Article
Bob Kuszewski · Jun 19, 2020 5m read

Migrate from Java Business Host to PEX

Migrate from Java Business Host to PEX

With the release PEX in InterSystems IRIS 2020.1 and InterSystems IRIS for Health 2020.1, customers have a better way to build Java into productions than the Java Business Host. PEX provides a complete set of APIs for building interoperability components and is available in both Java and .NET. The Java Business Host has been deprecated and will be retired in a future release.

Advantages of PEX

- Allows developers to create any Production component in either Java or .NET
- More complex message structures can be passed between components
- Simplified settings
- Simplified development workflow with no need for ObjectScript.

The rest of this article focuses on how to migrate existing Java Business Host code to PEX.

Overview

The classes and interfaces used for PEX are different from Java Business Host (JBH). We'll provide an overview of the differences here, but the full documentation will give you more depth.

- Converting a Business Service
- Converting a Business Operation
- Settings
- Messages
- Logging

Converting a Business Service from Java Business Host to PEX

In order to build a PEX Business Service, you need to implement com.intersystems.enslib.pex.BusinessService instead of com.intersystems.gateway.bh.BusinessService.

The design pattern used by PEX for Business Service has changed from one where the service is expected to start a thread to produce messages to one where the service implements a function that is called periodically to produce messages.

In JBH, your code would look something like this

```
@Override
public boolean OnInit(Production p) throws Exception {
  production = p;

  if (messageThread == null) {
    Messager messager = new Messager();
```

```
      messageThread = new Thread(messager);
      messageThread.start();
    }

    return true;
  }
```

In PEX, you just need to implement three functions

```
  public void OnInit() throws Exception {
    // Initialization
    return;
  }

  public Object OnProcessInput(Object messageInput) throws Exception {
    // Here is where you call SendMessage() or SendMessageAsync()

    return null;
  }

  public void OnTearDown() throws Exception {
    // Shut down
    return;
  }
```

You'll also need to change how settings are used, messages delivered, and logging is done. More on those below.

Converting a Business Operation from Java Business Host to PEX

In order to build a PEX Business Operation, you need to implement com.intersystems.enslib.pex.BusinessOperation instead of com.intersystems.gateway.bh.BusinessOperation.

The design pattern for Business Operations is structurally the same between JBH and PEX, but the parameters to two main entry points have changed.

Changes to OnInit()

In PEX, OnInit() takes no parameters.

Changes to OnMessage()

In PEX, OnMessage() is given a generic Object instead of the String used in JBH. This allows the author of the production to pass any sort of message desired.

In JBH your application might have looked something like this

```
  public boolean OnMessage(String message) throws Exception {
    // Business logic here
    return true;
  }
```

In PEX, the parameter is a generic Java Object that you need to cast appropriately, which allows you to transmit

more complex messages than just strings. Here's an example of how to extract a request that is a file stream.

```
  public Object OnMessage(Object request) throws Exception {
    com.intersystems.jdbc.IRISObject streamContainer = (com.intersystems.jdbc.IRISObj
ect)request;
    com.intersystems.jdbc.IRISObject str = (com.intersystems.jdbc.IRISObject)streamCo
ntainer.get("Stream");
    String originalFilename = (String)streamContainer.get("OriginalFilename");

    Long contentSize = (Long)str.get("Size");
    String content = (String)str.invoke("Read", contentSize);

    // Business logic here

    return null;
  }
```

You'll also need to change how settings are used, messages delivered, and logging is done. More on those below.

Settings

Declaration of settings has been simplified.

In JBH configuration was declared via a SETTINGS string and fetched via code that looks something like this:

```
  String setting = production.GetSetting("Min");
  if (!setting.isEmpty()) {
    min = Integer.parseInt(setting);
  }
```

In PEX, settings are just public member fields. These are automatically populated when the class is instantiated.

```
  public int Min = 0;
```

Any public member field is available to be set in your production as long as the member field is a base Java type (String, int, etc.).

Messages

Message sending is more powerful. In JBH messages are sent as strings. In PEX, messages are sent as objects - either IRISObject, for objects that are defined in ObjectScript, or a subclass of com.intersystems.enslib.pex.Message, for classes defined in Java.

In JBH, your code would look like this

```
  production.SendRequest(value.toString());
```

In PEX, it would be something like this

```
MyExampleMessageClass req = new MyExampleMessageClass("message to send");
SendRequestAsync(Target, req);
```

Logging

Logging functions are all similar, just named differently.

In PEX, you'd log an informational message via LOGINFO()

```
LOGINFO("Received message");
```

Object Gateway

The Java Business Host needed its own gateway. With PEX, you can use a single Java gateway for all your Java needs. Or you can use many gateways. It's up to you. Here's a good introduction to the java gateway.

Conclusion and Feedback

If you haven't tried PEX yet, what are you waiting for? PEX provides the ability to solve a far wider array of business problems with less code, plus you can now do everything in .NET as well.

If you have any questions or problems moving your JBH application to PEX, please reach out myself or the WRC.

#.NET #Best Practices #Interoperability #Java #InterSystems IRIS #InterSystems IRIS for Health