記事
Mihoko Iijima · 2020年10月27日 8m read

【はじめてのInterSystems IRIS】Interoperability（相互運用性）：コンポーネントの作成（ビジネス・プロセス）

この記事はこちらの投稿の続きの内容です。

前回の記事
では、システム統合に必要なコンポーネントの中から、ビジネス・オペレーションの作成について解説しました。

今回の記事では、確認した2つのビジネス・オペレーションを順番を守って呼び出しを行うビジネス・プロセスの作成について解説します。

- プロダクション
- メッセージ
- **コンポーネント**
  - ビジネス・サービス
  - **ビジネス・プロセス**
  - ビジネス・オペレーション（前回の記事）

ビジネス・プロセスは処理の調整役（司令塔）として働きます。

サンプルの中で行いたい処理の調整は、以下の内容です。

手順①　外部の Web API に**都市名**を渡し**気象情報**を問い合わせる
手順②　①の問合せ結果（**気象情報**）と、処理開始時に受信した購入商品名をDBへ登録する

サンプルのビジネス・プロセスでは、手順① の回答を待って手順② を動かすように調整します。

回答を待つ処理（＝同期を取る処理）ですが、例えば、手順① が数日返ってこない場合、どうなるでしょうか？

数日
回答を待
ち続けている間に
ビジネス・プロセスへ次々に新メッ
セージが渡された場合、メッセージは一旦**キュー**
に格納されるため消失しませんが、新メッセージの処理をビジネス・プロセスが実行できず処理に遅延が発生します。

メモ： ビジネス・プロセスとビジネス・オペレーションにはキューがあります。

そこで、プロダク
ションでは、ビジネス・プロセスの
動きとして、同期呼び出しがある場合に、**A)完全に同期を取る方法**と、
B)応答待機中は他の処理が動作できるようにビジネス・プロセス自身の状態を一旦データベースに保存し、実行環境を明け渡す方法 を用意しています。

A)完全に同期を取る方法は、

同期呼び出しを行っている間は、ビジネス・プロセスの処理は継続中となり全ての処理が終了するまで次のメッセージの処理は待たされる。
➡ 先入れ先出し方式で処理順を保障する必要がある場合に利用します。

B)応答待機中は他の処理が動作できるようにビジネス・プロセス自身の状態を一旦データベースに保存し、実行環境を明け渡す方法は、

同期呼び出しを行った時点で一旦データベースに自分自身の状態を保存する。応答メッセージを受信しメッセージを処理する順番が来たときにデータベースからオープンし次の処理を実行する。
（データベースへのビジネス・プロセスの保存・再オープンは IRIS が管理します）
➡ メッセージの処理順序が入れ替わっても良い場合（＝応答を待つ間に受信した別メッセージをどんどん処理してよい場合）に利用します。

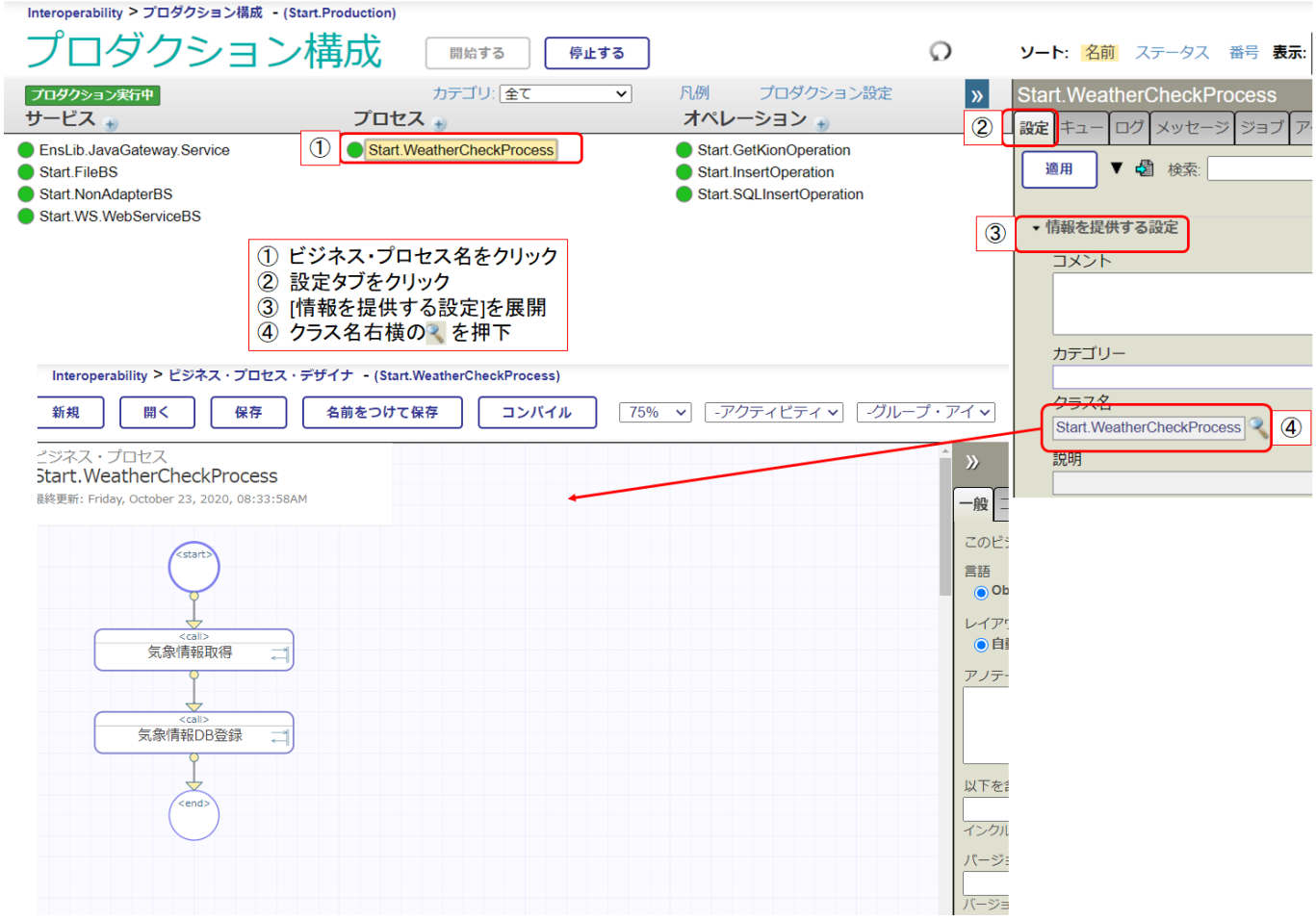
サンプルでは、B) を利用しています。

ビジネス・プロセスを作成するエディタは2種類あり、処理用の箱（アクティビティ）を配置し、実行内容を定義しながら実装できるビジネス・プロセス・エディタと、スタジオや VSCode 上で ObjectScript
を使用して作成する方法があります。

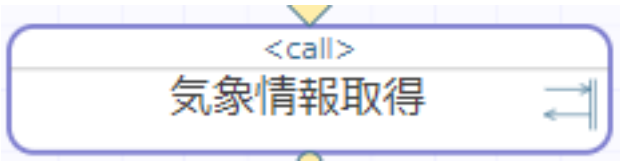
ビジネス・プロセス・エディタを使用する場合、コンポーネントの呼び出しに call
アクティビティを使用しますが、このアクティビティは B)
の方法で実装されます。もちろん、ビジネス・プロセス・エディタでも **A)** の方法を実装できますが、その場合は call アクティビティを使用しません（code アクティビティを使用します）。

では、作成方法について解説します。

ビジネス・プロセス・エディタを利用する場合、管理ポータルで記述します。
また、プロダクション構成画面からビジネス・プロセスを開くこともできます。下図は、その手順です。



このエディタで

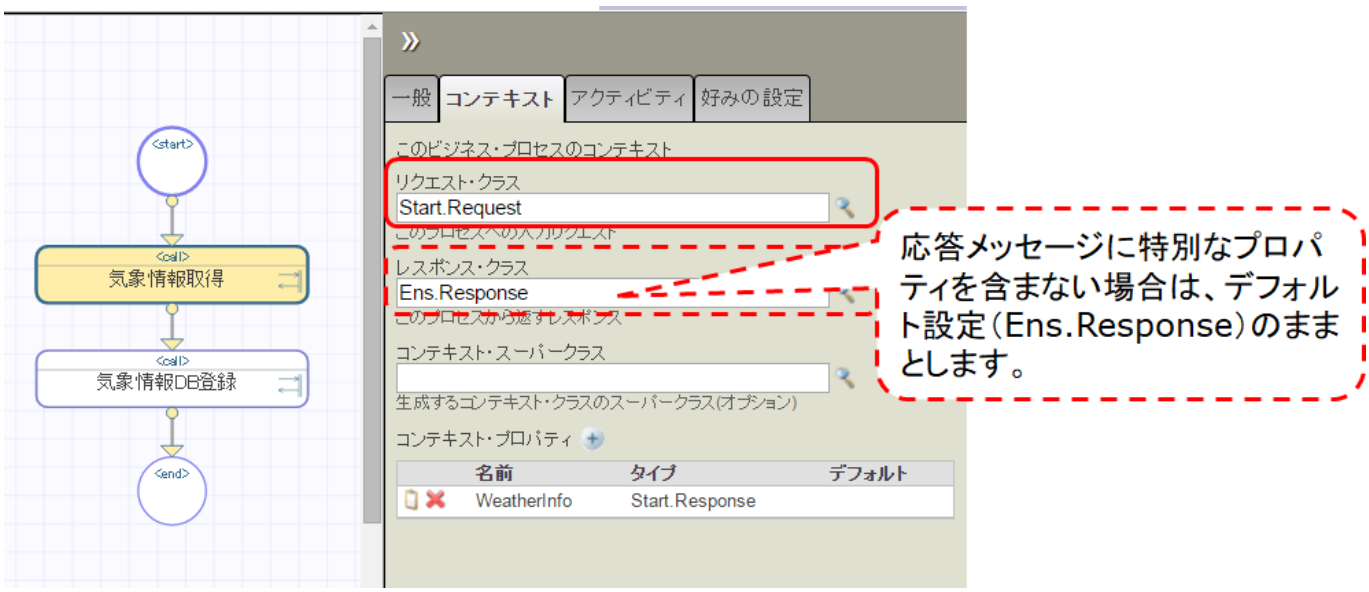


のようなアイコンはアクティビティと呼び、<call>
のマークがあるものは、他コンポーネントの呼び出しができるアクティビティです。

アイコンの右端に 

の記号がありますが、この記号は応答メッセージが返る（＝同期呼び出しを行う）マークです。<call>アクティビティのデフォルトは非同期呼び出し設定のため、必要に応じて変更します。

さて、ビジネス・プロセスは、ビジネス・オペレーションと同様に要求メッセージを受信すると起動するコンポーネントです。
サンプルでは、要求メッセージ：Start.Request
を受信すると起動し、応答メッセージは返送しない設定としています。

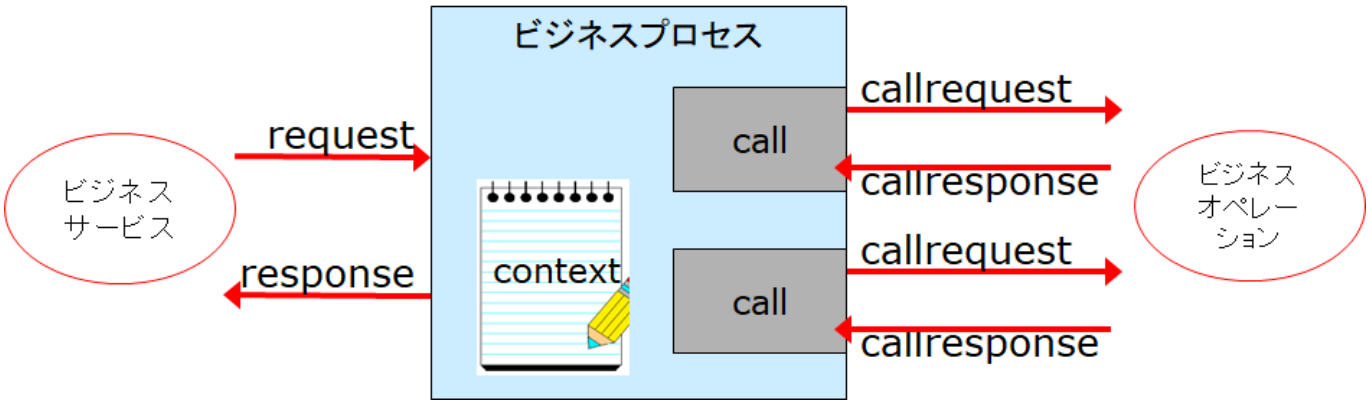


ビジネス・プロセスでは、いろいろな場面でメッセージが登場します。

ビジネス・プロセスに送信される要求メッセージ

<call>
アクティビティを利用して他コンポーネントを呼び出すときに送信する要求メッセージ（＋応答メッセージ）

ビジネス・プロセス・エディタでは、どこからどこへ送受信したメッセージか確認できるようにメッセージを格納するオブジェクト名を明確に分けています。



- request（基本要求）

ビジネス・プロセスの起動のきっかけ
になったメッセージでサンプルでは、Start.Request
（ビジネス・プロセス・エディタ上、コンテキストタブのリクエストの設定に指定するメッセージ）

- response（基本応答）

ビジネス・プロセスの呼び出し元に返送する応答メッセージ（サンプルでは利用していません）
（ビジネス・プロセス・エディタ上、コンテキストタブのレスポンスの設定に指定するメッセージ）

- callrequest（要求メッセージ）

<call>アクティビティで指定したコンポーネントを呼び出すときに送信する要求メッセージ

- callresponse（応答メッセージ）

<call>アクティビティで指定したコンポーネントから返送される応答メッセージ

callrequest と callresponse は、<call>アクティビティの呼び出し処理が終了すると消去されるオブジェクトです。
それ以外は、ビジネス・プロセスの処理が終了するまで消えないオブジェクトです。

さて、callresponse が消えると困ることがあります。

それは、今回のサンプルにもあるように、
コンポーネントを呼び出す際、それ以前に呼び出したコンポーネントの応答結果を利用したい場合に、応答メッセージが消失してしまうため次のコンポーネントで使用する予定だった情報が消えてしまう。

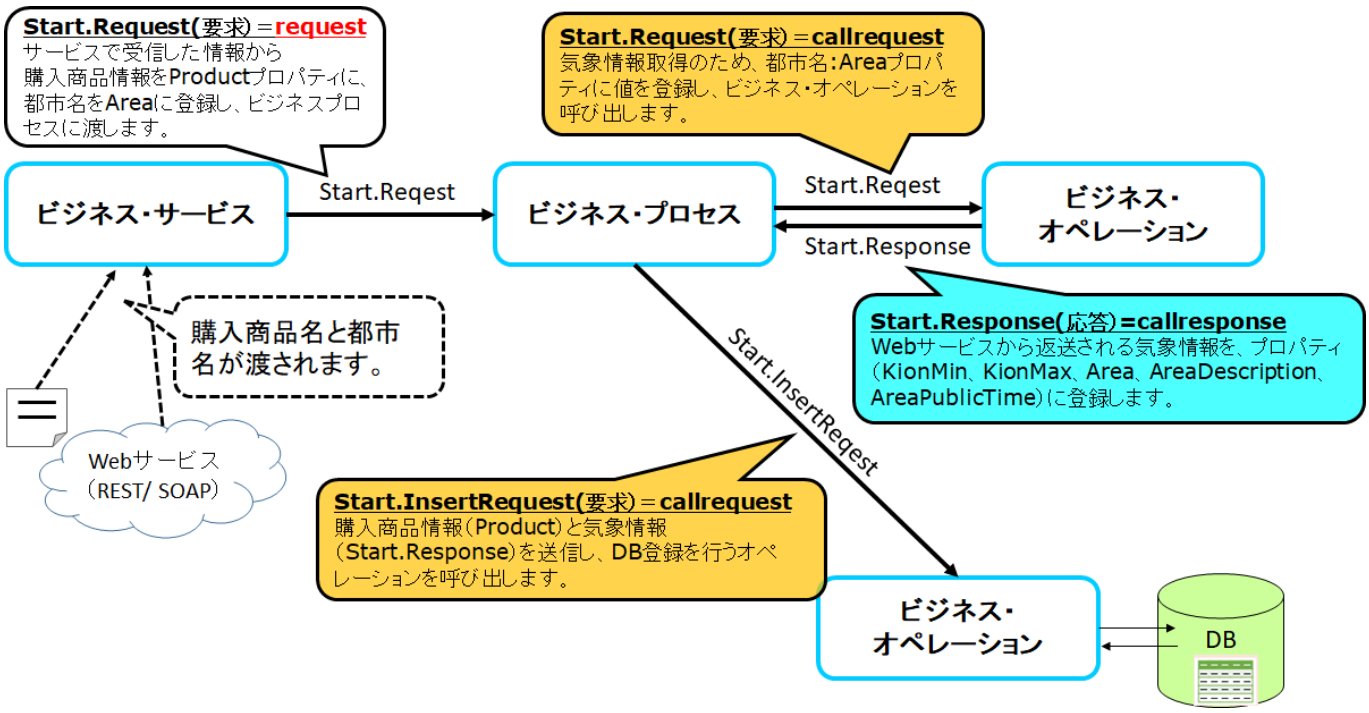
困りました

どうしたらいいでしょうか・・・・・

そんなとき、context オブジェクトを利用します。

context オブジェクトは、request／response
同様、ビジネス・プロセスの処理が終了するまで生き残るオブジェクトです。
しかも context は汎用オブジェクトなので、プロセスエディタ内で定義できます。
context のほかに、response オブジェクトも引き継ぐデータの退避にあうプロパティがあれば、利用できます。

では、今回の流れを再度確認しましょう。

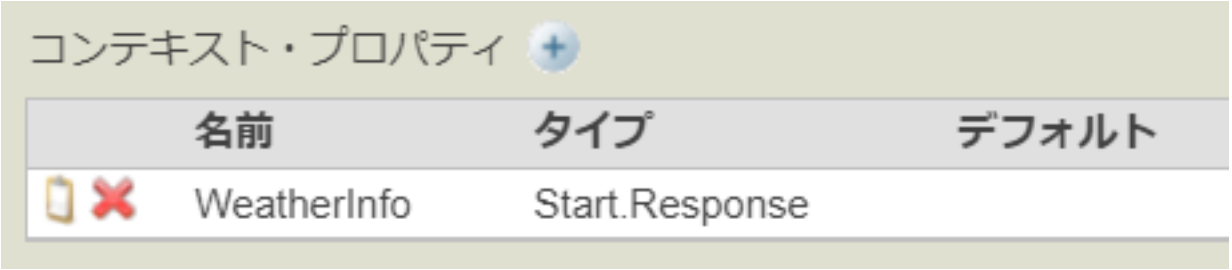


水色の吹き出しの応答メッセージ：Start.Response は処理が終了すると消去されるオブジェクトです。

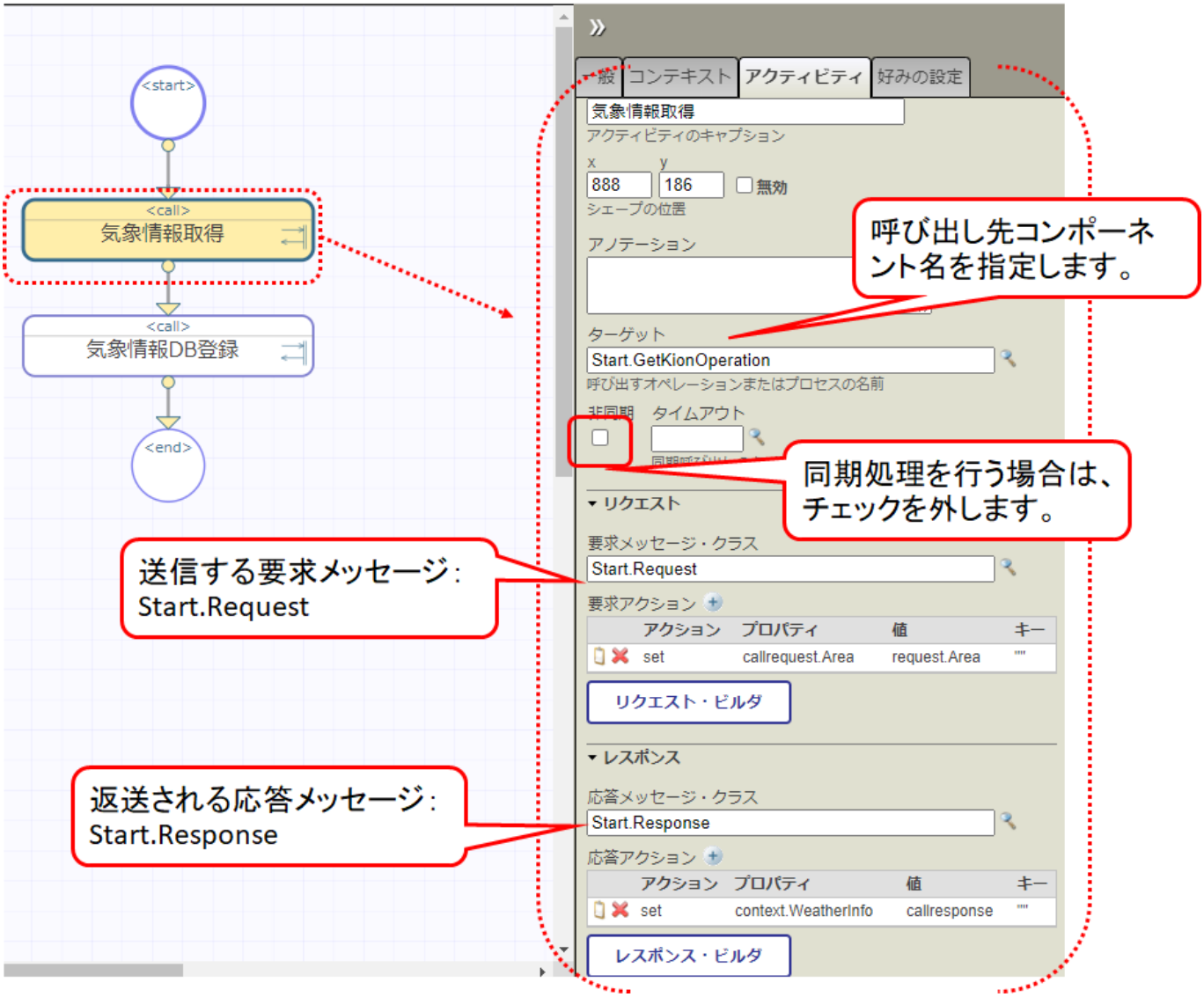
次に実行する [DB更新用ビジネス・オペレーション]
へ送信するメッセージに、気象情報が格納された応答メッセージ（Start.Response）を利用したいため、context
オブジェクトへ応答メッセージ（Start.Response）のプロパティ値を全て代入できるように実装します。

では、context プロパティの設定はどうなっているでしょうか。

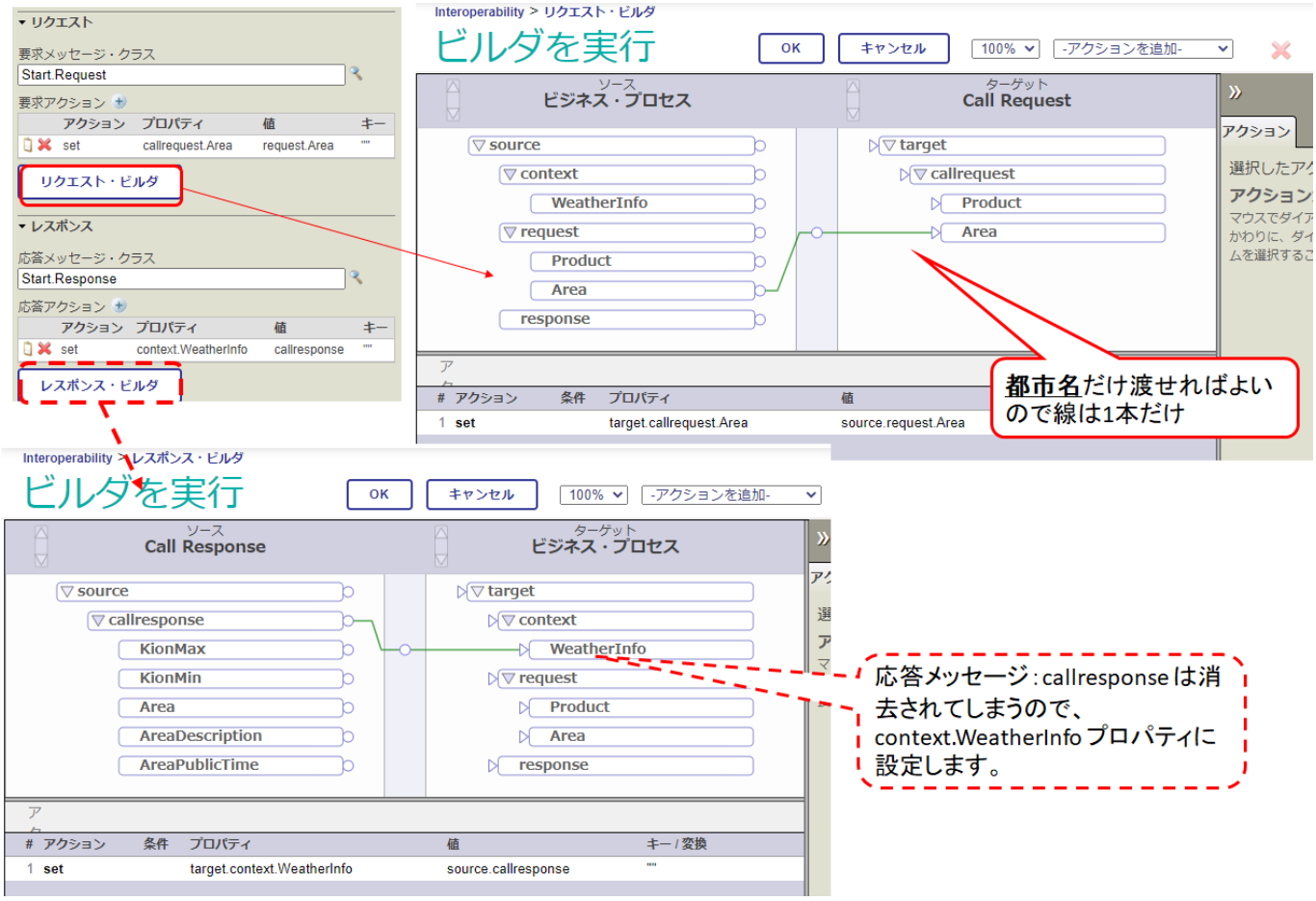
ビジネス・プロセス・エディタのコンテキストタブにある「コンテキスト・プロパティ」にプロパティを定義します。
今回は、応答メッセージ（Start.Response）のプロパティ全てを context
オブジェクトへ退避したいので、プロパティのタイプ指定はStart.Response を設定しています。



つづいて、<call>アクティビティ内の設定を確認します。

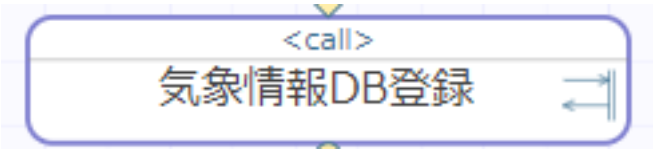


要求メッセージと応答メッセージには、〇〇・ビルダ というボタンがついています。
このボタンを押下すると、それぞれのメッセージのプロパティに何を登録したらよいか？を指定できる線引きエディタを起動します。



この後、データベース更新依頼用ビジネス・オペレーション（Start.SQLInsertOperation か、Start.InsertOperation）を同様に<call>アクティビティで呼び出したら完成です。

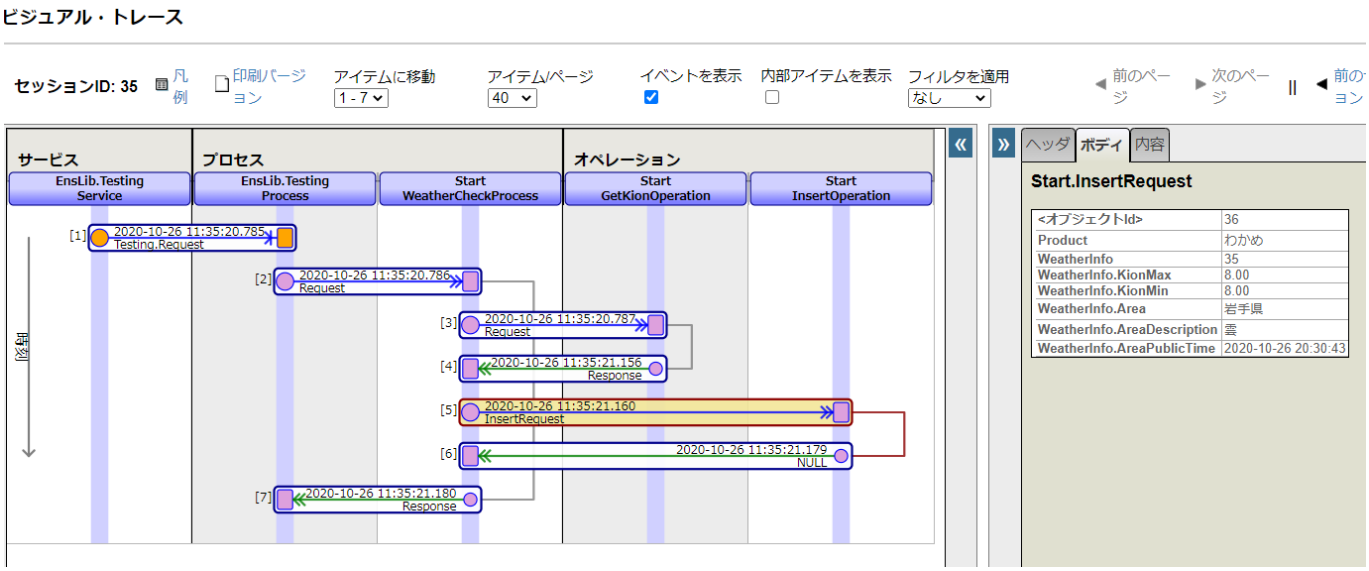
（詳細は、ビジネス・プロセスの



の設定をご参照ください）

確認が完了したらテストします。テストの方法は、ビジネス・オペレーションのテスト方法と共通です（こちらの記事をご参照ください）。

テスト後のトレースは以下の通りです。



ビジネス・プロセスは調整役
なので、同期実行の順番を守りながら定義されたコンポーネントを順次呼び出しているのが確認できました。

メモ1：サンプルでは、call
アクティビティしか扱っていませんが、その他にもデータ変換など様々なアクティビティがあります。

メモ2：ビジネス・プロセス・エディタ以外で ObjectScript
だけで作成するビジネス・プロセスは、Ens.BusinessProcess
クラスを継承します。ビジネス・プロセスエディタで作成した場合は Ens.BusinessProcessBPL
クラスを継承します。

ビジネス・プロセスはシステム連携処理の調整役です。
ビジネス・プロセス・エディタは以下の種類のメッセージ用変数（request/response/callrequest/callreponse/context）を用意しています。
ビジネス・プロセス・エディタで作成したビジネス・プロセス
は、コンポーネントの呼び出しに同期処理があっても、他のメッセージを遅延させない仕組みで動作できます。

次は、いよいよ最後のコンポーネント：ビジネス・サービスの開発方法をご紹介します。

#ビジネスプロセス（BPL） #初心者 #相互運用性 #InterSystems IRIS #InterSystems IRIS for Health

ソースURL:
https://jp.community.intersystems.com/post/%E3%80%90%E3%81%AF%E3%81%98%E3%82%81%E3%81%A6%E3%81%AEintersystems-iris%E3%80%91interoperability%EF%BC%88%E7%9B%B8%E4%BA%92%E9%81%8B%E7%94%A8%E6%80%A7%EF%BC%89%EF%BC%9A%E3%82%B3%E3%83%B3%E3%83%9D%E3%83%B

C%E3%83%8D%E3%83%B3%E3%83%88%E3%81%AE%E4%BD%9C%E6%88%90%EF%BC%88%E3%83%93%E3%82%B8%E3%83%8D%E3%82%B9%E3%83%BB%E3%83%97%E3%83%AD%E3%82%BB%E3%82%B9%EF%BC%89