した項目を貼り付けます。

Tools メニュー オプション

Tools メニュー オプションにあるのは、**Validate** のみです。

Validate メニュー オプションは、スクリプトの順序およびステップ プロパティの使用方法が、Cisco Unity Express Script Engine で必要な一般的な構文に準拠していることを確認する場合に使用します。

Window メニュー オプション

Window メニュー オプションは、表 4 に示すように、複数のファイルを Design ペインに表示する方法を制御する場合に使用します。

表 4　Window メニュー オプション

オプション	説明
Cascade	ウィンドウを重ねた状態でファイルを表示します。
Tile Horizontally	同じ大きさのウィンドウを横に並べてファイルを表示します。
Tile Vertically	同じ大きさのウィンドウを縦に並べてファイルを表示します。

Help メニュー オプション

Help メニュー オプションは、表 5 に示すように、Cisco Unity Express Script Editor 画面、フィールド、およびソフトウェアの、詳細な情報を表示する場合に使用します。

表 5　Help メニュー オプション

オプション	説明
Help	Cisco Unity Express Script Editor の画面とフィールドに関する情報を表示します。
About	Cisco Unity Express Script Editor ソフトウェアのバージョン番号を表示します。

ツールバーの使用方法

Cisco Unity Express Script Editor ツールバーには、メニュー バーの一部のオプションと同じオプションを選択できるアイコンがあります。図 2 を参照してください。

図 2　Cisco Unity Express Script Editor ツールバー

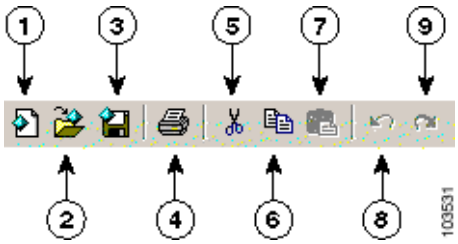


表 6 に、図 2 のツールバーの、番号が付けられた各ツールの機能を示します。

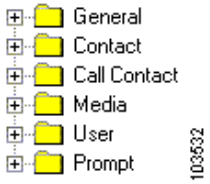
表 6　Cisco Unity Express Script Editor ウィンドウのツールバーの機能

ツール番号	説明
1	スクリプトを作成します。
2	スクリプトを開きます。
3	スクリプトを保存します。
4	選択したファイルを印刷します。
5	選択した項目を切り取ります。
6	選択した項目をコピーします。
7	選択した項目を貼り付けます。
8	前のコマンドを元に戻します。
9	前のコマンドをやり直します。

Palette ペインの使用方法

Cisco Unity Express Script Editor の **Palette** ペインには、スクリプトの作成に使用できるすべてのステップが含まれています。図 3 を参照してください。

図 3　Cisco Unity Express Script Editor の Palette ペイン



各 Palette ペイン カテゴリのステップの詳細については、次の各項を参照してください。

- General ステップ（P.74）
- Contact ステップ（P.93）
- Call Contact ステップ（P.99）
- Media ステップ（P.104）
- User ステップ（P.132）
- Prompt ステップ（P.136）

パレットの内容を表示するには、Palette ペインでパレット アイコンの左側にあるプラス記号（+）をクリックします。

スクリプトを作成するには、**Palette** ペインでステップをクリックし、**Design** ペインで後続のステップの最上部にドラッグします。各ステップにより、特定の機能が実行され、基礎となるプログラミングの一部分が作成されます。ほとんどのステップは、**Design** ペインに配置した後でカスタマイズできます。

Cisco Unity Express Script Editor で許可されている場所にステップを移動するまで、カーソルには国際的な禁止記号が表示されます。

（注）　カスタマイザ ウィンドウを開いているときに、ステップを **Design** ペインにドラッグしようとしても、**Design** ペインはステップを受け入れません。**Design** ペインにステップをドラッグする前に、開いているカスタマイザ ウィンドウをすべて閉じてください。Cisco Unity Express Script Editor ウィンドウの後ろに 1 つまたは複数のカスタマイザ ウィンドウが隠れている場合があります。

Design ペインの使用方法

Cisco Unity Express Script Editor の **Design** ペインでは、スクリプト開発の最も多くの部分を実行します。

スクリプトの開始

Cisco Unity Express Script Editor ウィンドウの **Design** ペインで新しいスクリプトを開始するには、**File > New** を選択します。Cisco Unity Express Script Editor によって Start ステップが自動的に追加され、ステップを追加する開始点が示されます。

ステップの追加

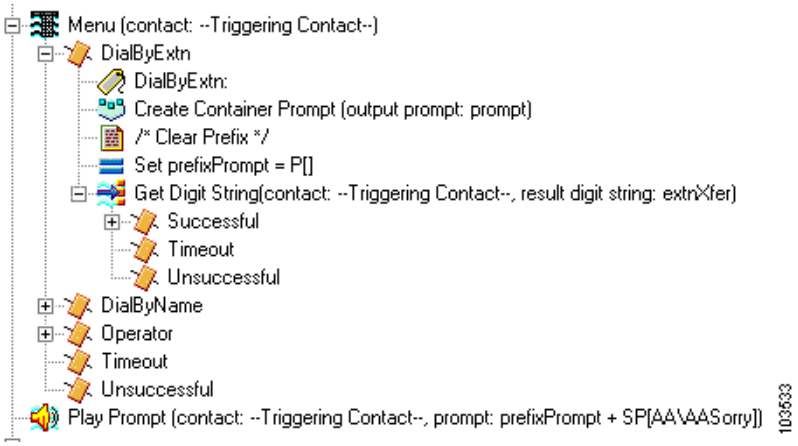
ステップをスクリプトに追加するには、**Palette** ペインからステップ アイコンをドラッグして、**Design** ペインで後続のステップの上にドロップします。作成するスクリプトの論理的な順序でステップを配置します。

スクリプト内のステップの順序を変更するには、個々のステップ アイコンを変更前の場所から変更先の場所へドラッグします。ステップを削除するには、ステップ アイコンを選択して **Del** キーを押します。

スクリプトを終了するには、**General** パレットをクリックして **End** をスクリプトにドラッグします。**End** ステップが表示されます。

図 4 に、**Design** ペインでのスクリプトの表示例を示します。

図 4　　Design ペインでのスクリプト例



多くのステップには出力ブランチがあります。このブランチの下にステップを追加し、ステップの終了条件に基づいて必要なスクリプト ロジックを指定します。

上に示す図 4 の場合、たとえば **Menu** ステップには次の 5 つの出力ブランチがあります。

- DialByExtn
- DialByName
- Operator
- Timeout
- Unsuccessful

出力ブランチには、多くの場合、ステップとその他の出力ブランチが含まれます。図 4 で、たとえば **DialByExtn** 出力ブランチにはその下に 5 つのステップがあり、そのうちの 1 つ（**Get Digit String** ステップ）には 3 つの出力ブランチがあります。

ステップの下のスクリプトを展開するには、ステップ アイコンの左側にあるプラス記号（+）をクリックします。ステップの下のスクリプトを閉じるには、ステップ アイコンの左側にあるマイナス記号（-）をクリックします。

実行時には、各スクリプトはステップを結ぶ縦線で示される階層の順序に従います。図 4 で、たとえばスクリプトが **Get Digit String** ステップの **Timeout** 出力ブランチに達すると、スクリプトは **Menu** ステップのレベルで次のステップ、つまり、この例では **Play Prompt** ステップに移動します。

ステップのカスタマイズ

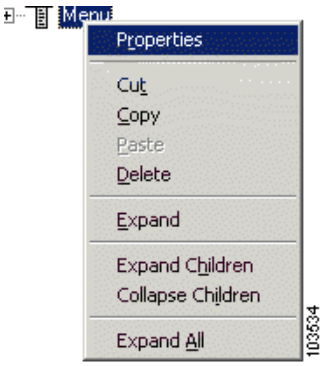
Cisco Unity Express Script Editor のほとんどのステップは、カスタマイザ ウィンドウというウィンドウを開いてカスタマイズできます。カスタマイザ ウィンドウにはプロパティというテキスト フィールドがあり、スクリプトの必要を満たすようにこのフィールドを設定できます。ステップに応じて、プロパティは複数のタブにグループ化されることがあります。

ステップのカスタマイザ ウィンドウを表示するには、次の手順を実行します。

ステップ 1 **Design** ペインで、カスタマイズするステップを右クリックします。

Properties ポップアップ メニューが表示されます。図 5 に **Menu** ステップの **Properties** ポップアップ メニューの例を示します。

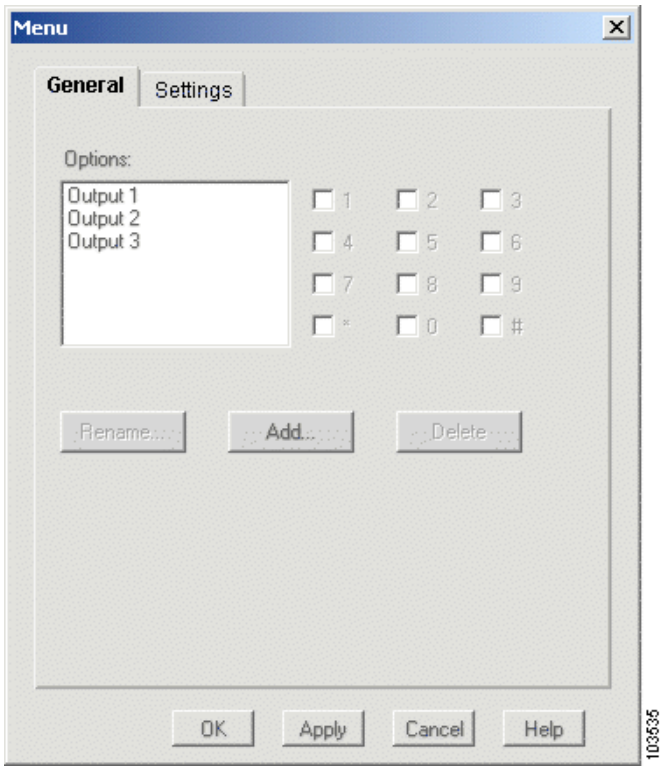
図 5　Menu ステップの Properties ポップアップ メニュー



ステップ 2 **Properties** をクリックします。

ステップのカスタマイザ ウィンドウが表示されます。図 6 に Menu カスタマイザ ウィンドウの例を示します。

図 6　　Menu カスタマイザ ウィンドウ



ステップ 3　ステップをカスタマイズするための適切なデータを入力します。各ステップに対応するカスタマイザ ウィンドウのプロパティについては、このガイドの後半にある各ステップの説明を参照してください。

各カスタマイザ ウィンドウには、次の 4 つのボタンがあります。

- **OK**：変更を適用し、カスタマイザ ウィンドウを閉じる。
- **Apply**：変更を適用し、カスタマイザ ウィンドウは閉じない。
- **Cancel**：変更を適用せずにカスタマイザ ウィンドウを閉じる。
- **Help**：このステップの文脈依存ヘルプを表示する。

カスタマイザ ウィンドウには、ステップ内のさまざまなプロパティを修正および表示するために使用する、追加のボタンが用意されている場合もあります。

ステップ 4　ステップの変更が終了したら、該当するボタンをクリックします。

Variable ペインの使用方法

Cisco Unity Express Script Editor の **Variable** ペインでは、スクリプトで使用する変数の追加と修正を行います。P. 15 の図 7 を参照してください。変数は、スクリプトがステップの実行時に使用するデータを格納します。変数は Cisco Unity Express Script Editor ウィンドウの **Variable** ペインで定義した後、スクリプト内のすべてのステップで使用できます。P. 7 の図 1 を参照してください。

また、サブフローで定義した変数に対してスクリプト用に定義した変数をマップすることもできます。サブフローとは、プライマリ スクリプトという別のスクリプトの一部として機能するステップのセットです。サブフローでは変数を使用して操作した後、その変数に格納されたデータをプライマリ スクリプトに返すことができます。スクリプトはほかのスクリプトと変数を共有できません。ただし、プライマリ スクリプトが変数の値をデフォルト スクリプトに自動転送するデフォルト スクリプトの場合は除きます。

変数の値は実行中に変化することがあります。

この項では、次の各トピックについて説明します。

- 変数の定義（P.15）
- 組み込み基本変数の型の使用方法（P.16）
- パラメータを使用した変数のエクスポート（P.18）
- 組み込み拡張変数の型の使用方法（P.19）

変数の定義

新しい変数を定義するには、Cisco Unity Express Script Editor ウィンドウの **Variable** ペインで、左上隅にある **New Variable** アイコンをクリックします。**Edit Variable** ウィンドウが表示されます。図 7 を参照してください。

図 7　　Variable ペインと Edit Variable ウィンドウ

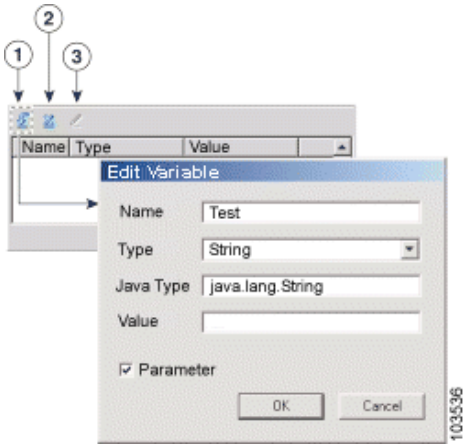


表 7 に、**Variable** ペインの各ツールの機能を示します。

表 7　Variable ペインのツールバーの機能

ツール番号	説明
1	New Variable アイコン
2	Delete Variable アイコン
3	Modify Variable アイコン

Edit Variable ウィンドウを使用して変数を定義した後、変数が **Variable** ペインに表示されます。変数を選択し、**Modify** アイコンまたは **Delete** アイコンを使用して必要な変更を加えることができます。

表 8 に、**Edit Variable** ウィンドウのフィールドを示します。

表 8　Edit Variable のプロパティ

プロパティ	説明
Name	宣言する変数の名前。
Type	宣言する変数の型。使用可能な変数の型については、P.16 の「組み込み基本変数の型の使用方法」を参照してください。
Java Type	コンピュータの CLASSPATH 環境変数を使用して検索される完全修飾クラス名。 （注）　このフィールドには、**Type** ドロップダウン メニューで選択した組み込みデータ型に対応する実際の Java 型が表示されます。
Value	最初に変数に割り当てるデータ。入力するデータ型は、**Type** フィールドで宣言したデータ型と一致している必要があります。
Parameter	オンにした場合、このスクリプトを使用するアプリケーションを準備するときに、このパラメータの値が自動受付 Web インターフェイスで設定されます。

組み込み基本変数の型の使用方法

表 9 に、組み込み基本変数の型を示します。

表 9　変数の型

変数の型	説明	Java クラス名	変数の入力形式
Boolean	Boolean 変数は true または false のいずれかで、主に Cisco Unity Express Script Editor の **General** パレットの **If** ステップで使用されます。	java.lang.Boolean	• t、f • true、false
Character	Character 変数には英数字が含まれます。	java.lang.Character	• 小文字の a ～ z • 大文字の A ～ Z • 数字の 0 ～ 9 • エスケープ シーケンスの “\t”、“\r”、“\0”、“\n”、“\f”、“\\”、“\” • “\uXXXX” により、16 進数 Unicode XXXX を使用する文字の表記が可能です。

表 9　変数の型（続き）

変数の型	説明	Java クラス名	変数の入力形式
Float	Float 変数には 10 進数が含まれます。	java.lang.Float	• 3.14159 • 2E-12 • -100
Integer	Integer 変数には、数値 -2147483648 ～ 2147483647 が含まれます。	java.lang.Integer	• 234556789 • 0 • -23
String	String 変数には、Unicode 文字 “\u0000” ～ “\uffff” のセットが含まれます。	java.lang.String	• “Hello”、“C:\WINNT\win.ini”: この形式ではエスケープ文字も Unicode 文字もサポートされません。 • u“\”This is a quoted string\””、u“\tHello”、u“\u2222\u0065”、u“C:\\WINNT\\win.ini” など：この形式では Character 型で説明したものと同じエスケープ シーケンスまたは Unicode 文字がサポートされます。
Date	Date 変数には日付情報が含まれます。	java.util.Date	• D[12/13/03] • D[Dec 13, 2003] • D[January 20, 2003] • D[Tuesday, April 12, 2003] • D[12/13/03] • D[12/13/03 5:50 PM] • D[April 1, 2003 12:00:00 AM PST] D[] の中に指定したパラメータは、次の 2 つの形式による組み合せに基づいて解析されます。 • “<date>” • “<date> <time>” Cisco Unity Express Script Editor では次の 4 つの <date> 指定形式がサポートされています。 • SHORT。すべて数字の形式。“12/13/03” など。 • MEDIUM。やや長い形式。“Jan 12, 2003” など。 • LONG。より長い形式。“January 12, 2003” など。 • FULL。完全に指定する形式。“Tuesday, April 12, 2003” など。

表 9　変数の型（続き）

変数の型	説明	Java クラス名	変数の入力形式
Time	Time 変数には時刻情報が含まれます。	java.sql.Time	• T[3:39 AM] • T[11:59:58 PM EST] T[] の角カッコの中に指定したパラメータは、形式 “<time>” に基づいて解析されます。 Cisco Unity Express Script Editor では次の 3 つの <time> 指定形式がサポートされています。 • SHORT。短い形式。“3:30 PM” など。 • MEDIUM。長い形式。“3:30:32 PM” など。 • LONG および FULL （同一）。より完全な形式。“3:30:42 PM PST” など。
BigDecimal	BigDecimal 変数には、スケールを持つ任意の精度の整数が含まれます。スケールは小数点の右側の桁数です。	java.math.BigDecimal	• 3.14159 • 2E-12 • -100
BigInteger	BigInteger 変数は、任意の精度の整数を表します。	java.lang.BigInteger	• 234556789 • 0 • -23
Double	Double 変数は、拡張 Float 変数を表します。	java.lang.Double	• 3.14159 • 2E-12 • -100
Long	Long 変数は、拡張 Integer 変数を表します。	java.lang.Long	• 234556789 • 0 • -23

パラメータを使用した変数のエクスポート

Edit Variables ダイアログボックスで **Parameter** チェックボックスをオンにすると、変数をパラメータとして宣言できます。

この機能によって、パラメータの値を自動受付 Web インターフェイスに設定できます。値は、値を使用するスクリプトの設定時に初期化されるため、Cisco Unity Express Script Editor でスクリプトを編集することなく、値を変更できます。このような変数を*エクスポートされた*変数と呼びます。

たとえば、新しい自動受付を AA ウィザードを使用して追加する場合、AA ウィザードの 2 番目のウィンドウ（**Script Parameters** ウィンドウ）に、パラメータがデフォルト値または現在の値と共にリストで示されます。このリストの値は変更できます。

Cisco Unity Express がパラメータについてサポートする変数の型には、**Number**、**Character**、**String**、**Boolean**、および **Prompt** があります。

組み込み拡張変数の型の使用方法

Cisco Unity Express Script Editor には、次の組み込み拡張変数の型があります。

Contact 変数

Java クラス名：com.cisco.contact.Contact

Contact 変数には、通話呼を示す連絡先が含まれます。

Contact 変数はパラメータとしてサブフローに渡すことができます。

Prompt 変数

Java クラス名：com.cisco.prompt.Playable

Prompt 変数には、発信者に対して再生するプロンプトを作成するためにスクリプトで使用する情報が含まれます。Prompt 変数は、1 つのプロンプトのように単純な場合、または連結された複数のプロンプトのように複雑な場合があります。

次の入力形式を使用して、Prompt 変数を定義できます。

- P[] または SP[]：空のプロンプトを表します。プロンプトは再生されません。
- P[AA\AAWelcome]、P[AA\AAWelcome.wav]、P[prompt2] など：User Prompts ディレクトリにあるユーザ定義プロンプトを表します。PC またはサーバの User Prompt ディレクトリに .wav ファイルが格納されています。
- SP[dialtone]、SP[gen\char\e.wav] など：System Prompts ディレクトリにあるシステム定義プロンプトを表します。
- DP[250]、DP[500] など：ミリ秒数で指定した遅延プロンプトを表します。
- S[Cisco] など：プロンプトが再生される連絡先の現在の言語規則に従って、指定されたテキストを明確に述べるプロンプトを表します。
- N[22.33]、N[-2E-23]、N[-1] など：プロンプトが再生される連絡先の現在の言語規則に従って、指定された番号の音声表現となるプロンプトを表します（たとえば「twenty-two point three three」）。
- #[1]、#[22] など：プロンプトが再生される連絡先の現在の言語規則に従って、指定された序数の音声表現となるプロンプトを表します（たとえば「first」、「twenty-second」）。
- $[12]、$[32.50]：プロンプトが再生される連絡先の現在の言語規則に従って、スクリプトが指定金額をシステム設定通貨の口語形式で再生するプロンプトを表します（たとえば「twelve dollars」、「thirty-two francs and fifty centime」）。

その他の拡張変数と同様に、Prompt 変数の式の入力形式は、変数の入力形式と同じです。

- S[] の角カッコの中に指定したパラメータを、文字列式の結果にすることもできます（たとえば S[lastName + firstName]）。
- DP[]、N[]、#[]、および $[] の角カッコの中に指定したパラメータを、数式の結果にすることもできます（たとえば DP[delay]、N[counter + 3]、#[position]、$[amount + 10.00]）。

Expression Editor で演算子を使用するときに、そのオペランドのうち少なくとも 1 つが Prompt である場合、2 つのオペランドによる 1 つの Prompt 連結となります。この場合、他方のオペランドは次の規則に従って、Prompt に変換されます。

- システムは Spelling ジェネレータ（S[ ] 形式と同様）または Character ジェネレータを使用して String of Character オペランドを Prompt に変換します。ジェネレータ型については、P.146 の「Spelling ジェネレータ」および P.145 の「Number ジェネレータ」を参照してください。
- システムは Number ジェネレータ（N[ ] 形式と同様）を使用して Number オペランドを Prompt に変換します。ジェネレータ型については、P.145 の「Number ジェネレータ」を参照してください。

- システムは Time ジェネレータ（P.147 の「Time ジェネレータ」を参照）を使用して Time オペランドを Prompt に変換します。
- システムは Date ジェネレータ（P.146 の「Date ジェネレータ」を参照）を使用して Date オペランドを Prompt に変換します。

User 変数

Java クラス名：com.cisco.user.User

User 変数には、ユーザ認証に役立つ情報が含まれます。

User 変数を値として手動で入力することはできません。User 変数を返すことができるのは、**Media** パレットの **Name To User** ステップだけです。

User 変数はパラメータとしてサブフローに渡すことができます。

Debug ペインの使用方法

Debug ペインは、スクリプトを検証する場合に使用します。

スクリプトを検証するときに、このペインに **Validation Error** メッセージが表示されます。エラーメッセージをダブルクリックすると、検証に失敗したスクリプト ステップに移動します。

プロンプトの使用方法

Cisco Unity Express Script Editor では、次の 2 種類のプロンプトを使用します。

- システム プロンプト：シスコのモジュールおよびシスコのサンプル スクリプトによって内部で使用されます。

（注） システム プロンプトはシステムによって内部で使用されます。将来のリリースで、すべてのシステム プロンプトを継続して使用できることを保証するものではありません。

- ユーザ プロンプト：ユーザが定義するプロンプトです。管理者が Cisco Unity Express GUI 管理者インターフェイスの **Voice Mail > Prompts** Web ページを使用するか、Greeting Management System（GMS; グリーティング管理システム）を呼び出すことによって管理できます。

すべての **Media** ステップおよび **Prompt** ステップでは、次の方法で指定されたプロンプトをサポートします。

- 文字列式：User Prompts ディレクトリにあるユーザ定義プロンプト。
- プロンプト式：実行時に動的に作成される。

（注） タイプが WAVE および G711 u-law 形式の RIFF ヘッダーを付けて、再生および録音されるすべてのプロンプトを定義する必要があります。

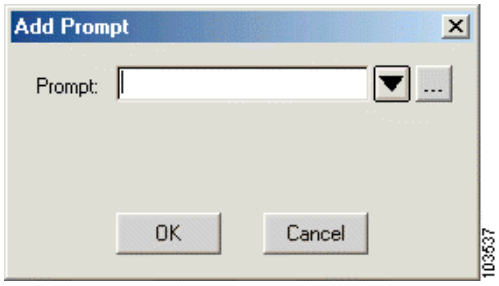
スクリプトはユーザ プロンプトとシステム プロンプトの両方を Prompt Repository から取得します。これらのプロンプトは、Cisco Unity Express GUI 管理者インターフェイスの **Voice Mail > Prompts** Web ページから、またはグリーティング管理システムを呼び出すことによって管理できます。プロンプト管理の詳細については、システムの Cisco Unity Express GUI および CLI 管理のドキュメントを参照してください。ドキュメントは http://www.cisco.com/univercd/cc/td/doc/product/voice/unityexp/index.htm から入手できます。

式の使用方法

設計時に正確な値がわからないときに、実行時に評価可能な数式を代わりに入力する場合、式が役立ちます。式の結果の型は予期した入力タイプ（設計時に確認した型）と一致する必要があります。

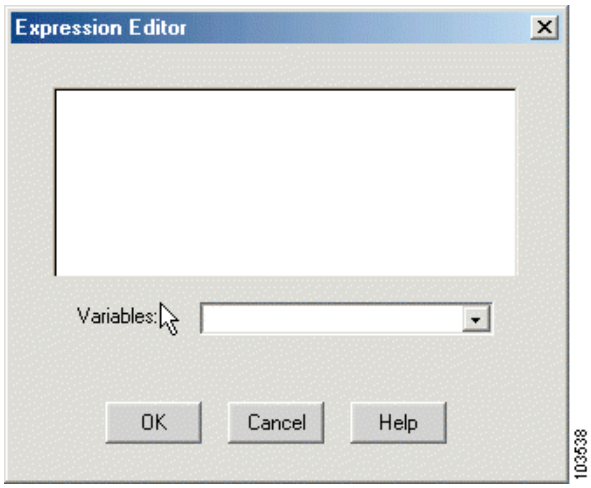
多くのステップのカスタマイザ ウィンドウには **Expression Editor (...)** ボタンがあり、式の入力に使用できます。図 8 を参照してください。

図 8　　カスタマイザ ウィンドウの Expression Editor ボタンの例



入力テキスト フィールドに直接式を入力できます。また、**Expression Editor (...)** ボタンをクリックして Expression Editor を開くことができます。図 9 を参照してください。

図 9　　Expression Editor



テキスト フィールドに式を入力できます。また、**Variable** ドロップダウン メニューを使用すると、スクリプトで事前に定義した変数にすぐアクセスできます。**Variable** ドロップダウン メニューから変数を選択すると、変数名が入力テキスト フィールドに表示されます。

式を入力したら、**OK** をクリックします。Expression Editor が閉じます。

式の入力形式の使用方法

次の表に、各式の型に対応する式の入力形式を示します。

表 10　式の入力形式

式の型	説明	式の入力形式
Boolean	Boolean 式は true または false のいずれかになります。	• true • false
Character	Character 式には英数字が含まれます。	• 小文字の a ～ z • 大文字の A ～ Z • 数字の 0 ～ 9 • エスケープ シーケンスの “\t”、“\r”、“\0”、“\n”、“\f”、“\\”、“\” • “\uXXXX” により、16 進数 Unicode XXXX を使用する文字の表記が可能です。
Float	Float 式には 10 進数が含まれます。 スクリプトはすべての Float 値を Double 表現で保持し、値を Float 変数に割り当てるときにだけ型キャストを行います。この機能により、Java における Float 値の格納方法によって精度が失われることを防ぐことができます。 スクリプトが Float で値を保持できない場合、スクリプトは自動的に値を BigDecimal として格納し、値を変数に割り当てるときに Float への型キャストを行います。このとき、多少の精度が失われます。	• 3.14159 • 2E-12 • -100
Integer	Integer 式には、数値 -2147483648 ～ 2147483647 が含まれます。 スクリプトは値をまず Integer として解析します。この試行に失敗すると、スクリプトは値を Long として解析します。この試行に失敗すると、値を BigInteger として解析します。スクリプトは、この値を変数に割り当てるときに、値を Integer に型キャストする場合があります。スクリプトが値を Integer として表現できない場合、結果は不明になる可能性があります。	• 234556789 • 0 • -23
String	String 式には、Unicode 文字 “\u0000” ～ “\uffff” のセットが含まれます。	• “Hello”、“C:\WINNT\win.ini”：この形式ではエスケープ文字も Unicode 文字もサポートされません。 • u“\”This is a quoted string\””、u“\tHello”、u“\u2222\u0065”、u“C:\\WINNT\\win.ini” など：この形式では Character 式型で説明したものと同じエスケープ シーケンスまたは Unicode 文字がサポートされます。

表 10　式の入力形式（続き）

式の型	説明	式の入力形式
Date	Date 式には日付情報が含まれます。 D[] の中に指定したパラメータは、次の 2 つの形式による組み合せに基づいて解析されます。 • “<date>” • “<date> <time>” Cisco Unity Express Script Editor では次の 4 つの <date> 指定形式がサポートされています。 • SHORT。すべて数字の形式。“12/13/03” など。 • MEDIUM。やや長い形式。“Jan 12, 2003” など。 • LONG。より長い形式。“January 12, 2003” など。 • FULL。完全に指定する形式。“Tuesday, April 12, 2003” など。	• D[12/13/03] • D[Dec 13, 2003] • D[January 20, 2003] • D[Tuesday, April 12, 2003] • D[12/13/03] • D[12/13/03 5:50 PM] • D[April 1, 2003 12:00:00 AM PST]
Time	Time 式には時刻情報が含まれます。 T[] の角カッコの中に指定したパラメータは、形式 “<time>” に基づいて解析されます。 Cisco Unity Express Script Editor では次の 3 つの <time> 指定形式がサポートされています。 • SHORT。短い形式。“3:30 PM” など。 • MEDIUM。長い形式。“3:30:32 PM” など。 • LONG および FULL （同一）。より完全な形式。“3:30:42 PM PST” など。	• T[3:39 AM] • T[11:59:58 PM EST]
BigDecimal	BigDecimal 式には、スケールを持つ任意の精度の整数が含まれます。スケールは小数点の右側の桁数です。	• 3.14159 • 2E-12 • -100
BigInteger	BigInteger 式は、任意の精度の整数を表します。	• 234556789 • 0 • -23
Double	Double 式は、拡張た Float 変数を表す。 スクリプトが Double で値を保持できない場合、スクリプトは自動的に値を BigDecimal として格納し、値を変数に割り当てるときに Double への型キャストを行います。このとき、多少の精度が失われます。	• 3.14159 • 2E-12 • -100
Long	Long 式は、拡張 Integer 変数を表します。 スクリプトは値まずを Long として解析します。この試行に失敗すると、値を BigInteger として解析します。スクリプトは、この値を変数に割り当てるときに、値を Long に型キャストする場合があります。スクリプトが値を Long として表現できない場合、結果は不明になる可能性があります。	• 234556789 • 0 • -23

Expression Editor での演算子の使用方法

Cisco Unity Express Script Editor では、次の演算子を使用できます。実行の優先順位に従って説明します。

1. カッコ（...）：任意の式で機能します。カッコ内に含まれる式に優先順位を与えることができます。
2. 乗算（*）、除算（/）：任意の数式（Integer、Long、Float、Decimal、BigInteger、BigDecimal）で機能します。

   数値オペランドは、テスト前に有効な型へ適切にプロモートされます。
3. 加算（+）、減算（-）：任意の数式（Integer、Long、Float、Decimal、BigInteger、BigDecimal）で機能します。

   数値オペランドは、テスト前に有効な型へ適切にプロモートされます。
4. より小（<）、より大（>）、以下（<=）、以上（>=）

   比較オペランドは String、Character、および Number オペランドだけで機能します。
5. 等しい（==）、等しくない（!=）

   <null> 定数のテストは 2 つの等価演算子でサポートされています。
6. および（&&）：Boolean 式だけで機能します。
7. または（||）：Boolean 式だけで機能します。
8. 連結（+）

   少なくともオペランドの 1 つが String であり、別の 1 つがプロンプトではない場合、後者は `String.value( )` メソッドを使用して String に変換されます。その結果、新しい String は両方のオペランドの String 表現の連結となります。通常、String.valueOf() メソッドは連結中のオブジェクトの toString() メソッドを呼び出すか、またはオブジェクトがヌルの場合は文字列「null」を返すだけです。

   オペランドが Character の場合、すべてが連結されて 1 つの新しい String になります。

エラー処理の設定

Cisco Unity Express Script Editor では、さまざまな方法でエラーを処理するスクリプトを作成できます。

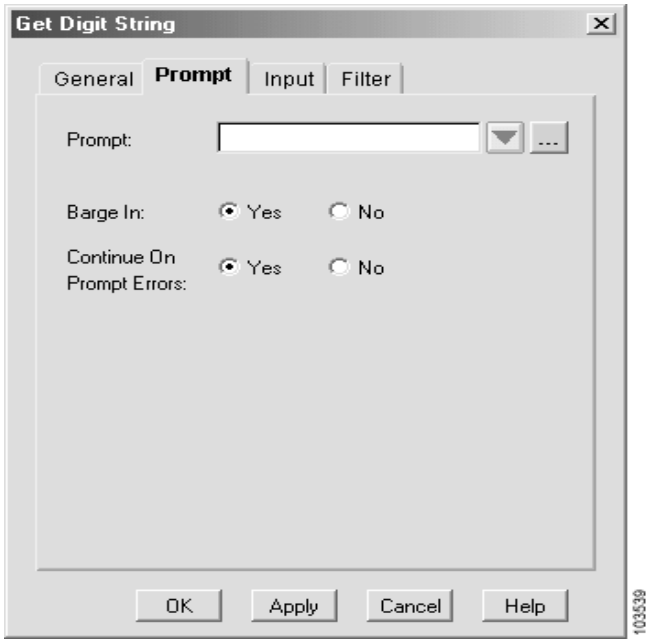
基本エラー処理の設定

Continue on Prompt Errors

Continue on Prompt Errors オプションを使用すると、スクリプトが無効な入力を受け取ったときに（**Invalid Audio Format** や **File Not Found** など）、スクリプトを継続して実行できます。

Cisco Unity Express Script Editor には、Media パレットにあるステップのカスタマイザ ウィンドウに **Continue on Prompt Errors** オプションがあります。P.104 の「Media ステップ」を参照してください。たとえば、図 10 は **Get Digit String** カスタマイザ ウィンドウの **Prompt** タブを示しています。

図 10　Continue on Prompt Errors オプション：Get Digit String カスタマイザ ウィンドウの Prompt タブ



有効な場合、ステップは再生されるプロンプトのリストにある次のプロンプトを続行します。リストにある最後のステップであれば、発信者の入力を待ちます。

Continue on Prompt Errors を有効した場合は、プロンプト エラーを無視し、このプロンプトの再生が成功したものとして処理を続行するようにスクリプトに指示します。たとえば、一連のプロンプト 1 + 2 + 3 でプロンプト 1 番が失敗した場合、ステップはプロンプト 2 番を続行します。プロンプト 3 番が失敗すると、ステップはプロンプト 3 番が正しく再生されたものとして発信者の入力を待ち、処理を続行します。

Continue on Prompt Errors を無効にした場合は、Media ステップは例外を生成します。この例外はスクリプトで処理できます。

使用可能なプロンプトの例外は次のとおりです。

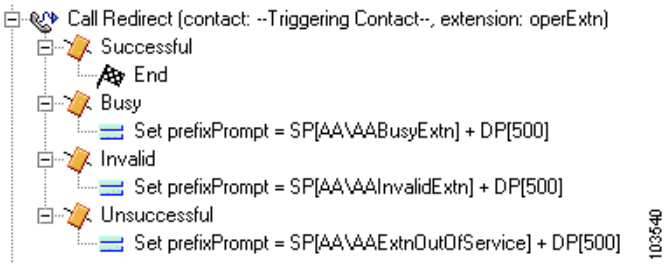
- PromptException
- UndefinedPromptGenerator
- UndefinedPromptException
- InvalidPromptArgumentException
- UnsupportedPromptExpression

エラー出力ブランチ

エラー出力ブランチには、一般的なエラーが発生したときの操作について指示するロジックが含まれます。

図 11 に、スクリプトの **Call Redirect** ステップの下にあるエラー出力ブランチを示します。

図 11　Call Redirect ステップのエラー出力ブランチ



この図の **Call Redirect** ステップには、無効な内線用と使用不能の内線用の両方のロジックが含まれています。

（注）　スクリプトでは、予期されるエラー条件についてだけ、エラー出力ブランチを提供します。システム エラーについては提供しません。

拡張エラー処理の設定

On Exception Goto ステップの使用方法

General パレットの **On Exception Goto** ステップ（P.87 の「On Exception Goto」を参照）は、例外が生成されたときにスクリプト内の指定された場所に実行を渡します。

スクリプトで特定の例外に対して **On Exception Goto** ステップを使用することにより、特定の例外用に新しいハンドラを登録するか、または既存のハンドラを上書きすることができます。

登録処理はスクリプト全体に影響を与えます。割り当てられたハンドラは、その例外が発生する場所（指定ステップの前、中、または後）に関係なく、スクリプトをアクティブにします。ステップの実行後、新しいハンドラが再登録されるか、**General** パレットの **On Exception** Clear ステップで例外が消去されるまで、このハンドラは登録されています。

サブフローの原因となる例外の場合、スクリプトは最初にサブフローの例外ハンドラを調べます。指定された例外に何も定義されていない場合は、例外によってサブフローが中止され、Cisco Unity Express アプリケーションが親スクリプト内の例外ハンドラを検索します。この処理は、スクリプトが例外ハンドラを見つけるか、スクリプトの最上位に達するまで続行されます。

例外ハンドラが登録されていない場合、スクリプトは中止され、エラー処理は、エラー処理の最終レベル、つまりデフォルト スクリプトに戻ります。

デフォルト スクリプトの使用方法

デフォルト スクリプトは、Cisco Unity Express Script Editor がデフォルトのシステム処理をアクティブなすべての連絡先に適用する前の、ユーザ定義エラー処理の最終レベルです。

Cisco Unity Express Script Editor からこのデフォルト スクリプトが起動されるのは、次の条件の場合です。

- メイン スクリプトが中止される。これは、次のいずれかの理由で発生します。
  - 検出されていない例外が発生する。
  - 正しく検証されていないため、Cisco Unity Express アプリケーション ソフトウェアがプライマリ スクリプトを起動できない。
- Cisco Unity Express アプリケーション ソフトウェアがアプリケーションで同時に実行できるセッション数の制限に到達したため、着信を中止する必要がある。

どちらの場合も、デフォルト スクリプトが実行される前に Cisco Unity Express Script Editor によってアクティブなすべての連絡先に中止のマークが付けられます。これらの連絡先は、デフォルト スクリプトの実行の結果、転送またはリダイレクトされた場合でも、最後の状態が ABORTED となります。

（注） デフォルト スクリプトの目的は、メイン スクリプトの失敗時に通話を適切に終了することであり、プライマリ スクリプトが対象とした元のサービスを提供するフォールバックを用意することではないことに注意してください。この違いは重要です。このデフォルト スクリプトの実行にシステム リソースを使用すると、システム パフォーマンスが低下する可能性があるためです。プライマリ スクリプトの失敗回数が多い場合は、同じ操作を試みる別のスクリプトを用意するのではなく、このプライマリ スクリプトを修正してください。

プライマリ スクリプトが正常に終了する場合、デフォルト スクリプトは実行されません。プライマリ スクリプトの終了時にまだ連絡先がアクティブになっている場合、処理済みのマークが付いていないアクティブな連絡先はすべて中止され、処理済みのマークが付いたアクティブな連絡先はすべてそのまま終了します。この場合、設計上の問題がないかどうかプライマリ スクリプトを確認してください。

（注） デフォルト スクリプトはシステム上の問題について連絡先に最終的なフィードバックを示すだけで、サービスの継続やサービスの再開は行わないことに注意してください。

連絡先がまだアクティブであれば、デフォルト スクリプトがある場合はその実行後に、システムによって CallContact スクリプトが適用されます。CallContact スクリプトは通知として「We are currently experiencing system problems, please call back later」というプロンプトを再生し、その後に高速の通話中信号が続きます。

スクリプト割り込みの設定

スクリプト割り込みは、スクリプトの別の部分に戻るために、またはスクリプトの実行を停止するために、スクリプトの現在の処理に外部イベントが割り込むことができる機能です。

スクリプト割り込みは、通常、連絡先の 1 つがリモートで終了したことをスクリプトに通知する必要があるときに使用します。たとえば、発信者が電話を切った場合などです。

（注） どの場合でも、必要なスクリプト割り込みをトリガーするイベントは、スクリプトがほかのステップを実行中にいつでも発生する可能性があります。

スクリプトはデフォルトで、任意のステップが実行される前に自動的に割り込み可能になります。外部イベント（上で説明したようなイベント）がスクリプトに割り込む場合、スクリプトは次のステップの実行を開始する前に、特定のイベントに対する適切な処理に基づいて処理を続行します。

2 つの連続するステップを割り込みの可能性がない状態で実行するには、この 2 つのステップをサブフローに移し、そのサブフローをスクリプトが処理する間は割り込みをすべて無効にする必要があります。

Cisco Unity Express Script Editor では、外部イベントの発生時にスクリプトが内部のステップに割り込むことができるかどうかを指定できる “interruptible” オプションが、一部のステップに用意されています。

連絡先がリモートで終了するときに、スクリプトは次のいずれかの操作を実行します。

- 発信者が電話を切ると、スクリプトは（可能であれば）割り込まれて、ContactInactiveException が生成される。この例外は、**General** パレットの OnExceptionGoto ステップで検出されて、正しく処理されます。
- 発信者が電話を切り、使用可能な例外処理ロジックがない場合、スクリプトはただちに中止される。
- 複数の連絡先を管理している場合、**OnExceptionGoto** ステップはリモートで終了した連絡先を識別できない。その代わり、既知のすべての連絡先変数でループできるように Label を指定し、**General** パレットの **Get Contact Info** ステップを使用して Active フラグを検索する必要があります。

スクリプトが割り込み可能ではないときに割り込みイベントが発生した場合、スクリプトは再度割り込み可能になったときに自動的に割り込まれます。たとえば、割り込み不可のマークが付いたサブフローを実行しているときは、スクリプトは割り込み可能ではありませんが、そのサブフローが終了して制御が親に戻ると、すぐに割り込みを処理します（そのプライマリ スクリプトが割り込み可能な場合）。