GlassFish ESB v2.1 Field Notes
JavaScript Codelets to Make BPEL Process Wait for a Random Duration Up to a Maximum

Michael.Czapski@sun.com
November 2009, Release 1.0.0

Table of Contents

Introduction

In some specific circumstances, for example when testing high availability and failover scenarios, it may be desirable to make a BPEL process wait for a random amount of time, not exceeding some maximum duration, before continuing.

This Note describes the JavaScript Codelet which, given a maximum duration in Milliseconds, will return a random time up to that maximum duration, as an ISO8601 Duration Literal, suitable for use in the BPEL Wait activity. An example process that uses this Codelet is also developed and discussed.

This Note relies on the material presented in the Blog Entry “GlassFish ESB v2.1 - Using JavaScript Codelets to Extend BPEL 2.0 Functionality”, at http://blogs.sun.com/javacapsfieldtech/entry/glassfish_esb_v2_1_using.

ISO Duration Literals and Random Time Periods

ISO 8601 Standard, which unfortunately is not free hence is not available on-line, defines how a Duration Literal must be constructed. Duration data types in XML Schema and BPEL are both ISO 8601 Duration types. See Wikipedia, “ISO 8601”, http://en.wikipedia.org/wiki/ISO_8601, for a discussion of ISO 8601.

It will suffice to say that a Duration Literal P1Y2M3DT4H5M6S defines a time P(eriod) of 1 Y(ear), 2 M(onths), 3 D(ays), T(ime) 4 H(ours), 5 M(inutes), 6.0 S(econds). Duration literals can be abbreviated so 10 minutes can be represented as DT10M, and 10 Seconds as DT10S.

To make a BPEL process wait for a random amount of time one must first get a random time value not exceeding some maximum, construct a Duration Literal using that time value and assign the resulting literal to the BPEL Wait activity.

In this Note a JavaScript Codelet will be used obtain a random time value and convert it to ISO8610 Duration Literal. This JavaScritp Codelet will be invoked in-line in a BPEL 2.0 process.

Random ISO8601 Duration Java Script Codelet

Description

This Codelet receives a Maximum Time Value, in Milliseconds, calculates a random integer between 1 and that value, and returns this as a time period represented by an ISO 8601 Duration Literal. The Duration Literal will be returned with a maximum precision of Days (as in no Months or Years will be shown) therefore it is not advisable to set the Maximum Time Value to a value greater then approximately 30*24*60*60*1000, or 2,592,000,000 milliseconds.

Interface

The XML Schema shown below defines the structure of input and output messages.

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    targetNamespace="http://xml.netbeans.org/schema/GetRandDurSchema"
    xmlns:tns="http://xml.netbeans.org/schema/GetRandDurSchema"
    elementFormDefault="qualified">
    <xsd:element name="MaxNumMillisReq">
        <xsd:complexType>
            <xsd:sequence>
                <xsd:element
                    name="MaxNumMillis" type="xsd:int"/>
            </xsd:sequence>
        </xsd:complexType>
    </xsd:element>
    <xsd:element name="ISODurRes">
        <xsd:complexType>
            <xsd:sequence>
                <xsd:element
                    name="ISODur" type="xsd:duration"/>
            </xsd:sequence>
        </xsd:complexType>
    </xsd:element>
</xsd:schema>
```

The Abstract WSDL shown below uses this XML Schema to define messages which will be used in BPEL to interface between BPEL and JavaScript code.

```
<?xml version="1.0" encoding="UTF-8"?>
<definitions
    name="GetRandDurIFAbs"
    targetNamespace="http://j2ee.netbeans.org/wsdl/HL7Rcvr/GetRandDurIFAbs"
    xmlns="http://schemas.xmlsoap.org/wsdl/"
    xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    xmlns:tns="http://j2ee.netbeans.org/wsdl/HL7Rcvr/GetRandDurIFAbs"
    xmlns:ns="http://xml.netbeans.org/schema/GetRandDurSchema"
    xmlns:plnk="http://docs.oasis-open.org/wsbpel/2.0/plnktype">
```

```
    <types>
        <xsd:schema
            targetNamespace="http://j2ee.netbeans.org/wsdl/HL7Rcvr/GetRandDurIFAbs">
            <xsd:import
                namespace="http://xml.netbeans.org/schema/GetRandDurSchema"
                schemaLocation="GetRandDurSchema.xsd"/>
        </xsd:schema>
    </types>
    <message name="getRandDurRequest">
        <part name="sMaxNumMillisReq" element="ns:MaxNumMillisReq"/>
    </message>
    <message name="getRandDurResponse">
        <part name="sIOSDurRes" element="ns:ISODurRes"/>
    </message>
    <portType name="GetRandDurIFAbsPortType">
        <operation name="getRandDur">
            <input name="input1" message="tns:getRandDurRequest"/>
            <output name="output1" message="tns:getRandDurResponse"/>
        </operation>
    </portType>
    <plnk:partnerLinkType name="GetRandDurIFAbs">
        <plnk:role
            name="GetRandDurIFAbsPortTypeRole"
            portType="tns:GetRandDurIFAbsPortType"/>
    </plnk:partnerLinkType>
</definitions>
```

Sample Instance Documents

The following XML Instance Documents represent the structure of the input message the JavaScript Codelet expects and works with, and the structure the JavaScript Codelet is expected to return to BPEL on completion.

```
<MaxNumMillisReq
xmlns="http://xml.netbeans.org/schema/GetRandDurSchema">
  <MaxNumMillis>30000</MaxNumMillis>
</MaxNumMillisReq>

<ISODurRes xmlns="http://xml.netbeans.org/schema/GetRandDurSchema">
  <ISODur>PT0H0M20.0S</ISODur>
</ISODurRes>
```

JavaScript Code

```
;
// begin script
//
function log(vStr) {
    java.lang.System.out.println("===>>> " + vStr);
}
log("Begin script");

function randDurUpToMillis(vMillis) {
        var vDurMillis = Math.floor ( Math.random ( ) * vMillis + 1 );
        return iso8601DurFromMillis(vDurMillis);
}

function iso8601DurFromMillis(vMillis) {
        var vMillisInSec = 1000;
        var vMillisInMin = 60 * vMillisInSec;
        var vMillisInHrs = 60 * vMillisInMin;
        var vMillisInDay = 24 * vMillisInHrs;

        var vDays = Math.floor(vMillis / vMillisInDay);
        vMillis -= (vDays * vMillisInDay);
        var vHrs = Math.floor(vMillis / vMillisInHrs);
        vMillis -= (vHrs * vMillisInHrs);
        var vMins = Math.floor(vMillis / vMillisInMin);
        vMillis -= (vMins * vMillisInMin);
        var vSecs = vMillis / vMillisInSec;
```

```
        var vDur = "P" + vDays + "DT" + vHrs + "H" + vMins + "M" + vSecs + "S";
        return vDur;
}


/*
 * Example instance documents this script expects and returns
 *
<MaxNumMillisReq xmlns="http://xml.netbeans.org/schema/GetRandDurSchema">
  <MaxNumMillis>30000</MaxNumMillis>
</MaxNumMillisReq>
 *
<ISODurRes xmlns="http://xml.netbeans.org/schema/GetRandDurSchema">
  <ISODur>PT0H0M0.0S</ISODur>
</ISODurRes>
 *
 */

default xml namespace = "http://xml.netbeans.org/schema/GetRandDurSchema";
var vJSMaxNumMillisReqXML = new XML(vJSMaxNumMillisReq);
var vMaxNumMillis = vJSMaxNumMillisReqXML.MaxNumMillis;

vJSISODurRes = <ISODurRes><ISODur>PT0H0M0.0S</ISODur></ISODurRes>;
vJSISODurRes.ISODur = randDurUpToMillis(vMaxNumMillis);

log(vJSMaxNumMillisReqXML);
log("vMaxNumMillis : " + vMaxNumMillis );
log("vJSISODurRes : " + vJSISODurRes);

log("End script");
// end script

;
```

Example

Create a BPEL Module, **ISO8601DurWait**.

Create a New → XML Schema Document, **GetRandDurSchema**, and replace its content with the XSD shown in Section “Interface”. Check, validate and save.

Create a New → WSDL document, **GetRandDurIFAbs**, and replace its content with the WSDL shown in Section “Interface”. Check, validate and save.

Create a New → XML Schema, **ISO8601DurWaitTriggerSchema**, and replace its content with the following:

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema
    xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    targetNamespace="http://xml.netbeans.org/schema/ISO8601DurWaitTriggerSchema"
    xmlns:tns="http://xml.netbeans.org/schema/ISO8601DurWaitTriggerSchema"
    elementFormDefault="qualified">
    <xsd:element name="MaxDurationReq">
        <xsd:complexType>
            <xsd:sequence>
                <xsd:element
                    name="MaxDurMilliseconds" type="xsd:int"/>
            </xsd:sequence>
        </xsd:complexType>
    </xsd:element>
    <xsd:element name="DurRes">
        <xsd:complexType>
            <xsd:sequence>
                <xsd:element name="DurLiteral" type="xsd:string"/>
            </xsd:sequence>
        </xsd:complexType>
    </xsd:element>
</xsd:schema>
```

Create a New → WSDL Document, **ISO8601DurWaitTrigger**, as follows:

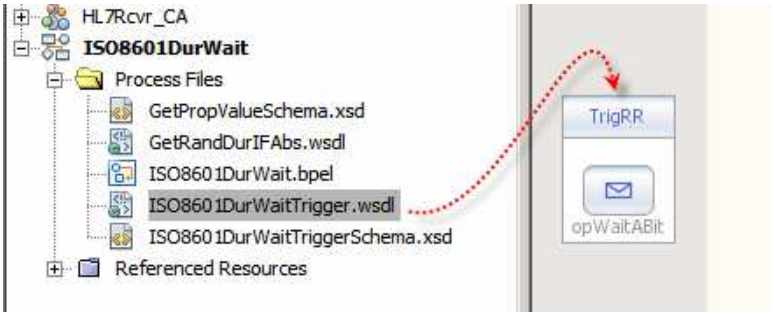
WSDL Type: **Concrete WSDL Document**
Binding: **SOAP**
Type: **Document Literal**
Operation Name: **opWaitABit**
Input:

Message Part Name: **sMaxDurReq**
Element Or Type: ISO8601DurWaitTriggerSchema → **sMaxDuration Req**

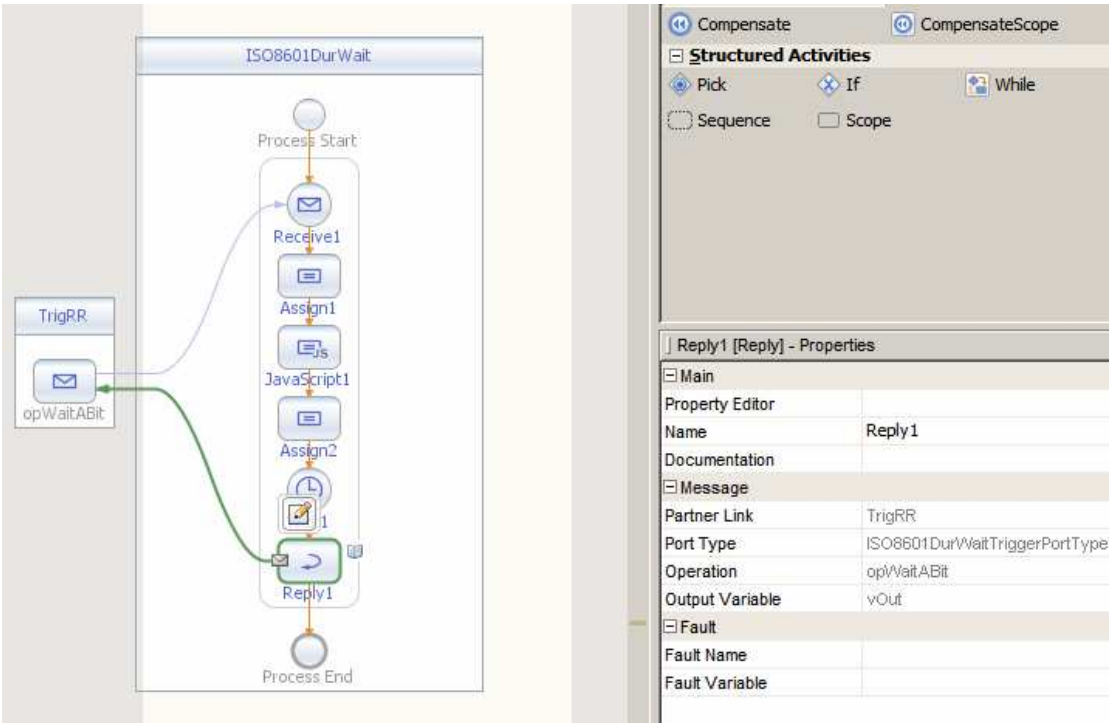
Output:

Message Part Name: **sDurRes**
Element Or Type: ISO8601DurWaitTriggerSchema → **sDurRes**

Drag WSDL ISO8601DurWaitTrigger onto the right-hand swim line and name the Partner Link TrigRR.



Add Receive, Assign, JavaScript, Assign, Wait and Reply activities to the process scope.
Connect Receive1 and Reply1 to TrigRR Partner Link and create variables vIn for the Receive1 and vOut for the Reply1.



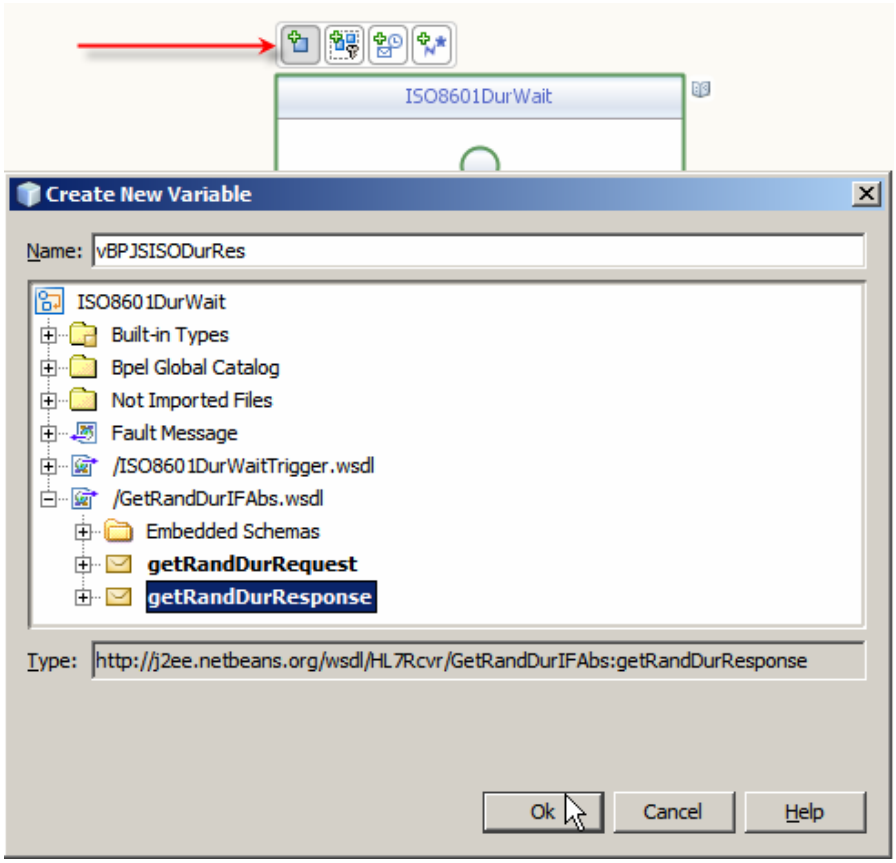
Select process scope and add two variables:

vBPJSMaxNumMillisReq of type:

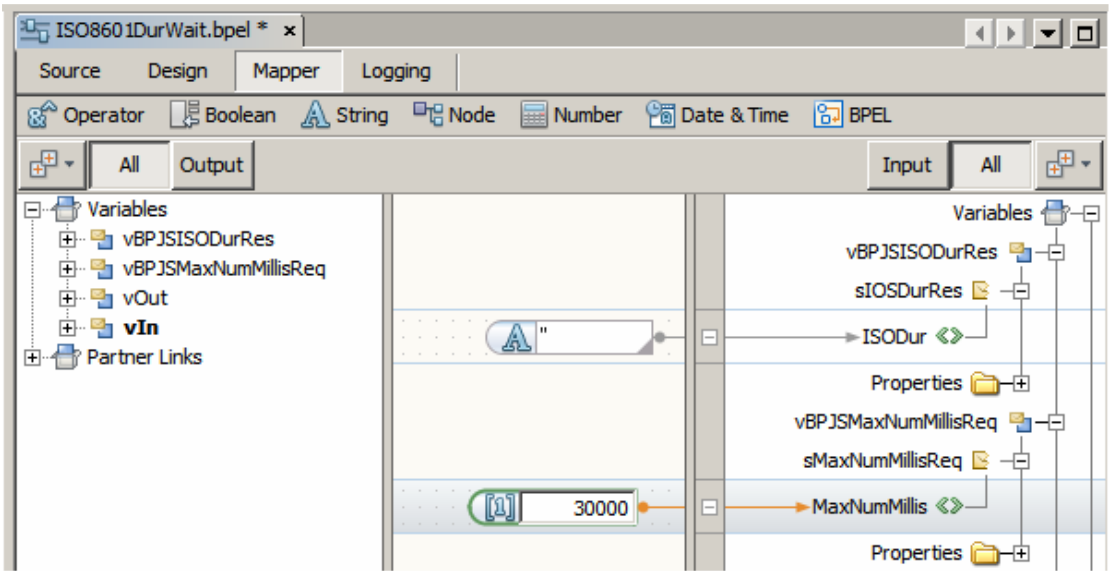
Not Imported Files → GetRandDurIFAbs → GetRandDurRequest

vBPJSISODurRes of ype:

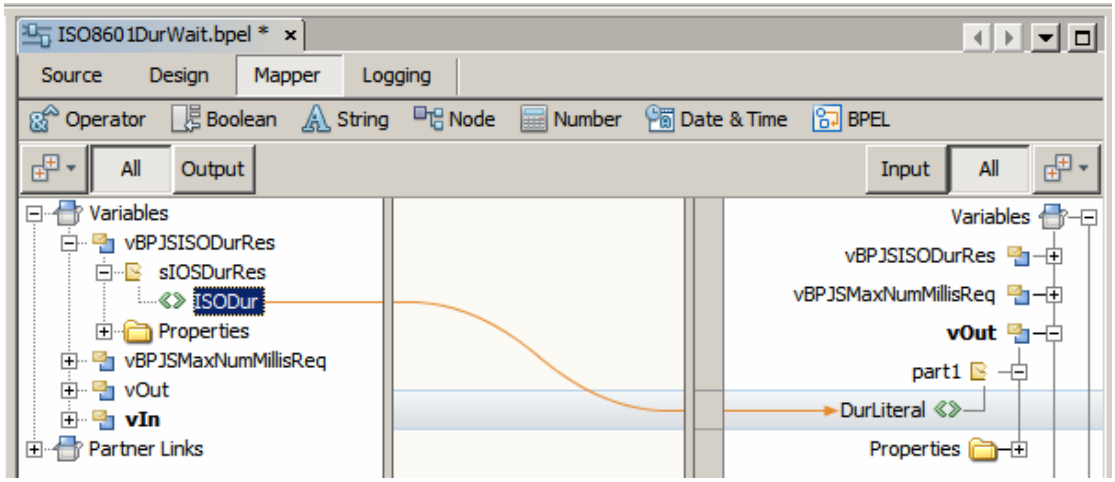
GetRandDurIFAbs → GetRandDurResponse



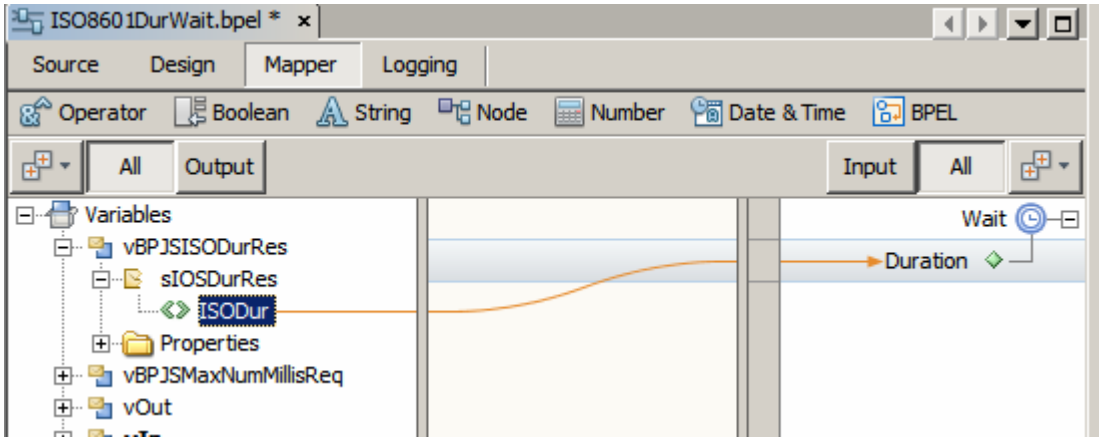
Double-click Assign1 and map as shown:



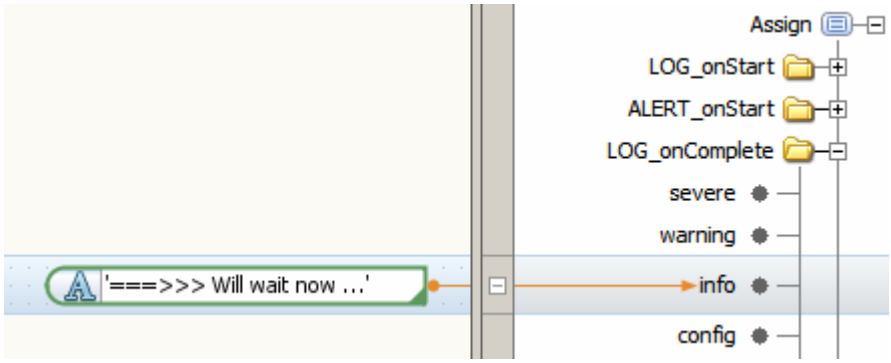
Double-click Assign2 and map as shown:



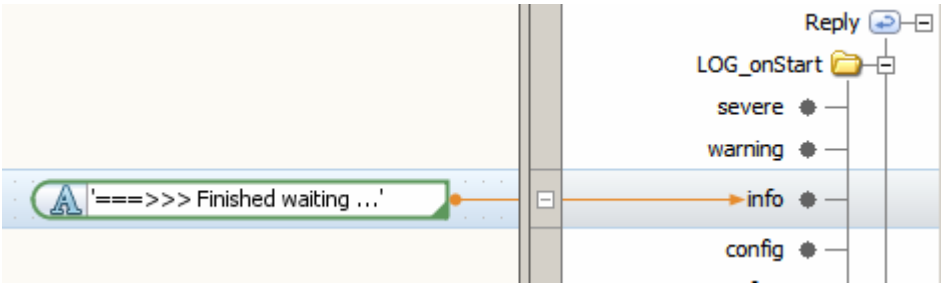
Switch to Design mode and double-click the Wait1 activity to open it in BPEL Mapper. Map as shown below.



Switch to Design mode. Select Assign2 activity, switch to Logging mode and add logging text as shown:

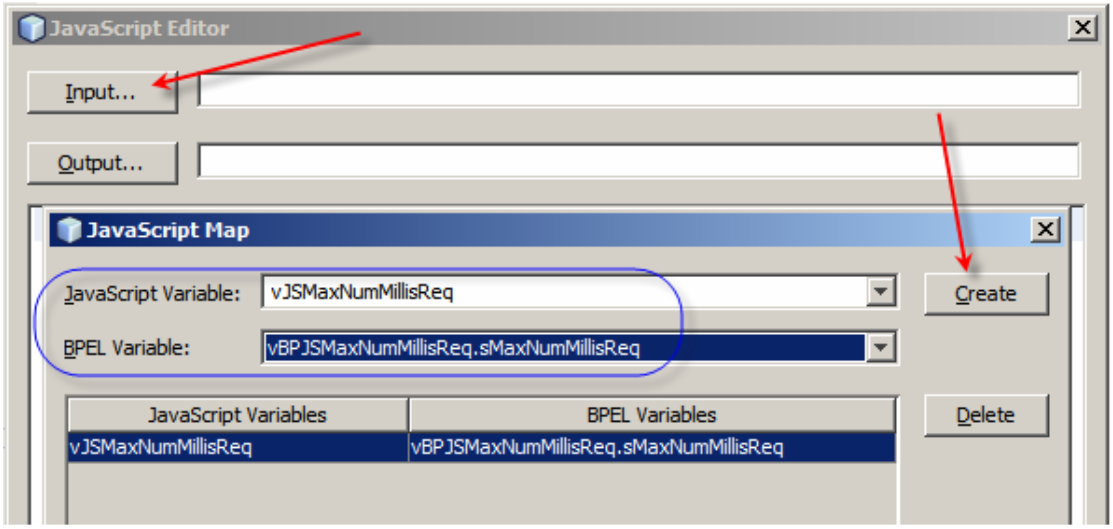


Switch to Design mode, select the Reply1 activity, switch to Logging mode and add logging text as shown:

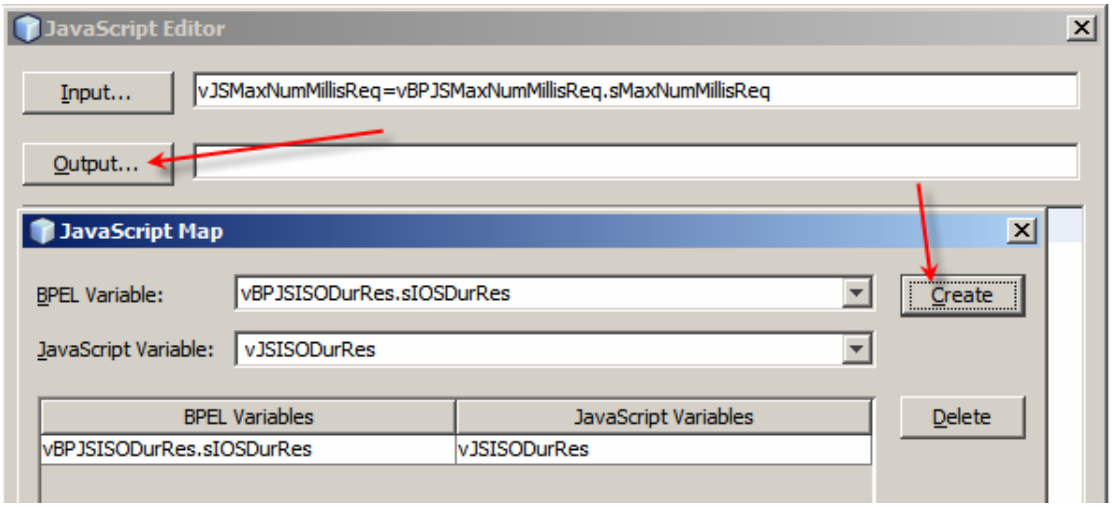


Switch to Design mode.

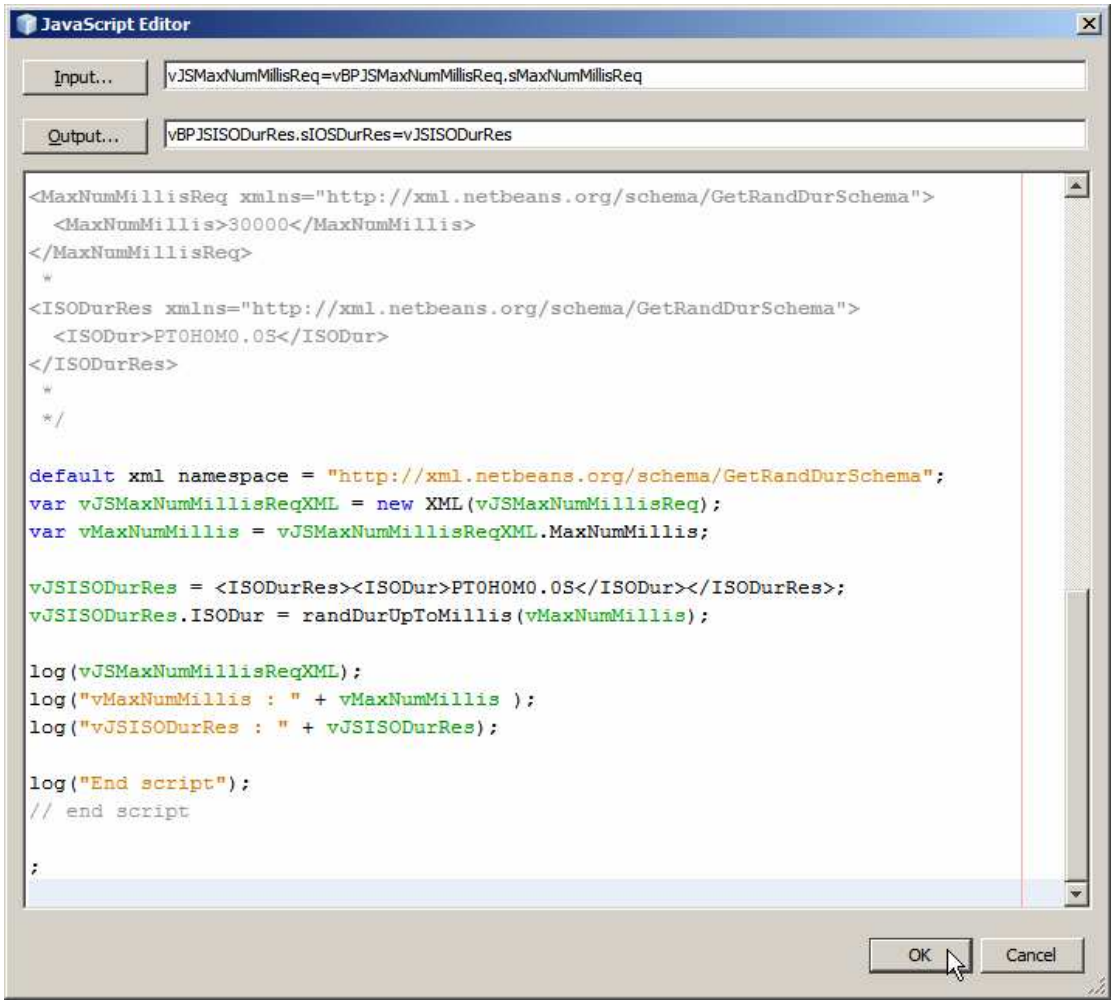
Double-click JavaScript activity to open JavaScript Editor. Click “Input…”, enter vJSMaxNumMillisReq, choose vBPJSMaxNumMillisReq.sMaxNumMillisReq, then click Create and OK.



Click “Output…”, choose vBPJSISODurRes.sISODurRes, enter vJSISODurRes, then click Create and OK.



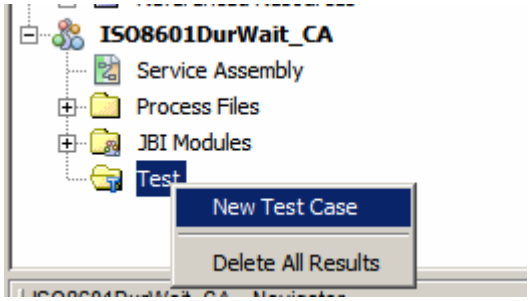
Paste the JavaScript script shown in section “JavaScript Code”, into the text box and click OK.



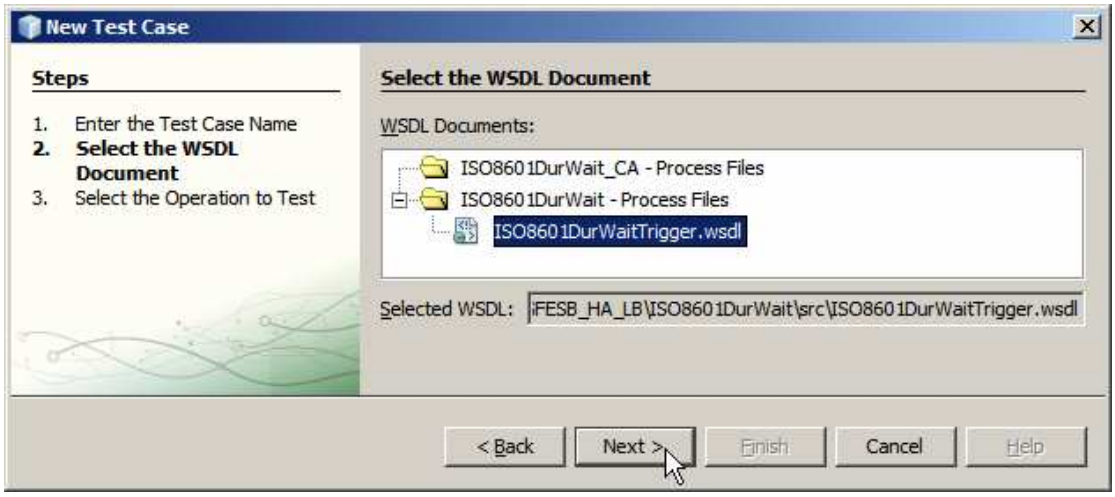
Save and Build the project.

Create New Project → Composite Application, **ISO8601DurWait_CA**. Drag the BPEL Module ISO8601DurWait onto CASA canvas, Build and Deploy.

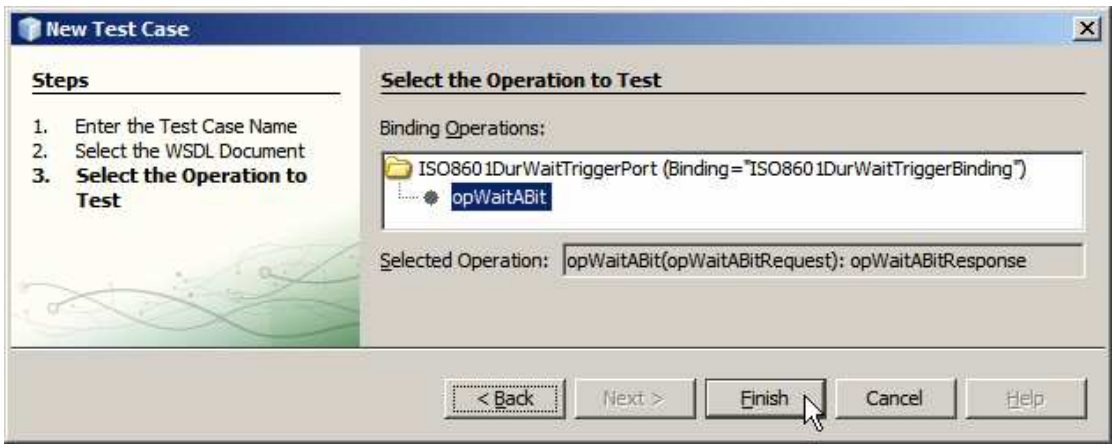
Create a New Test Case, TestCase1.



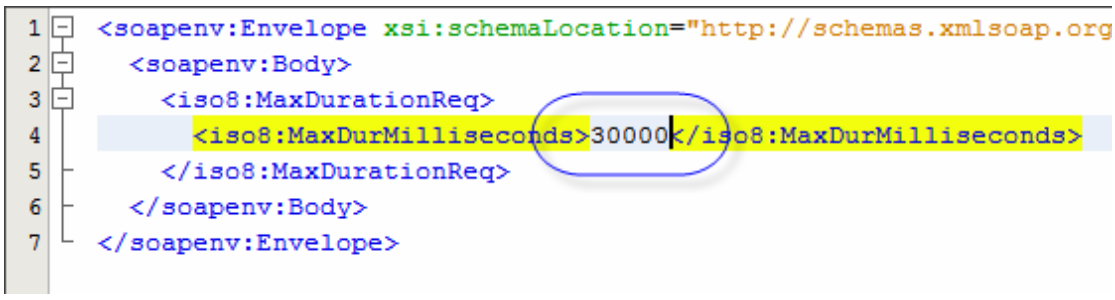
Use WSDL ISO8601DurWaitTrigger.



Choose operation opWaitABit.

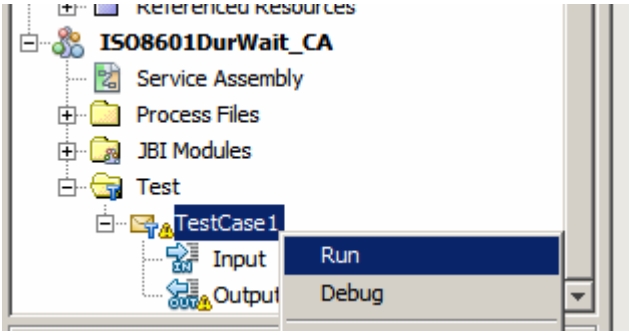


Replace “?3?” with 30000 in the SOAP request.

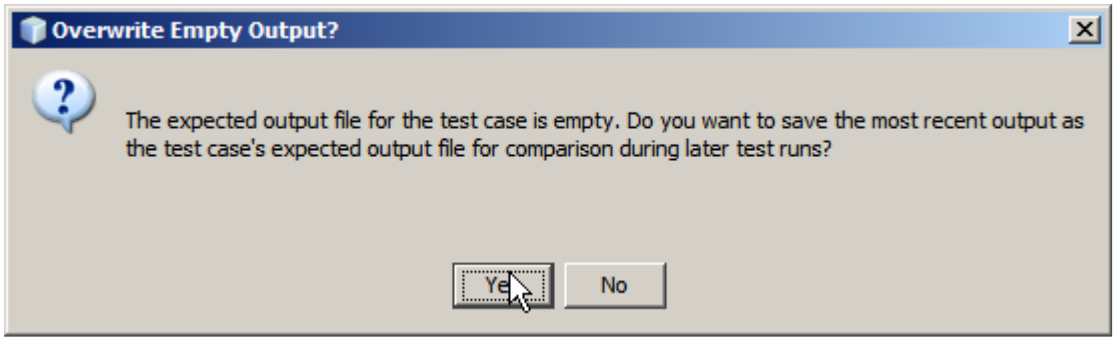


Save the modified request.

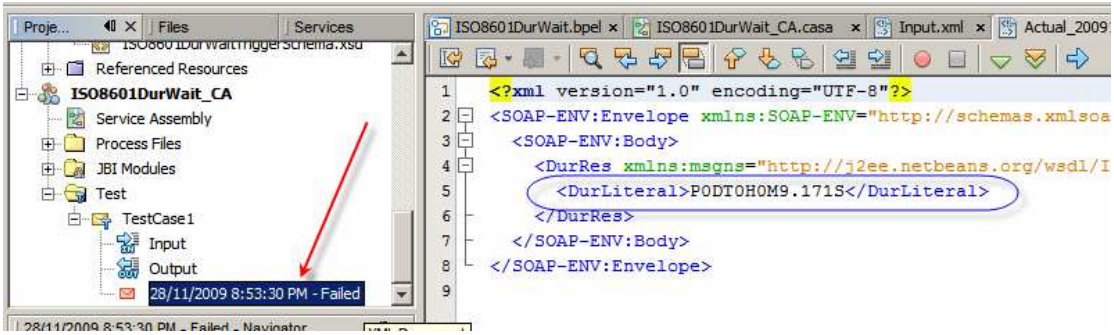
Right-click TestCase1 and choose Run.



Accept “Overwrite Empty Output” request.



Open and inspect test case output.



Locate tracer entries in the server.log and compare timestamps.

```
[#|2009-11-28T20:53:17.953+1100|INFO|sun-appserver2.1|com.sun.jbi.engine.bpel.
core.bpel.trace.BPELTraceManager|_ThreadID=62;_ThreadName=sun-bpel-engine-thre
ad-4;Process Instance Id=192.168.60.2:3fc9662:12539962bc0:-7f64;Service Assemb
ly Name=ISO8601DurWait_CA;Activity Name=Assign2;BPEL Process Name=ISO8601DurWa
it;|===>>> Will wait now ...|#]
```

```
[#|2009-11-28T20:53:27.156+1100|INFO|sun-appserver2.1|com.sun.jbi.engine.bpel.
core.bpel.trace.BPELTraceManager|_ThreadID=62;_ThreadName=sun-bpel-engine-thre
ad-4;Process Instance Id=192.168.60.2:3fc9662:12539962bc0:-7f64;Service Assemb
ly Name=ISO8601DurWait_CA;Activity Name=Reply1;BPEL Process Name=ISO8601DurWai
t;|===>>> Finished waiting ...|#]
```

About 10 seconds. Looks right ☺

This is a contrived example designed to illustrate how to use the JavaScript Codelet under discussion. The method will hold for any process where waiting for a random period of time not exceeding certain value is required.

Summary

This Note described the JavaScript Codelet which, given a maximum duration in Milliseconds, will return a random time up to that maximum duration, as an ISO8601 Duration Literal, suitable for use in the BPEL Wait activity. An example process that uses this Codelet was also developed and discussed.