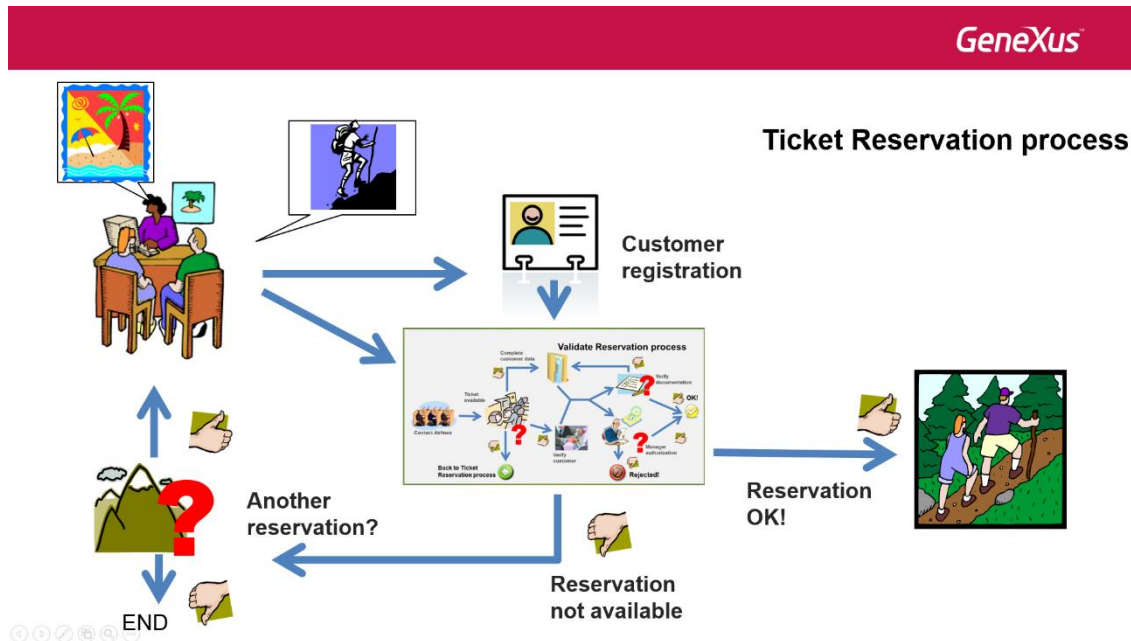


## 同時タスクの定義、エラーの検出と特定

旅行代理店が使用するチケット予約プロセスには、「Validate Reservation」と呼ばれるサブプロセスが含まれています。このサブプロセスは、入力された予約の詳細を検証します。



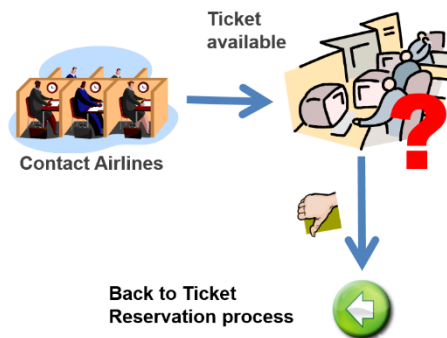
このプロセスでは、まず対応する航空会社に連絡して、要求された日付に利用可能なフライトがあるかどうかを確認することで、予約の空き状況が確認されます。



取得された情報には、利用可能なチケットがないことが示されている場合があります。 この場合、チケットの予約プロセスに戻って、顧客に別の予約を希望するかどうかを尋ねる必要があります。

GeneXus

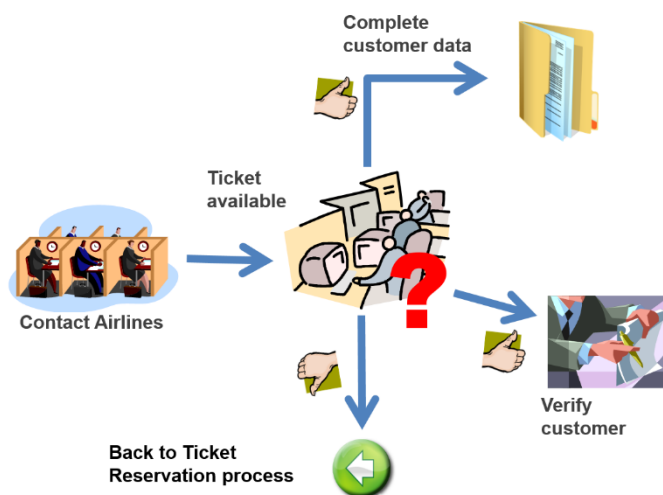
### Validate Reservation process



または、リクエストされた日付に利用可能なチケットがあり、検証プロセスが続行される場合もあります。そのためには 2 つの道をたどる必要があります。

GeneXus

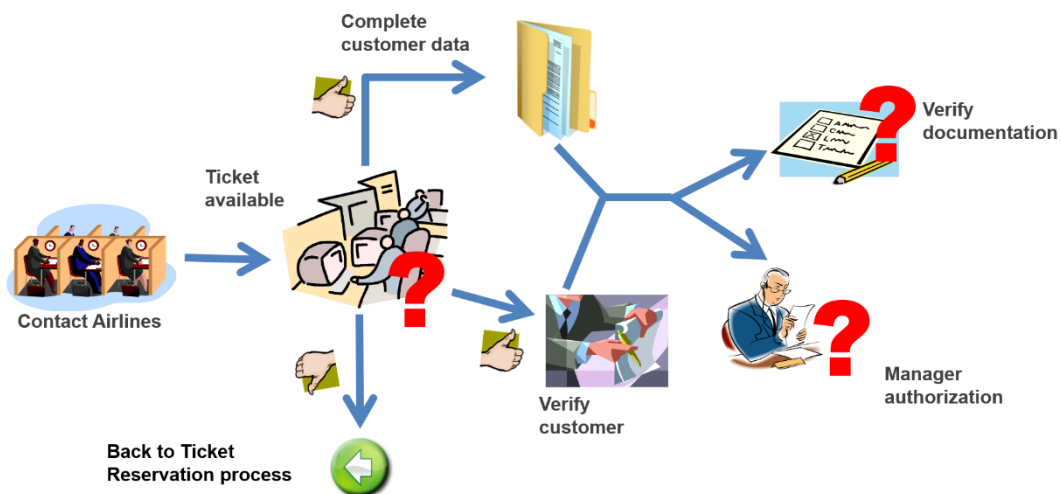
### Validate Reservation process



一方では、ビザ、パスポート、予防接種など、旅行に必要な顧客の詳細情報を記入する必要があります。さらに、旅行代金によっては、顧客の財務状況に関する情報を取得する必要があります。

GeneXus

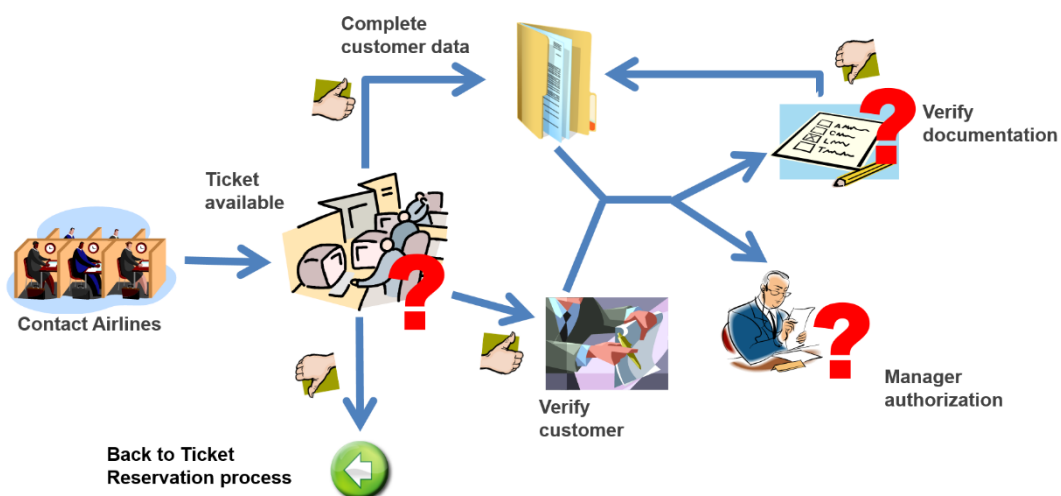
### Validate Reservation process



必要な情報をすべて取得したら、評価する必要があります。まず旅行に必要な書類を確認し、その後カスタマーケアマネージャーが予約を承認する必要があります。

GeneXus

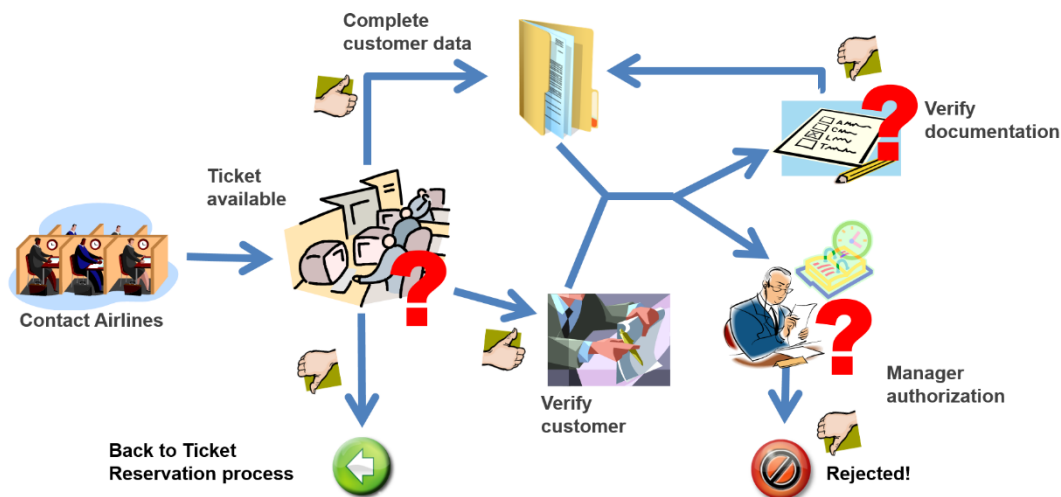
### Validate Reservation process



もし書類が足りない場合は、必要なものを取りに戻らなければなりません。

GeneXus

### Validate Reservation process

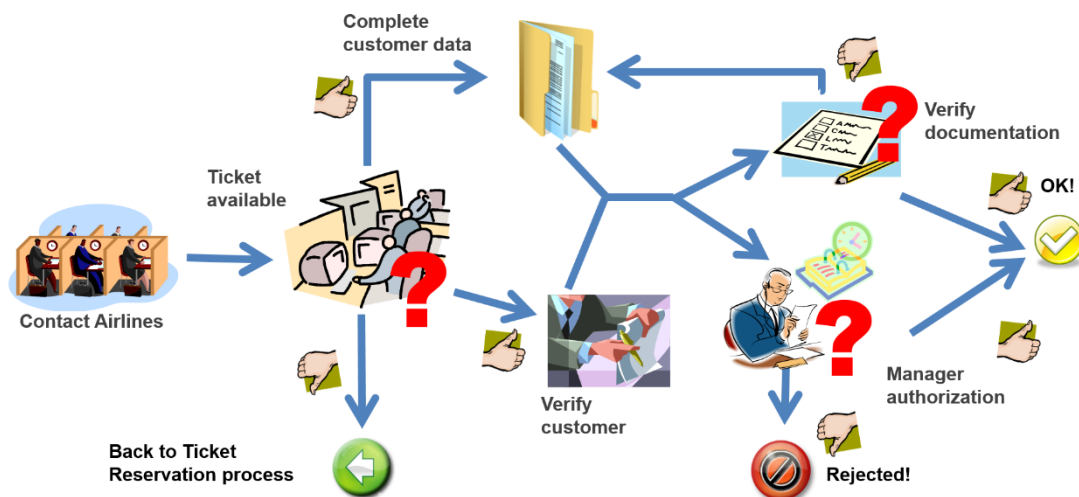


顧客ケアマネージャーは予約を調べ、一定の期間内に予約を承認するかどうかを決定する必要があります。 このため、このタスクについて思い出させるために定期的に通知が表示されます。

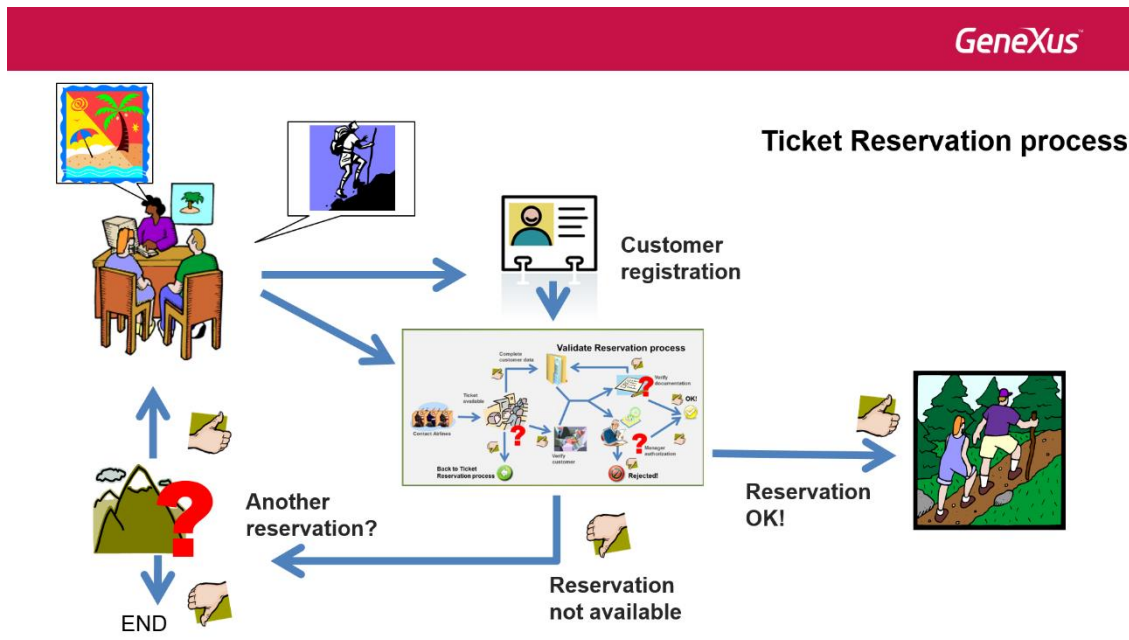
検討した結果、許可しないと判断した場合には、その理由を顧客に通知しなければなりません。 検証サブプロセスが終了し、チケット予約プロセスも終了します。

GeneXus

### Validate Reservation process



マネージャーが予約を承認し、すべての書類が提供された場合、予約は承認されます。

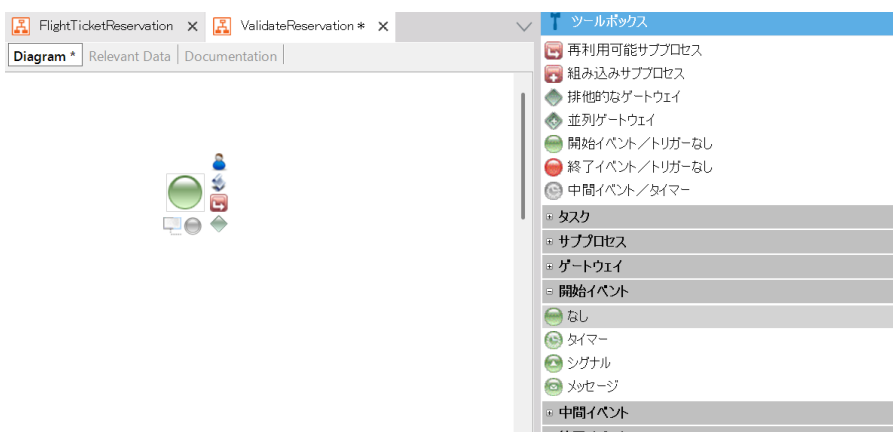


この場合、予約検証サブプロセスは終了します。 システムはメインのチケット予約プロセスに戻り、予約が確認されたことをお客様に通知します。

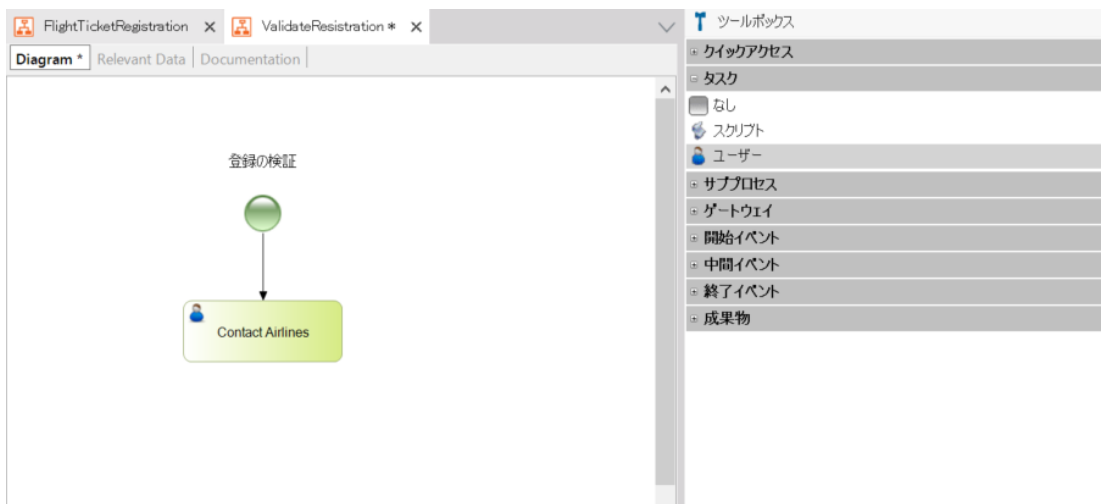
次に、GeneXus のビジネスプロセスダイアグラムを使用して、予約検証プロセスを段階的にモデル化します。

まず、ビジネスプロセスダイアグラムタイプのオブジェクトを作成し、「ValidateReservation」という名前を付けます。

プロセスの開始を示すために、ツールバーから「開始イベント」の「なし」シンボルをドラッグします。

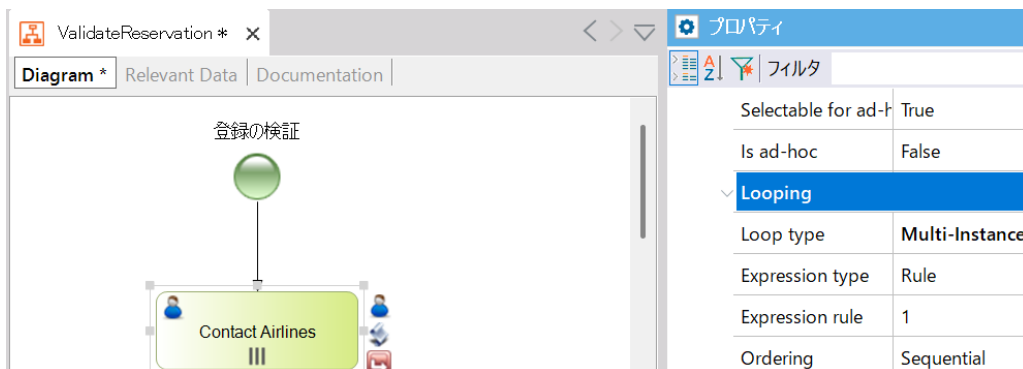


Name プロパティに「登録の検証」と設定し、次に航空会社に連絡する最初のタスクを追加します。これは対話型タスクであるため、ユーザータスクをダイアグラムにドラッグし、F2 キーを押して「Contact Airlines」という名前を付けます。最後に、開始ノードから結合します。

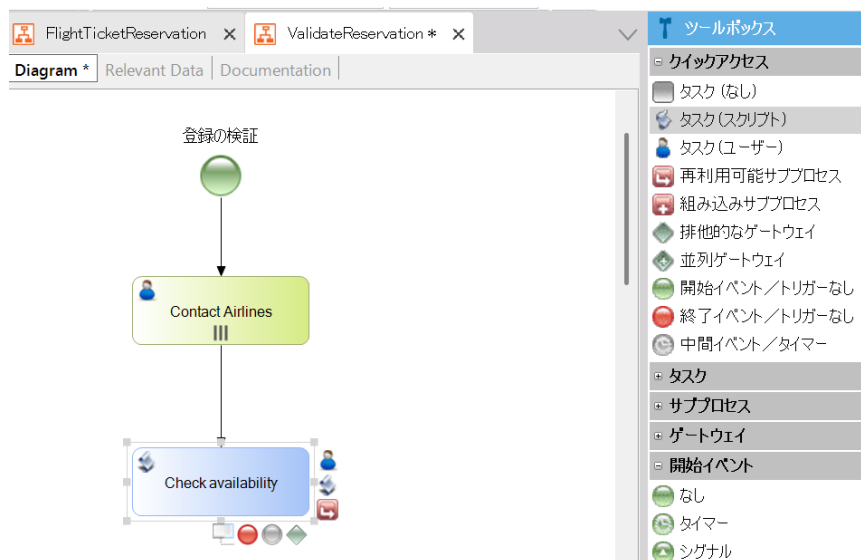


このタスクの特徴は、複数の航空会社に連絡する必要があるため、一定回数実行されることです。これは同時に実行できます。

これを定義するには、「Loop Type」プロパティを「Multi-Instance」に設定します。また、「Expression rule」プロパティに「1」を設定します。



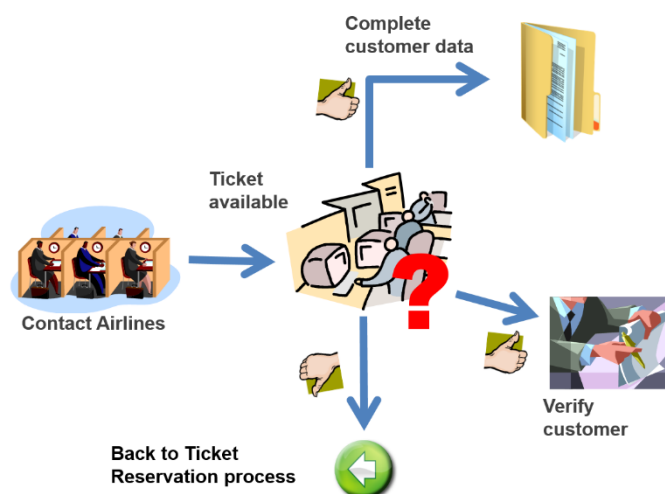
すべての航空会社に連絡したら、入手した情報を調べて、要求された日付に利用可能なフライトがあるかどうかを確認する必要があります。これをモデル化するには、「Check availability」というスクリプトタスクを挿入し、「Contact Airlines」タスクから接続します。



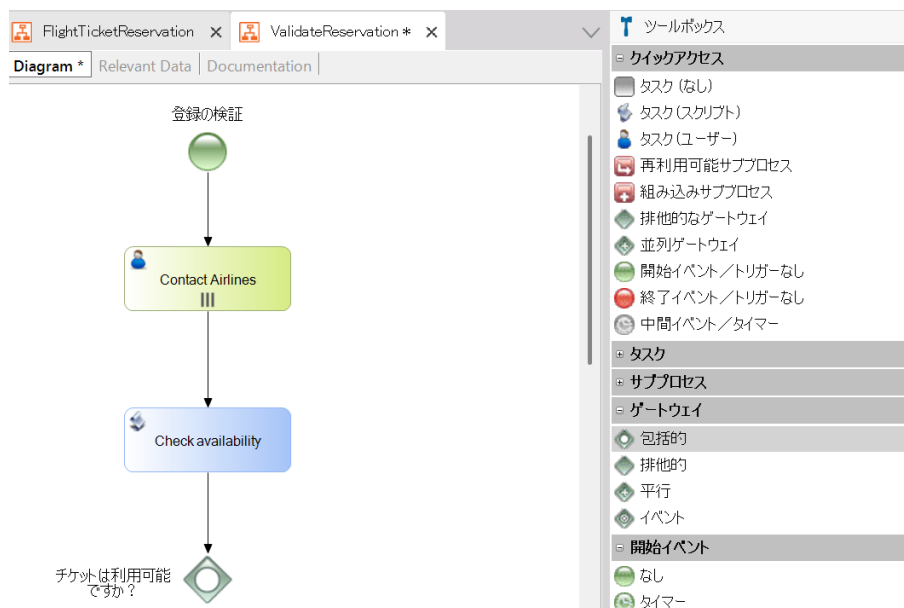
次に、これらの問い合わせの結果を評価する必要があります。利用可能なチケットがない場合は、メインの予約プロセスに戻り、顧客に別の予約を希望するかどうかを尋ねる必要があります。利用可能なフライトがある場合は、旅行の詳細を完了することと、顧客の財務状況を確認することの2つの手順を同時に行う必要があります。

GeneXus

### Validate Reservation process



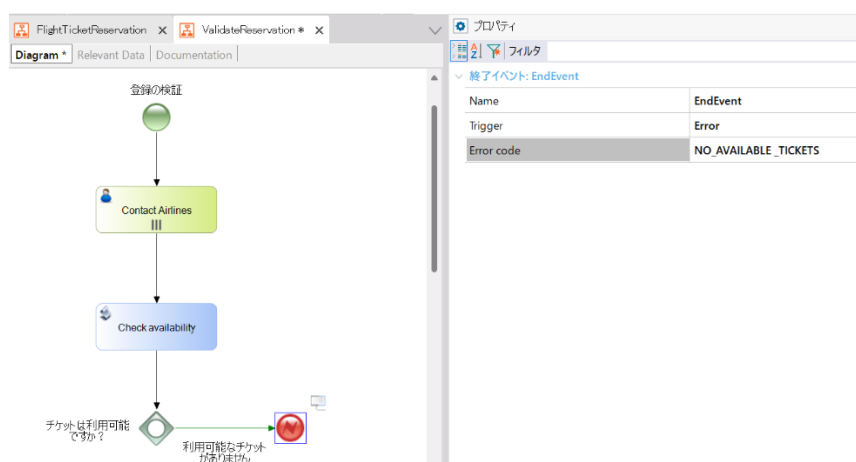
フローは複数のパスをたどる必要があるため、これをモデル化するために排他的ゲートウェイを使用することはできません。したがって、ツールバーから包括的ゲートウェイのシンボルをドラッグし、空き状況の確認タスクから接続し、「チケットは利用可能ですか?」という説明を追加します。



このタイプのノードでは、そこから出てくる各パスに対して条件が定義され、フローは条件を満たすすべてのパスをたどります。

考えられるパスは、利用可能なチケットがない場合です。この場合は検証するチケットがないため、チケット検証のプロセスをキャンセルする必要があることを示す必要があります。さらに、メインプロセスに戻って顧客に通知し、予約の新しい日付をリクエストする可能性を提供する必要があります。

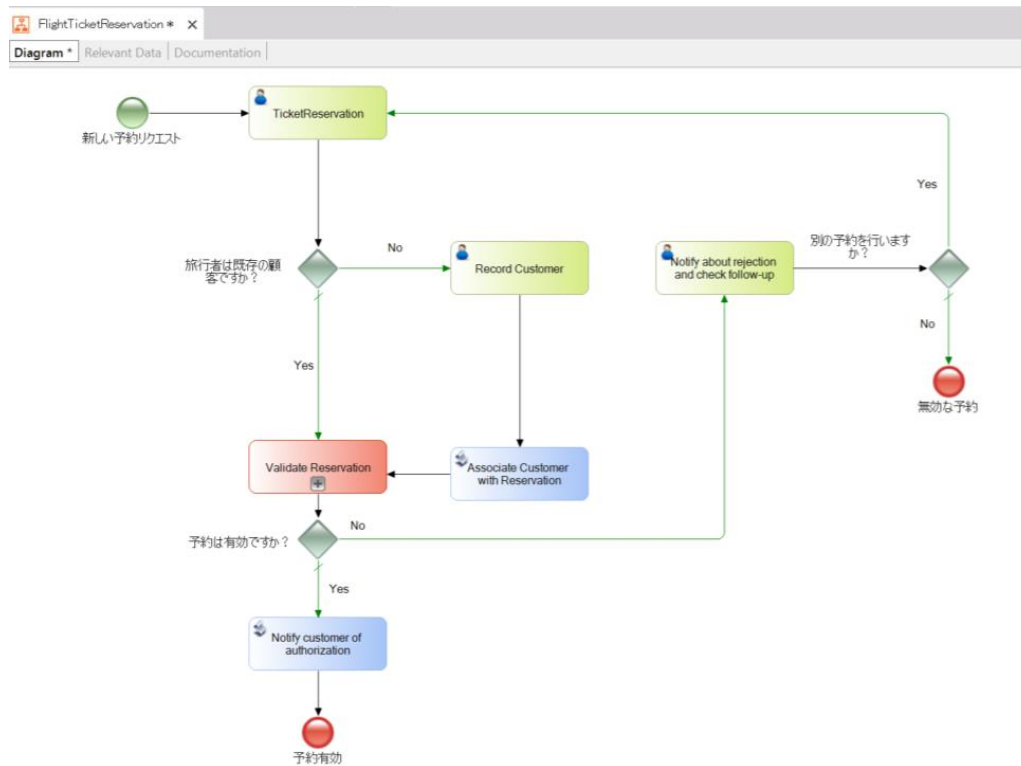
終了イベントのエラーを挿入することで、この「エラーによる終了」の状況をモデル化できます。また、包括的なゲートウェイから接続し、「利用可能なチケットがありません」という説明を追加します。「Error Code」プロパティに「NO\_AVAILABLE\_TICKETS」と入力します。



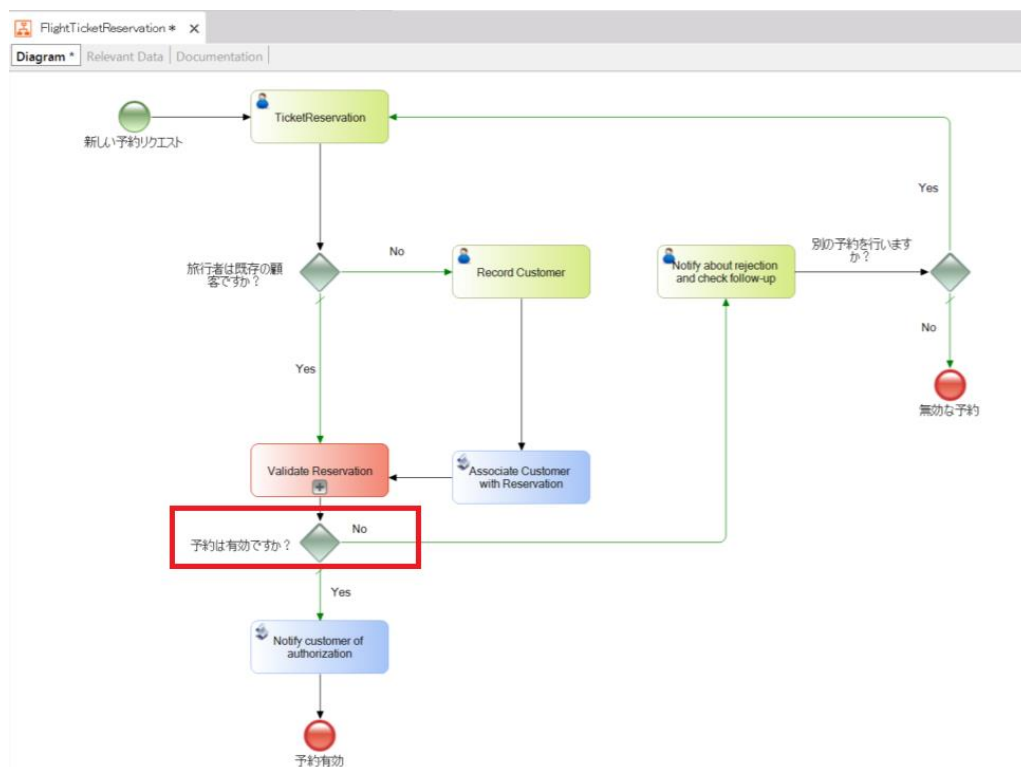
プロパティ	
終了イベント: EndEvent	
Name	EndEvent
Trigger	Error
Error code	NO_AVAILABLE_TICKETS



このタイプの終了イベントは、フローが別のプロセスで続行できるようにするエラーをスローします。このエラーを見つけるために中間エラーイベントを追加します。



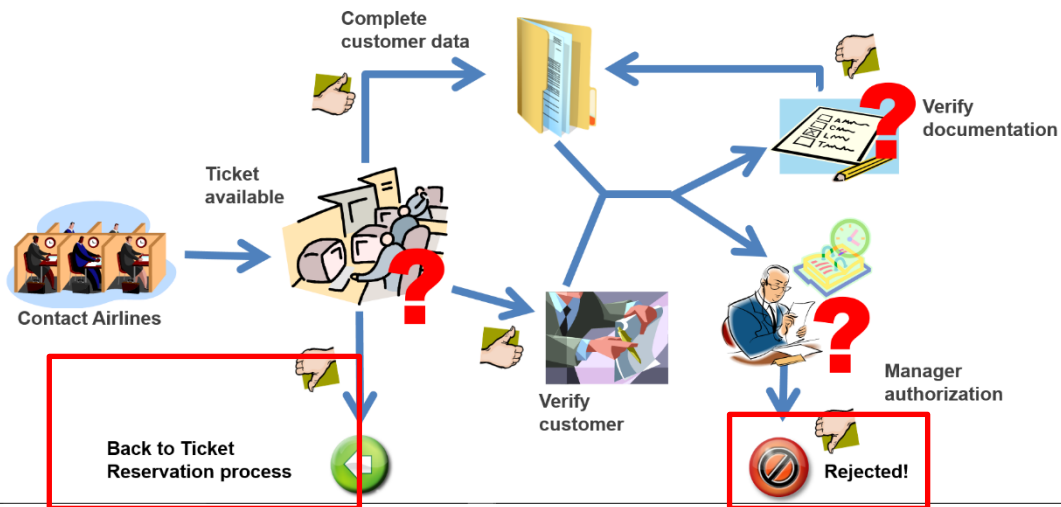
メインのチケット予約のプロセスに戻ると、予約が有効かどうかを評価するために排他的ゲートウェイが必要だったことがわかります。



ただし、このモデルでは、予約が無効である理由はわかりません。前にも述べたように、利用可能なチケットがない場合、またはカスタマーケアマネージャーが予約を許可しない場合、予約は完了しないことがあります。

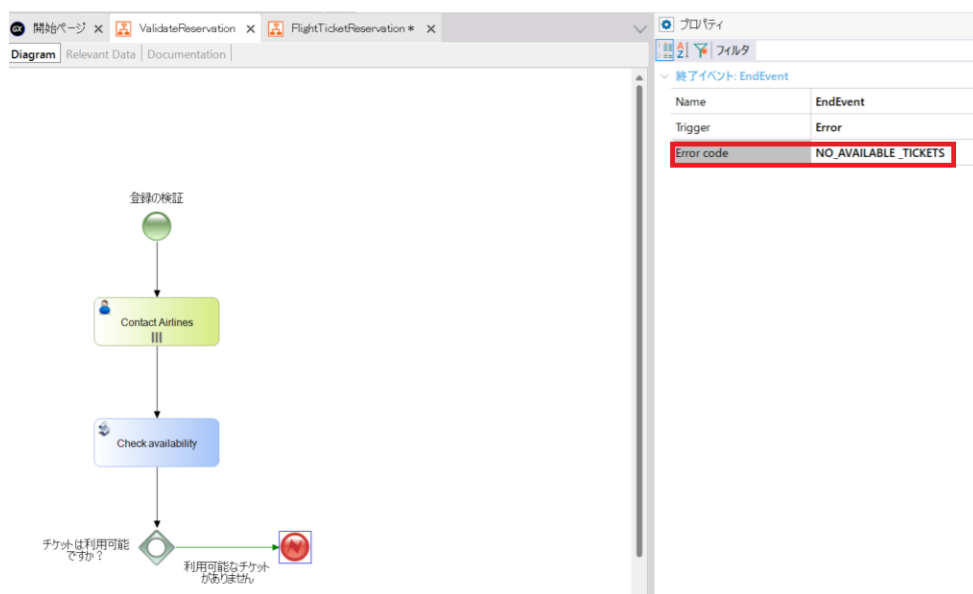
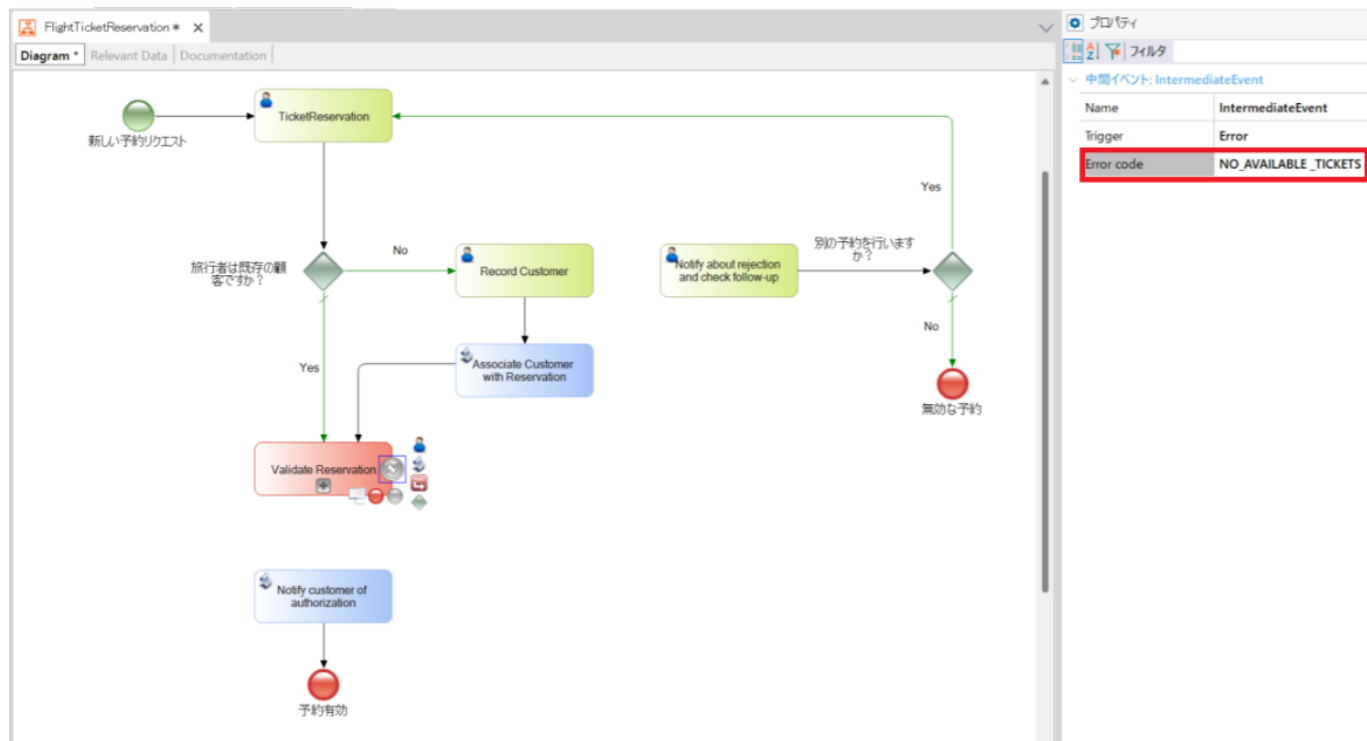
GeneXus™

### Validate Reservation process

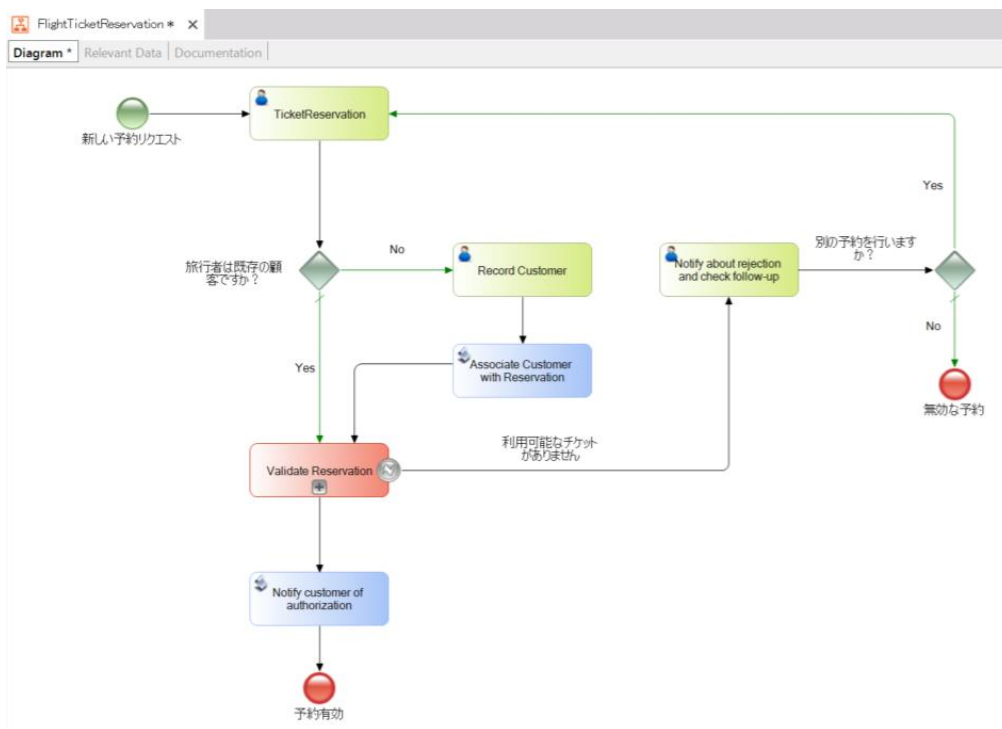


このモデルでは、「エラー終了イベント」と「エラー中間イベント」を利用して、サブプロセスでエラーが発生した状況を特定し、メインプロセスでさまざまなアクションを実行することができます。

これをモデル化するには、予約の有効性を評価したゲートウェイを削除し、サブプロセス シンボルに追加する「エラー中間イベント」に置き換えることによって、「FlightTicketReservation」プロセスを変更します。このイベントのプロパティを編集し、エラーコードに「NO\_AVAILABLE\_TICKETS」と入力します。このエラーコードは、「ValidateReservation」サブプロセスの「エラー終了イベント」で定義されたものと一致する必要があります。



このイベントを「Notify about rejection and check follow-up」というタスクに関連付けます。 この接続に「利用可能なチケットがありません」という説明を追加し、「ValidateReservation」 サブプロセスを「Notify customer of authorization」 タスクに接続してダイアグラムを完成させます。



このようにして、利用可能なチケットがないために予約を完了できない場合をモデル化しました。

カスタマーケアマネージャーが予約を承認しないケースをモデル化するために、別の「エラー中間イベント」をサブプロセスに追加します。次に、ツールバーからスクリプトタスクをドラッグし、「Notify rejection to customer」という名前を設定して、エラーイベントから接続します。これに対して、「カスタマーケアマネージャーによって拒否されました」という説明を追加します。

最後に、「無効な予約」という説明を持つ終了イベント/トリガーなしを追加し、通知タスクから接続します。

