Posting a message to IBM Business Process Manager Event Manager

If you want to post a message from an external system to IBM® BPM Event Manager, you must use the message structure that is described in this topic.

You can use Java Message Service (JMS) to post a message to the Event Manager. See Maintaining and monitoring IBM Business Process Manager Event Manager to learn how the Event Manager processes incoming requests.

Understanding the message structure

The message that you post to the Event Manager must contain an event name (the message event ID generated when you create an undercover agent) and it can contain parameter names and values in key-value pairs. (The topic Creating and configuring an undercover agent for a message event describes the message event ID that you should use for the event name.) The message must be structured as XML as shown in the following example:

```
<eventmsg> <!-- The process app acronym and event name are required:
```

If you do not include the snapshot name in the message, the Event Manager uses the default snapshot on the target Process Server for start message events. For intermediate message events, if you do not include the snapshot name in the message, all active snapshots receive events. To learn more about active and default snapshots, see

Configuring runtime settings for installed snapshots.

> **Note**
>
> If the value of the <value> element contains XML elements or similar content, you need to wrap the value in a CDATA tag as shown in the preceding example. For information about passing parameter values for each complex business object (variable type), see the following section.

Passing complex variable types to undercover agents

You can use undercover agents to instantiate services and BPDs automatically, without human interaction by an IBM BPM participant. When you include a message event in an IBM BPM BPD, you must assign an undercover agent to it for the message event to run at run time. The Event Manager, which is part of the IBM Process Server, handles the scheduling and queuing of undercover agents. For more information, see Undercover agents.

In addition to user-created complex business objects (variable types), the following complex business objects can be used to invoke undercover agents at run time:

Table 1. Complex business objects that can be used to invoke undercover agents at run time

Variable type	Syntax
Record	XMLDocument
Date and Time	XMLElement
Boolean	XMLNodeList
Map	ANY (default)

For example, to invoke an undercover agent using an XML message, let's say your runtime process contains a message event that waits for an incoming message. This message contains an input parameter whose value includes the Customer Name, Description, and Age.

First, create the service and then associate the service with an undercover agent (the service describes what happens when the undercover agent is invoked at run time). Then,

send the message with the following syntax:

```
<eventmsg> <event processApp="[acronym]" snapshot="[snapshot_name]" u
```

The following sections provide examples of how to pass the content of the `<value>` element. The conversion from the event XML format to a complex type is handled automatically by the IBM BPM engine.

When the Any type is used to pass a parameter value, the actual IBM BPM type must be supplied using the type attribute of the corresponding element. The `type` attribute can be omitted only when IBM BPM knows the exact `type` or when the type is `String`. The value of the attribute must be an existing IBM BPM type-or in case of an `array`, an IBM BPM type concatenated with the `[]` string at the end.

Passing IBM BPM Structured types

All structured objects are passed as XML structures, where every object property corresponds to an XML element.

For example:

Variable type: Customer - Name: String (John Doe) - Description: String (Single) - Age: Integer (30)

is mapped to:

```
XML: <value> <Name>John Doe</Name> <Description>Single</Description>
```

Keep the following important rules in mind:

- Every object property is mapped to an XML Element with the same name as the property name. The element name is case-sensitive. For example, if the property is Name, the element name must be <Name>, not <name>.
- All XML element attributes are reserved. If you pass an attribute (excluding `type`), an error is returned because the passed Document is not valid.
- When an array is passed, it is mapped as a sequence of XML elements with the same name. There are two possibilities:
  Passing root level arrays: Every object array item must be placed as content of an `<item>` element. For example:

```
<value> <item> <Name>John Doe</Name> <Description>Married</Descrip
```

Passing array properties: Every object in the object array is converted to XML using the object property name as an XML Element name. For example:

```
<value> <Name>John Doe</Name> <Description>Single</Description> <A
```

- If an object property is `null` , the corresponding element is skipped.

Passing Record type

The Record type is serialized in the same way as Structured types. However, because all values are considered of type ANY, the type information must also be passed (using the `type` attribute) in order for the correct objects to be instantiated during de-serialization.

Passing Date/Time types

The format for passing dates is `yyyy/MM/dd HH:mm:ss.S z` .

Example:

- `<value>2004/03/11 14:02:20.0 PST</value>`
- `<value>2000/02/20 11:10:20.0 GMT+6:00</value>`

When the value is converted to the Calendar Java object, it preserves the time zone, and no other modifications (such as adjusting it to the server time zone) is performed.

Passing Boolean type

The valid values for the Boolean type are `true` or `false`(case is not considered).

Example:

```
<value>TRUE</value>
```

Passing Map type

A Map type is passed to an undercover agent using the following structure:

```
<value> <entry> <key> … </key> <value> … </value> </entry> </value> F
```

Because all values and keys in this case need to be of the ANY type, the type information

must also be passed in order for the correct objects to be instantiated during deserialization. If the object is of the `String` type, the type does not need to be specified.

Passing XMLDocument type

An XML Document is passed as an XML escaped string.

Example:

```
<value> <![CDATA[ <?xml version="1.0"?> <Customer> <Name>John Doe</Na
```

Passing XMLElement type

An XML Element is passed as an XML escaped string.

Example:

Related topics:

Building a sample inbound integration

Creating implicit SOAP headers for inbound web service integrations

Posting a message to IBM Business Process Manager Event Manager

Publishing IBM Business Process Manager web services

Do you want to...

Open a ticket and download fixes at the **IBM Support Portal**

Find a technical tutorial in **IBM Developer**

Find a best practice for integrating technologies in **IBM Redbooks**

Explore, learn and succeed with training on the **IBM Skills Gateway**