IBM BPM Analytics

IBM® Business Process Manager Analytics (technology preview) enhances IBM Business Process Manager capabilities for process analytics by bringing the benefit of big data into business operations.

Attention: IBM BPM Analytics is delivered as a technology preview. This feature is provided for evaluation only and is not intended for use in a production environment.

Two scenarios are considered:

- For IBM BPM Analytics, to provide, directly within IBM Business Process Manager, enhanced process analytics features that target task workers and business manager roles in IBM BPM.
- For analytics products from other software vendors, to provide enhanced features to publish process data to external data and analytics solutions that originate from IBM, customers, and partners.

This implementation enhances the existing IBM BPM Dynamic Event Framework (DEF) to publish business events in JSON message format. The reformatted and aggregated messages are serialized, the query-optimized documents are stored on an Elasticsearch server, and the documents are fed to dashboards through the Kibana tool.

Where to find files

The main components of IBM BPM Analytics are the following archives: `BPMEventEmitter.war`, `EventSummaryAgent.tar` and `BPMDashboardKibana.zip`. You can find `BPMEventEmitter.war` in the *`install_root`*`/BPM/analytics` directory, where *`install_root`* is the directory where you installed IBM Business Process Manager. Please download `EventSummaryAgent.tar` and `BPMDashboardKibana.zip` from https://github.com/ibmbpm/bpm_analytics

Within the `BPMEventEmitter.war` archive, the `BPMEventEmitter.yml` configuration file is in the `dmgr_profile_name\config\cells\cell_name\clusters\cluster_name\analytics\config` and `dmgr_profile_name\config\cells\cell_name\nodes\node_name\servers\server_name\analytics\config\` directories.

Installing Elasticsearch and Kibana

You start with installing the Elasticsearch and Kibana tools.

Before you begin

IBM® Business Process Manager Analytics might not support the current versions of the binary files that are available from the https://www.elastic.co/ website. To use IBM BPM Analytics in your IBM BPM deployment environment, make sure to comply with the following software requirements.

- Elasticsearch and Kibana
  Versions 5.1.x, 5.4.x, and 5.5.x are supported.
  Important: By default, Elasticsearch and Kibana are not secured. To use these tools with real business data, make sure that your installation is sufficiently secured. Multiple, widely used solutions are available, form secure proxy solutions, such as Nginx, free community plug-ins, and subscription-based vendor solutions. Make sure to choose the best solution to meet your operational requirements. For more information, see the Nginx website or the website of the selected solution, and the Security for IBM BPM Analytics page.
- Java Runtime Environment (JRE)
  IBM JRE 7 (Java 7): Both the EventSummaryAgent and BPMEventEmitter applications require Java 7. Therefore, IBM BPM JRE must also be Java 7. IBM JRE 8 (Java 8) is also supported.

Procedure

1. Install Elasticsearch by following the official instructions on the elastic Docs Installation page.
2. Install Kibana by following the official instructions on this elastic Docs Kibana User Guide page.

What to do next

After you Elasticsearch and Kibana are installed, you can start to enable IBM Business Process Manager Analytics.

Enabling IBM BPM Analytics

To take benefit of IBM® Business Process Manager Analytics, you must first configure the Dynamic Event Framework (DEF) and then install and configure the BPMEventEmitter application.

About this task

You configure IBM BPM Analytics in two phases:

1. You configure the Dynamic Event Framework (DEF) to subscribe to IBM BPM events that are emitted from process applications.
2. You install and configure the BPMEventEmitter application to emit IBM BPM events to Elasticsearch or popular big data analytics tools, such Apache Spark, by using Kafka.

A script is provided for you to run these configurations with the most efficient options already predefined. This script is the *install_root*`/BPM/Lombardi/tools/def/EnableBPMAnalytics.py` file. This script has the following main effects:

- It calls the `SampleConfigureJSONEventsToJMS.py` script, in the same directory, for the following actions:
  - Create a Java Message Service (JMS) queue connection factory with default name is `monitorCF`, a JMS queue with default name `monitorQueue`, and a JMS activation specification with default name `defAS` as messaging resources in the application server cell scope. If resources with the same name exist in the cell scope, a numeric suffix is appended to the name of the new resource, for example `monitorCF1`, `monitorQueue1`, `defAS1`.
  - Define a destination with default name `monitorDestination` on the IBM BPM deployment environment bus.
  - Configure JSON native events to JMS.
  - Reload event listeners.
- This script installs and configures the BPMEventEmitter application on the single cluster if you have a single cluster topology, or on the support cluster if you have a three-cluster topology.
- This script also changes the class loading order to **Classes loaded with local class loader first (parent last)**.

The application does not start automatically, though, because a manual post job to configure the Elasticsearch/Kafka connection information is necessary.
Important: If you do not use Performance Data Warehouse (PDW), it is better to disable it before you enable IBM BPM Analytics. For more information, see the Disabling tracking data generation web page.

Procedure

1. Make sure that the IBM BPM deployment environment is created.
2. Start the deployment manager and the deployment environment.
3. From the *install_root*`/profiles/Dmgr01/bin` directory, run the `wsadmin -lang jython -f <install_root>/BPM/Lombardi/tools/def/EnableBPMAnalytics.py`
4. Restart the Deployment Environment. (If the BPMEventEmitter application just installed does not start, it doesn't matter. The Elasticsearch/Kafka connection information must be configured before the application can be started.) Validate that DEF was enabled successfully by starting a process in process portal. Select Service Integration > Service Integration Bus Browser in admin console, click into the bus destination (default name is "monitorDestination") which DEF uses. The queue depth should be larger than 0.
5. Optional: If you have multiple deployment environments in one cell, edit the appropriate fields in `SampleConfigureJSONEventsToJMS.py` script.

   For more information, see Generating JSON native DEF events.

6. Optional: Specify your register interest in receiving events.

   For more information, see Generating JSON native DEF events.

7. Edit the property values in the `BPMEventEmitter.yml` configuration file as appropriate for your server topology to configure the Elasticsearch or Kafka connection information.

   The `BPMEventEmitter.yml` file is in the `dmgr_profile_name\config\cells\cell_name\nodes\node_name\servers\server_name\analytics \config\` and `dmgr_profile_name\config\cells\cell_name\clusters\cluster_name\analytics\config` directories.

   Edit the values for the cluster (the single cluster in a single cluster topology or the support cluster in a three-cluster topology) and for the servers that belong to the cluster, depending on your server topology. Follow the instructions in Editing the BPMEventEmitter configuration file.

8. Optional: If you want to use a message hub service, configure the message broker to create a message hub service and a topic by using the Bluemix console.

   Follow the instructions in Configuring the message broker.

9. Full resynchronize the nodes.
10. Start the BPMEventEmitter application from the WebSphere® Application Server administration console. After you log in, expand **Applications > Application Types > WebSphere enterprise applications** in the navigator panel, then select **BPMEventEmitter_war** and click **Start**.

The default address is `http://<Server_Address>:9060/admin`.

The application retrieves messages from the DEF monitor event queue and sends them to Elasticsearch or Kafka. If something goes wrong, please check cluster member ’s SystemOut.log (the single cluster in a single cluster topology or the support cluster in a three-cluster topology).

Results

After a successful start, you can see status messages that are similar to the following example in the SystemOut.log.

```
[5/18/17 12:15:16:466 IST] 00000152 LifeCycleMana I Application Name:IBM BPM
Analytics
[5/18/17 12:15:16:466 IST] 00000152 LifeCycleMana I Application
Version:8.5.7.201703
[5/18/17 12:15:16:467 IST] 00000152 LifeCycleMana I Build Level:20170518_28
[5/18/17 12:15:49:509 IST] 0000b44b LifeCycleMana I   I:  DEF2ES MDB started.
[5/18/17 12:15:49:509 IST] 0000b44b ConfigConnect I   I: Kafka connection
disabled.
[5/18/17 12:15:49:727 IST] 0000b44b ConfigConnect I   I: ElasticSearch
connection created.
```

If the application fails to start, see Troubleshooting IBM BPM Analytics.

Editing the BPMEventEmitter configuration file

To configure the Elasticsearch or Kafka connection information, you set various properties in the configuration file of the BPMEventEmitter application.

About this task

This configuration file is in YAML format.
Note: If Kafka is not installed on your environment, keep the properties to their default values and leave the `enabled` property in the `kafkaConfiguration` section to `false`.

Procedure

1. Update the appropriate values for your environment and save the updated file back to the WAR package.

   `enabled`
   When the `enabled` parameter in the `kafkaConfiguration` section is set to `true`, the raw JSON events are written to the Kafka topic. Set this property to `false` if Kafka is not installed.
   `bpmCellName`
   The field value of the generated message. If you want to analyze messages from different IBM® BPM clusters, specify different names across different clusters.
   `esConfiguration`
   The configuration for Elasticsearch servers.
   `index`
   The name of the Elasticsearch index where the generated data is stored.
   `hosts`
   Lists all server addresses in the Elasticsearch cluster.

   - If you did not enable SSL/TLS security on your Elasticsearch server, use your IP address directly, for example `192.168.0.1:9200,127.0.0.1:9200`
   - If you enabled SSL/TLS security, add the `https` prefix to indicate that the HTTPS protocol must be used, for example `https://192.168.0.1:9200`, `https://127.0.0.1:9200`

   `username`, **`password`**
   Use these parameters to enable basic authentication on your Elasticsearch cluster. For the password field, you can use plain text or an encoded password. To encode your password, you can use the EventSummaryAgent application by running the `encodePassword` function, which returns the encoded value, as follows.

```
EventSummaryAgent -encodePassword elastic
<xor>cXxraGFIdw==
```

If you enabled SSL/TLS on your Elasticsearch configuration, you must set the **`httpsTrustType`** value to reflect the trust type that you used. The following options are available:

`Default`

The agent takes the HTTPS communications that are accepted by the JVM default settings.

`ALL`

The application accepts all HTTPS communications. Preferably, use this value for test purposes only.

`CRT`

This option provides the CA certificate `.crt` file for the HTTPS connections that the application accepts. You must set the **`trustFileLocation`** value as an absolute address. Ensure that the IBM BPM server can access that file. For example:

```
httpsTrustType: CRT
trustFileLocation: /opt/IBM/BPM/elasticSearch.crt
```

`JKS`

This option is similar to the CRT option and provides the Java keystore `.jks` file for the HTTPS connections that the application accepts. You must set the **`trustFileLocation`** value as an absolute address. Ensure that the IBM BPM server can access that file. For example:

```
httpsTrustType: JKS
trustFileLocation: /opt/IBM/BPM/elasticSearch.jks
```

`hostnameVerifier`

This property takes a Boolean value. In a production environment, leave it at `false`. If your test environment uses a CRT certificate that contains a wrong host or IP address, you can set **`hostnameVerifier`** to `true` for testing purposes only.

2. Configure security.

   See Security for IBM BPM Analytics.

3. Optional: If you want to use a message hub service, configure the message broker.

   See Configuring the message broker.

Configuring the message broker

If you choose Bluemix MessageHub as your message broker provider, the topic is not created automatically. You create a message hub service and a topic by using the Bluemix console. A code sample is provided to help you get started.

About this task

Currently, two kinds of message broker providers are supported: Apache Kafka (https://kafka.apache.org/) and Bluemix MessageHub. Bluemix MessageHub is a Kafka cloud service, therefore it shares the configuration section with Apache Kafka. For Apache Kafka, if the topic does not exist, it is created automatically. But for Bluemix MessageHub, you need to create a topic by using the Bluemix console.

Procedure

1. Log in to the IBM Bluemix console at https://console.bluemix.net/dashboard/services/
2. Create a message hub service.
3. View the service credentials and set `kafka.bootstrap.servers` and `apikey` to your Kafka configuration.
4. Create a topic, for example `bpm-monitor-topic`.

   By default, the message broker is disabled. To enable it, change the `enabled` value to `true`. All Kafka support properties can take a `kafka` prefix.

   For example, you can set the Kafka standard `bootstrap.servers` property to `kafka.bootstrap.servers`. If you need to user multiple servers, they can be separated by commas.

   To enable MessageHub as a message broker, you can use this code sample.

```
kafkaConfiguration:
    #================================
    #Set 1. Sample for using Kafka
    #================================
    #kafka.bootstrap.servers: localhost:9092
    monitor.topic: bpm-monitor-topic
    enabled: true
    # Enable the following properties when kafka security(SSL) is on.
Do not fill in any value when SSL is off.
    #kafka.security.protocol: SSL
    #kafka.ssl.truststore.location:
    #kafka.ssl.truststore.password: test1234

    #kafka.ssl.keystore.location:
    #kafka.ssl.keystore.password: test1234
    #kafka.ssl.key.password: test1234
```

```
    #====================================
    #Set 2. Sample for Bluemix MessageHub
    #====================================
    kafka.bootstrap.servers: "kafka01-prod02.messagehub.services.eu-
gb.bluemix.net:9093,kafka03-prod02.messagehub.services.eu-
gb.bluemix.net:9093"
    apiKey: "eS8gBqby1Zd**********************TnSxF6bRRC6"
    kafka.client.id: kafka-java-console-sample-producer
    kafka.security.protocol: SASL_SSL
    kafka.sasl.mechanism: PLAIN
    kafka.sasl.jaas.config:
org.apache.kafka.common.security.plain.PlainLoginModule required
username="USERNAME" password="PASSWORD";
    kafka.ssl.protocol: TLSv1.2
    kafka.ssl.enabled.protocols: TLSv1.2
    kafka.ssl.endpoint.identification.algorithm: HTTPS
    kafka.acks: '-1'
```

What to do next

You can now start the BPMEventEmitter application.

Installing, configuring, and running EventSummaryAgent

The EventSummaryAgent application monitors the new events that are added to the configured Elasticsearch index and generates summary events.

Before you begin

Before you configure EventSummaryAgent, prepare your work as follows:

- Identify the target Elasticsearch server hosts. If security is enabled on the Elasticsearch server, find out the user name, password, and SSL/TLS settings.
- Identify which Elasticsearch index is used to store the data that is generated by the BPMEventEmitter application.
- Make sure that a working Java virtual machine (JVM) is installed and set the `JAVA_HOME` environment variable.
- Although not mandatory, it is a good practice to configure a different Elasticsearch index for the source index (the **`esSourceConfiguration`** parameter) and for the summary target index (the **`esTargetConfiguration`** parameter). It helps you maintain indexes more easily when you observe that the data increases more rapidly than your initial expectation.

About this task

The main purpose of the EventSummaryAgent application is to aggregate events. However, if you set the `archive/enabled` parameter to `true` in the configuration file, EventSummaryAgent also archives event data to ObjectStore files after events are aggregated. For more information about archiving and restoring events, see Archiving and restoring.

Procedure

1. To install and update the configuration file, extract the EventSummaryAgent.tar file from the package and save it to the server where you will run this agent.

   The sample configuration file is the `EventSummaryAgent/conf/EventSummaryAgent.yml` file, in YAML format.

2. Edit the property values as appropriate for your environment, directly in the file or in a copy of it.

   `facadeType`
   This parameter can take only the Elasticsearch value.
   `pickerTimeGap`

This property defines a time gap that is used internally to prevent random data order. The suggested time is at least 10 seconds and the default value is 60 seconds. The value is expressed in milliseconds.

- **enableBusinessDataAggregator**
  This property enables the default business data aggregator, which aggregates the groups for tracking business data fields for a process instance. The default value is true. For more information about tracking business data, see Handling tracked business data.
- **QualityOfService**
  This property defines the quality of service that is used to control the agent behavior when errors are received. You can set this property to one of the following modes:

  - **strictMode**: The agent retries when an error is received so as to prevent data loss on summary types.
  - **ignoreError**: The agent ignores the error and continues processing. If an error occurs, the content for that message might be missing in the generated summary of events. Moreover, if that message is the only one that is related to a certain process or task, the entire process or task might be missing in the generated summary.

3. **esSourceConfiguration**: This property defines the configurations for the storage of Elasticsearch raw events.
4. **esTargetConfiguration**: This property defines the configurations for the storage of Elasticsearch combined events.
5. **hosts**: Lists all server addresses in the Elasticsearch cluster.
   - If you did not enable SSL/TLS security on Elasticsearch servers, use your IP address directly. For example:`192.168.0.1:9200,127.0.0.1:9200`
   - If you enabled SSL/TLS security, add the `https` prefix to indicate that the HTTPS protocol must be used. For example: `https://192.168.0.1:9200`, `https://127.0.0.1:9200`
6. **index**: This property is used as the name of the Elasticsearch index, which receives the events that are generated by the BPMEventEmitter application and stores the generated data.
7. **username**, **password**
   Use these parameters to enable basic authentication on your Elasticsearch server configuration. For the **password** field, you can use plain text or an encoded password. To encode your password, you can use the `encodePassword` function of EventSummaryAgent, which returns the encoded value, as follows:
   ```
   EventSummaryAgent.sh/bat -encodePassword elastic
   <xor>cXxraGFIdw==
   ```
   If you enabled SSL/TLS on your Elasticsearch server configuration, you must set the **httpsTrustType** value to reflect the trust type that you used. The following options are available.
   - Default

   The agent takes the HTTPS communications that are accepted by the JVM default settings.
   ALL
   The application accepts all HTTPS communications. Preferably, use this value for test purposes only.

CRT

This option provides the CA certificate `.crt` file for the HTTPS connections that the application accepts. You must specify the **`trustFileLocation`** value as an absolute address. Ensure that the IBM® BPM server can access that file. For example:

```
httpsTrustType: CRT
trustFileLocation: /opt/IBM/BPM/elasticSearch.crt
```

JKS

This option is similar to the CRT option and provides the Java Keystore `.jks` file for the HTTPS connections that the application accepts. You must specify the **`trustFileLocation`** value as an absolute address. Ensure that the IBM BPM server can access that file. For example:

```
httpsTrustType: JKS
trustFileLocation: /opt/IBM/BPM/elasticSearch.jks
```

`hostnameVerifier`

This property takes a Boolean value. In a production environment, leave it at `false`. If your test environment uses a CRT certificate that contains a wrong host or IP address, you can set **`hostnameVerifier`** to true for testing purposes only.

8. Configure security.

   See Security for IBM BPM Analytics.

9. Use the command line parameters as follows.
   - On Linux:
   - `EventSummaryAgent.sh [-configFile <configFileLocation>] | -encodePassword <passwordValue> | -version | -restore | -help`

   - On Windows:
   - `EventSummaryAgent.bat [-configFile <configFileLocation>] | -encodePassword <passwordValue> | -version | -restore | -help`

   - **`-configFile`***<configFileLocation>*
     The optional **`-configFile`** parameter specifies the path to the configuration file that you created. You can use this parameter when you start the agent. If you do not specify any path, the default path is install_root/conf/EventSummaryAgent.yml
   - **`encodePassword`***<passwordValue>*
     Use this parameter to generate an encoded password as indicated in step 2.
   - **`-version`**
     This parameter prints version information to screen.
   - **`-restore`**
     This parameter starts the EventSummaryAgent application in restore mode. For more information about restoring events, see Archiving and restoring.
   - **`-help`**
     This parameter prints help information to screen.

For any issues, see Troubleshooting IBM BPM Analytics.

10. Start the EventSummaryAgent application by running the following command from the `install_root/bin` directory.
    - On Linux:

      ```
      ./EventSummaryAgent.sh
      ```

      or

      ```
      ./EventSummaryAgent.sh -configFile ../conf/EventSummaryAgent.yml
      ```

    - On Windows:

      ```
      ./EventSummaryAgent.bat
      ```

      or

      ```
      ./EventSummaryAgent.bat -configFile ../conf/EventSummaryAgent.yml
      ```

If the agent starts successfully, the log entry is similar to the following one:

```
<time stamp>com.ibm.bpm.mon.oi.combine.EventSummaryAgentdoJob
INFO: CWMCD2007I: Starting to retrieve process or activity events from the
data source.
```

This message indicates that the agent is ready to begin processing the messages. If necessary, you can stop the EventSummaryAgent application by pressing Ctrl-C from the command window. If the application is running in the background, you can stop it by entering the **`kill -2 <process_id>`** command.

High availability considerations: The EventSummaryAgent application supports High Availability(HA) in active-passive modes. It means that you can start two EventSummaryAgent instances with the same configuration. The first instance runs in active mode, the second instance in passive mode. If EventSummaryAgent detects that both an active and a passive instance exist, no additional instance starts. If the active instance is stopped or corrupted, the passive instance takes over and automatically switches to active mode. In this case, you can start another passive instance as a backup.

Importing dashboard definitions

After you have installed, configured, and started the BPMEventEmitter and EventSummaryAgent applications, you use the Kibana tool to import dashboard definitions so that data that reflect user tasks, process performance, business data, and so on, can be displayed in dashboards.

Before you begin

The `BPMDashboardKibana.zip` archive should be downloaded from https://github.com/ibmbpm/bpm_analytics. When you open the `BPMDashboardKibana.zip` archive, you get two JSON files that are used to define dashboards in Kibana: `BPMDefaultDashboard.json` and `HiringSampleDashboard.json`.
Before you can import the `BPMDefaultDashboard.json` dashboard definition to Kibana, the following requirements must be met.

1. Make sure that the BPMEventEmitter and EventSummaryAgent applications are running.
2. Make sure that you have a process instance with at least one user task completed. Otherwise, due to a Kibana limitation, you might receive an error message indicating that the saved **`field`** parameter is invalid and prompting you to select a new field. For more information, see the issue description on the Field stats API dependency page.

Procedure

1. Optional: Create an index alias.

   If you are not using `bpm-summary` as the index name, you need to map all the indexes that you used to the `bpm-summary` alias. You can find detailed API information on the Index Aliases page of the Elastic website.

   The following example uses a REST API in Kibana Dev Tools to set `bpm-summary` as the alias of index `bpm-summary2`.

   ```
   POST /_aliases
   {
       "actions" : [
           { "add" : { "index" : "bpm-summary2", "alias" : "bpm-summary" }
   }
       ]
   }
   ```

2. Create index patterns.
   a. In the Kibana window, click **Management > Index Patterns**
   b. [Kibana 5.1.x, 5.4.x] Click the + button

[Kibana 5.5.x and later version] Click the "Create Index Pattern" button.

c. [Kibana 5.1.x, 5.4.x] Enter `bpm-summary` in the **Index name or pattern** field, clear the **Index contains time-based events** check box, and click **Create**.
[Kibana 5.5.x and later version] Enter `bpm-summary*` in the **Index pattern** field, click "advanced options" link, enter `bpm-summary` in the **Index pattern ID** field. Select the option "**I don't want to use the Time Filter**" in the "Timer Filter field name" drop-down list, and click **Create**.

The page for your index is displayed.

d. Enter `duration` in the search field.

Two field names are returned in the Fields column of the search results table: **`processTotalDuration`** and **`activityTotalDuration`**.

e. Click the  Edit button at the end of the **`processTotalDuration`** line.
f. From the **Format** list, select **Duration**, from the **Input Format** list, select **Milliseconds**, and from the **Output Format** list, select **Minutes**.

These options turn the **`processTotalDuration`** unit from milliseconds to minutes.

g. Repeat the two previous steps for the **`activityTotalDuration`** parameter.

3. Import the dashboard definition.
   a. In the Kibana window, select **Management > Saved Objects**.
   b. Click **Import** and select the `BPMDefaultDashboard.json` file.

   After importing, the Edit Saved Objects page shows 5 extra Dashboards, 6 extra Searches, and 25 extra Visualizations.

4. To open the imported dashboard in Kibana, select **Dashboard** and open IBM BPM **Process Performance**.

   You can then display the dashboards in turn by clicking each tab. These dashboards are designed only for user tasks. System tasks and other activities are not taken into account.

   - The Process Performance dashboard is the default dashboard to display process-related statistics.
   - The Task dashboard is the default dashboard to display user-task related statistics.
   - The Team Performance dashboard is the default dashboard to display team-related statistics.
   - The KPI dashboard is empty. You can add your custom KPI visualization here.
   - The Business Data dashboard is also empty. You can add your custom data visualization here.

In the next step, you populate the Business Data dashboard with data from the Hiring sample.

5. Optional: Import the Hiring Sample dashboard.

   This dashboard is provided as a sample Business Data dashboard for the Hiring Sample.

   . In the EventSummaryAgent/conf/EventSummaryAgent.yml file, enable the sample business data aggregator.
   `enableBusinessDataAggregator:true`

   a. Restart the EventSummaryAgent application.
   b. In IBM® Process Portal, run a process instance for Standard HR Open New Position in the Hiring Sample and complete it.

      Important: You must complete process instances so that all the raw data of the Elasticsearch index is included.

   c. Click **Management > Index Patterns**, select the `bpm-summary` index, and click the  Refresh button.

      Kibana retrieves the new fields for the business data that is generated in the Elasticsearch index.

   d. In Kibana, click **Management > Saved Objects**, click **Import**, and select the `HiringSampleDashboard.json` file.
   e. Click **Business At a Glance** to see the custom Business Data dashboard for the Hiring Sample.

Handling tracked business data

When process variable fields are set to automatic tracking in IBM® Process Designer, you can have the business data of tracked groups aggregated to a single event document.

Through IBM Process Designer, the fields of process variables can be declared as “Performance Tracking”. The tracked fields that belong to that process are automatically recorded as a tracking group if the **Enable Autotracking** check box is selected. For example, if `loanAmount`, `loanee_name`, `loan_age`, and `status` are defined as auto-tracked fields in Process Designer, the Dynamic Event Framework (DEF) event includes these tracked fields, and the **`trackedFields`** attribute of the activity event includes the attribute names, as follows:

```
"trackedFields": {
            "loanAmount.integer": 2000,
            "loanee_name.string": "Brian",
            "loan_age.integer": 35,
            "status.string": "Origination"
          }
```

Enabling the business data aggregator

By default, the business data aggregator is enabled in the EventSummaryAgent configuration file. In the `EventSummaryAgent/conf/EventSummaryAgent.yml` file, the **`enableBusinessDataAggregator`** parameter is set to `true`. This default value means that the Process Summary event includes business data as two new fields: **`mergedTrackedFields`** and **`trackedFields`**. For each process instance, the event maintains one document for all tracking groups. If the tracked fields are updated through the process flow, the document reflects the latest values.

If the process includes a subprocess or linked process, that subprocess or linked process belongs to the main process. The main process still creates one single document for the business data. In the following example, `AT_Loan_record1495615322291` is the name of the tracking group that belongs to the main process, and `AT_linked_loan_record` is the name of the tracking group for the linked process.

```
"trackedFields": {
            "AT_Loan_record1495615322291": {
              "loanAmount.integer": 2000,
              "region.string": "UK",
              "status.string": "Origination",
              "loanee_name.string": "Brian",
              "loanee_age.integer": 35
            },
            "AT_linked_loan_record": {
              "loanAmount.integer": 2000,
              "loanee_name.string": "Brian",
              "loan_age.integer": 35,
              "status.string": "Origination"
            }
          }
```

Correlating tracking groups

When subprocesses or linked processes exist, the fields that are tracked by the main process and those that are tracked by the subprocesses or linked processes are stored in the different tracking groups, but sometimes in the business logic, these tracking fields might stand for the same business entity. In this case, you might need to correlate the tracking groups that belong to the same starting process instance. As shown in the following configuration example, you can configure the correlation field name per application and per business process definition (BPD). The following rules apply:

- You cannot assign multiple correlation fields to the same process and same application.
- You cannot correlate tracking groups across process instances.
- Use the field name as the correlation value. Field values are determined by the latest generated activity event.
- You can use the (*) wildcard as the **`processApplicationName`** or **`processName`** value.

Configuration example

In the Bank CC Loan (sub) process of the Test Emit Monitor Event application, use the `loanee_name.string` tracked field as the correlation. In the Test Order application, all processes use the `order_id.string` tracked field as the correlation.
Note: Tracked field names must include the data type as a suffix because if a track field is defined in different tracking groups by the same name but a different type, that tracked field cannot be correlated. For example, if `loanAmount` is the field name and `integer` is its data type, the tracked field must be named `loanAmount.integer`.

```
innerProcessCorrelation:
  - processApplicationName: Test Emit Monitor Event
    processName: Bank CC Loan (sub)
    correlationFieldName: loanee_name.string
  - processApplicationName: Test Order
    processName: '*'
    correlationFieldName: order_id.string
```

After the correlation field `loanee_name.string` is set, the example in Enabling the business data aggregator is merged as follows:

```
"mergedTrackedFields": {
            "loanAmount.integer": 2000,
            "status.string": "Origination",
            "loanee_name.string": "Brian",
            "loan_age.integer": 35,
            "loanee_age.integer": 35,
            "region.string": "UK"
          }
```

The `loan_age.integer` and `loanee_age.integer` fields in the different tracking groups cannot be merged to a single tracked field because their names are different. Tracked fields that have the same names across different tracking groups are merged to a single value.

Drilling down the Process Performance and Task dashboards

When the **`enableBusinessDataAggregator`** parameter is set to `true` in the EventSummaryAgent configuration file, the Process Performance, Task, and Team Performance dashboards support drill-down by the predefined tracked fields as filter criteria. You can enter the filter at the top of the dashboard. For example, if you want to show only the task status for one process application, like Test Emit Monitor Event, you can enter the `Test Emit Monitor Event` process application name as the filter, and the dashboard will reflect the filter immediately.

Note: Because Kibana is powered by Elasticsearch, it supports the Lucene query string syntax and also uses some Elasticsearch filter capabilities. If you want to drill down into the Test Emit Monitor Event process application and focus only on the tasks that are related to the predefined tracked field `"Region" equals to "UK"`, you can enter the filter as follows:

```
processApplicationName:"Test Emit Monitor Event" AND
trackedFields.region.string:UK
```

Troubleshooting IBM BPM Analytics

If you encounter issues when working with IBM® Business Process Manager Analytics, this information can help you resolve them.

- Troubleshooting non generation of events
  If the Kibana window shows that the dashboard does not update correctly or has no new events coming in, work on these issues from the BPMEventEmitter and EventSummaryAgent applications.
- Troubleshooting exceptions
  When you work with the BPMEventEmitter and EventSummaryAgent applications, you might encounter errors that would be related, for example, to class loading order, protocol security, Java version, message queue setting, and so on. Learn here how to resolve such issues.

Troubleshooting non-generation of events

If the Kibana window shows that the dashboard does not update correctly or has no new events coming in, work on these issues from the BPMEventEmitter and EventSummaryAgent applications.

Procedure

1. Check whether the Dynamic Event Framework (DEF) event can be generated by IBM® BPM.
   a. Stop the BPMEventEmitter application temporarily: from the WebSphere® Application Server administration console, select **Service integration > Service Integration Bus Browser >** ***your_bus_name*** **> Destinations >** ***your_DEF_destination*** **> Queue points**.

      On that page, because the DEF event consumer, BPMEventEmitter, has been stopped, you can see the DEF events in the queue, and the queue depth is increasing.

   b. Otherwise, check the steps in Generating JSON native DEF events to make sure that DEF was successfully enabled.

      If the queue JNDI name is wrong in the `SampleConfigureJSONEventsToJMS.py` script, no explicit error message is printed to `SystemOut.log`. Instead, two first-failure data-capture (FFDC) logs are created.

Table 1.

Exception	Error message
FFDC in SystemOut .log	[8/28/17 13:49:42:903 CST] 0000018f FfdcProvider W com.ibm.ws.ffdc.impl.FfdcProvider logIncident FFDC1003I: FFDC Incident emitted on <ffdc_location>\<ffdc_file_name> com.ibm.bpm.def.impl.listeners.JmsTextDefEventListener 001 [8/28/17 13:49:42:918 CST] 0000018f FfdcProvider W com.ibm.ws.ffdc.impl.FfdcProvider logIncident FFDC1003I: FFDC Incident emitted on <ffdc_location>\<ffdc_file_name> com.ibm.bpm.def.impl.EventPointImpl 0002b
Exception in FFDC content	[8/28/17 13:49:42:903 CST] FFDC Exception:javax.naming.NameNotFoundException SourceId:com.ibm.bpm.def.impl.listeners.JmsTextDefEventListener ProbeId:001 Reporter:com.ibm.bpm.def.impl.listeners.JmsTextDefEventListener @92f35ae5 javax.naming.NameNotFoundException: Context: joshua138Cell01/clusters/BPM.AppCluster, name: jms/monQueue_wrong: First component in name monQueue_wrong not found. [Root exception is org.omg.CosNaming.NamingContextPackage.NotFound: IDL:omg.org/CosNaming/NamingContext/NotFound:1.0] at com.ibm.ws.naming.jndicos.CNContextImpl.mapNotFoundException(CNContextImpl.java:4564) at com.ibm.ws.naming.jndicos.CNContextImpl.doLookup(CNContextImpl.java:1822) at com.ibm.ws.naming.jndicos.CNContextImpl.doLookup(CNContextImpl.java:1777) at com.ibm.ws.naming.jndicos.CNContextImpl.lookupExt(CNContextImpl.java:1434) at com.ibm.ws.naming.jndicos.CNContextImpl.lookup(CNContextImpl.java:616) at com.ibm.ws.naming.util.WsnInitCtx.lookup(WsnInitCtx.java:165) at com.ibm.ws.naming.util.WsnInitCtx.lookup(WsnInitCtx.java:179)
Exception in FFDC content	[8/28/17 13:49:42:903 CST] FFDC Exception:com.ibm.bpm.def.spi.DefEventListenerException SourceId:com.ibm.bpm.def.impl.EventPointImpl ProbeId:0002b Reporter:com.ibm.bpm.def.impl.EventPointImpl@11bfa00c com.ibm.bpm.def.spi.DefEventListenerException: javax.naming.NameNotFoundException: Context: joshua138Cell01/clusters/BPM.AppCluster, name: jms/monQueue_wrong: First component in name monQueue_wrong not found. [Root exception is

Table 1.	
Exception	**Error message**
	org.omg.CosNaming.NamingContextPackage.NotFound: IDL:omg.org/CosNaming/NamingContext/NotFound:1.0] at com.ibm.bpm.def.impl.listeners.JmsDefEventListener.onFormattedEvent(JmsDefEventListener.java:87) at com.ibm.bpm.def.impl.EventPointImpl.fireDirectSync(EventPointImpl.java:116) at com.ibm.bpm.def.impl.EventPointImpl.access$200(EventPointImpl.java:44) at com.ibm.bpm.def.impl.EventPointImpl$1.run(EventPointImpl.java:198)

2. Check whether the ProcessEvent and ActivityEvent documents can be generated on the Elasticsearch server.
    a. Get the number of ProcessEvent objects by running the following command from a Kibana Dev Tools prompt: `GET bpm-events/ProcessEvent/_count`
    b. Get the number of ActivityEvent objects by running the following command from a Kibana Dev Tools prompt: `GET bpm-events/ActivityEvent/_count`
    c. Start the BPMEventEmitter application.

    You should see the number of the process events and activity events increase. Otherwise, check the BPMEventEmitter log, which is included in the `SystemOut.log` file on the server that hosts the BPMEventEmitter WAR package.

3. Check whether the ProcessSummary and ActivitySummary documents can be generated on the Elasticsearch server.
    a. Stop the EventSummaryAgent application.
    b. Get the number of ProcessSummary documents by running the following command from a Kibana Dev Tools prompt: `GET bpm-summary/ProcessSummary/_count`
    c. Get the number of ActivitySummary documents by running the following command from a Kibana Dev Tools prompt: `GET bpm-summary/ActivitySummary/_count`

    Further actions on processes or activities increases the number of documents.

    d. After the number of ProcessEvent and ActivityEvent documents increases, start the EventSummaryAgent application to check whether it works.

    After the EventSummaryAgent application has been started, this application starts picking up raw event documents and the number of combined type documents increases. If this number does not increase, check the logs in the *`install_root`*`/log` folder of the EventSummaryAgent application.

Note: The default idle time before EventSummaryAgent resumes is 1 minute. For more information, see Installing, configuring, and running EventSummaryAgent.

Troubleshooting exceptions

When you work with the BPMEventEmitter and EventSummaryAgent applications, you might encounter errors that would be related, for example, to class loading order, protocol security, Java version, message queue setting, and so on. Learn here how to resolve such issues.

Procedure

- If either of the following exceptions is received in the `SystemOut.log` file, work from the WebSphere® Application Server administration console to change the class loader order of the WAR module to **Classes loaded with local class loader first (parent last)**.
    - Exception 1:
      ```
      CNTR0020E: EJB threw an unexpected (non-declared) exception
      during invocation of method "onMessage" on bean
      "BeanId(BPMEventEmitter_war#
      BPMEventEmitter.war#BPMEventEmitterMDB, null)".

      Exception data: java.lang.NoSuchMethodError:
      org/apache/http/HttpHost.create(Ljava/lang/String;)Lorg/apache/ht
      tp/HttpHost;
      ```
    - Exception 2:
      ```
      0000b44b LifeCycleMana E class
      com.ibm.bpm.mon.oi.LifeCycleManageBeanstartMethod E: Create
      elastic search index mapping failed with
      exception.java.io.IOException: listener timeout after waiting for
      [10000] ms
      ```
- If either of the following exception is thrown, ensure that you specified the correct protocol in the **`esConfiguration.hosts`** parameter in the BPMEventEmitter configuration file.

  If Security Socket Layer (SSL) is enabled on the Elasticsearch server, the prefix for the transfer protocol must be `https`. This exception is thrown in the log files of both the Elasticsearch server and of the IBM® BPM system (SystemOut.log).

    - In the Elasticsearch server log:
      ```
      [2017-05-18T11:26:16,793][WARN
      ][o.e.x.s.t.n.SecurityNetty4HttpServerTransport] [PIyAGtg] caught
      exception while handling client http traffic, closing connection
      [id: 0xfd4fb4bb, L:0.0.0.0/0.0.0.0:9200 ! R:/9.112.249.138:61011]
      io.netty.handler.ssl.NotSslRecordException: not an SSL/TLS
      record:
      48454144202f6d6f6e69746f72323031373034204854502f312e310d0a486f7
      3743a206a6f736875616465763130303a393230300d0a436f6e6e656374696f6e
      3a204b6565702d416c6976650d0a557365722d4167656e743a204170616368652
      d487474704173796e63436c69656e742f342e312e3220284a6176612f312e372e
      30290d0a0d0a at
      io.netty.handler.ssl.SslHandler.decode(SslHandler.java:907)
      [netty-handler-4.1.6.Final.jar:4.1.6.Final]at
      ```

```
io.netty.handler.codec.ByteToMessageDecoder.callDecode(ByteToMess
ageDecoder.java:411) [netty-codec-4.1.6.Final.jar:4.1.6.Final]
```

- In the WebSphere Application Server log:

```
Caused by: org.apache.http.ConnectionClosedException: Connection
closed at
org.apache.http.nio.protocol.HttpAsyncRequestExecutor.endOfInput(
HttpAsyncRequestExecutor.java:344)at
org.apache.http.impl.nio.DefaultNHttpClientConnection.consumeInpu
t(DefaultNHttpClientConnection.java:261)at
org.apache.http.impl.nio.client.InternalIODispatch.onInputReady(I
nternalIODispatch.java:81)at
org.apache.http.impl.nio.client.InternalIODispatch.onInputReady(I
nternalIODispatch.java:39)at
org.apache.http.impl.nio.reactor.AbstractIODispatch.inputReady(Ab
stractIODispatch.java:114)
```

- Check the version of the Java runtime environment (JRE).

  Since the BPMEventEmitter application has been supporting the Java7 runtime environment (JRE), the IBM BPM server must also use Java7 or later. If IBM BPM uses Java6, the BPMEventEmitter application cannot start and throws the following exception in the SystemOut.log file.

```
[6/28/17 15:57:03:091 EDT] 00000142 CompositionUn E WSVR0194E:
Composition unit WebSphere:cuname=BPMEventEmitter_war in BLA
WebSphere:blaname=BPMEventEmitter_war failed to start.
[6/28/17 15:57:03:092 EDT] 00000142 MBeanHelper E Could not invoke an
operation on object:
WebSphere:name=ApplicationManager,process=server1,platform=proxy,node=n
odename1,
version=8.5.5.11,type=ApplicationManager,mbeanIdentifier=ApplicationMan
ager,cell=nodename1Node01Cell,spec=1.0
because of an mbean exception: com.ibm.ws.exception.RuntimeWarning:
Error while processing references for EJB-in-WAR:
com.ibm.ejs.container.EJBConfigurationException:
Bean class com.ibm.bpm.mon.oi.BPMEventEmitterMDB could not be loaded
```

- Check the Dynamic Event Framework (DEF) queue setting.

  If the DEF queue depth is not properly configured, events might be lost in the process. The default threshold is 50000. If the BPMEventEmitter application did not start or the event producer created events faster than the consumer can process, the number events above the threshold are lost. Set the threshold value to match the peak time of your system to avoid failure of the event consumer. For more information of this issue, see this IBM Support SIMPLimitExceededException technote.

- Check the subscription filter and edit it if necessary.

  If you subscribed to all the events in the advanced version and the `com.ibm.bpm.def.JmsDefEventListenerFactory` API is used to configure DEF subscription, you might get an exception in the SystemOut.log file.

```
[8/28/17 9:55:32:834 IST] 000000cb BPMEventEmitt E
CWMCD1020E: The BPEL or SCA message detected
but they are not supported by BPM Analytics in this version.
```

Edit the subscription filter as follows:

```
subscriptions=[
'*/*/BPD/*/*/*/*'
]
```

For more information, see Generating JSON native DEF events.

- Start the deployment environment before you run the script that enables IBM BPM Analytics.

  If you do not start the deployment environment before you run the **`wsadmin -lang jython -f <Install_Root>/BPM/Lombardi/tools/def/EnableBPMAnalytics.py`** command, you might see the following error message in the command line output:

  ```
  WASX7017E: Exception received while running file "
  EnableBPMAnalytics.py"; exception information:
  com.ibm.bsf.BSFException: exception from Jython:
  Traceback (innermost last):
    File "<string>", line 117, in ?
    File "SampleConfigureJSONEventsToJMS.py", line 524, in ?
    File "SampleConfigureJSONEventsToJMS.py", line 517, in
  createWASResources
  IndexError: index out of range: 0
  ```

  To fix the issue, manually remove the created resources, including the JMS queue connection factory, the queue destination, the JMS queue, the JMS activation specification, and the DEF listener, that are listed in command line output, start the deployment environment, and rerun the script.

Maintaining indexes

To optimize your use of Elasticsearch indexes, you can create separate indexes to avoid having one single, very large index. You can also rebuild an event summary index at any time.

About this task

Maintaining the index scale

As defined by the IBM® BPM Analytics system administrator, you might not want to keep all of the historical data in one single Elasticsearch index. One option is to create separated indexes for each period of time, for example one index per month or per quarter, depending on the number of events that the IBM BPM system created. In this case, you can wrap the multiple Elasticsearch indexes to one common index alias. For example, if you have index `monitor201703` and index `monitor201704`, you can use the following REST API to create the `source` index alias.

```
POST /_aliases
{
    "actions" : [
        { "add" : { "index" : "monitor201703", "alias" : "source" } },
        { "add" : { "index" : "monitor201704", "alias" : "source" } }
    ]
}
```

Index aliases that point to multiple indexes are read-only. Therefore, these index aliases cannot be used as the destination of the BPMEventEmitter and the EventSummaryAgent applications. However, because the EventSummaryAgent application only reads data and does not update it, it is safe to use such an index alias as a source event in the Elasticsearch configuration properties. In the previous example, you could use the `source` index alias as the Elasticsearch source configuration.

```
hosts: localhost:9200
    index: source
```

Rebuilding the event summary index

As long as raw events that were created by the BPMEventEmitter application are available, you can rebuild the summary event index at any time. Here is an example of the EventSummaryAgent configuration file. The combined summary event is set as the Elasticsearch target server. A user name and password are defined because security has been set on the Elasticsearch server. The progress is recorded on the target Elasticsearch server.

```
qualityOfService: strictMode

esSourceConfiguration:
    hosts: localhost:9200
    index: bpm-events
    username: elastic
    password: <xor>d3hrdXJEeXU=
    httpsTrustType:
    trustFileLocation:
    hostnameVerifier:
```

```
esTargetConfiguration:
    hosts: localhost:9200
    index: bpm-summary
    username: elastic
    password: <xor>d3hrdXJEeXU=
    httpsTrustType:
    trustFileLocation:
    hostnameVerifier:

  esIndex: oiprogress
  esType: readcursor
  esTaskIndex: restore_task_index

archive:
  enabled: false
```

Working with this example, you can rebuild the event summary index as follows.

Procedure

1. Stop the EventSummaryAgent application.
2. Delete the `bpm-summary` event summary index.
3. Delete the `oiprogress` progress record index.
4. Restart the EventSummaryAgent application.

Archiving and restoring

You can archive raw events to ObjectStore and restore them later to the Elasticsearch server if necessary.

To prevent data loss when Elasticsearch crashes or when data has been deleted by accident, IBM® BPM Analytics enables you to archive raw events from the Elasticsearch server to an ObjectStore file. Later, you can restore data from ObjectStore to the Elasticsearch server by using the EventSummaryAgent application.

Configurations for archiving and restoring

The `archive` and `ObjectStoreConfiguration` sections of the EventSummaryAgent configuration file are related to archiving and restoring data.
The `archive` section:

`enabled`
The enabled parameter controls whether the archive function is turned on or off. Its default value is false.

`minBulksize`, `maxBulkSize`
Use these parameters to specify the minimum and maximum number of raw events to be merged to one object file in ObjectStore.

The `objectStoreConfiguration` section:

IBM BPM Analytics supports only the `openstack-swift` API with the `temp auth v1.0` and `keystone auth v3.0` authentication mechanisms. The provided sample is configured for the `swift temp auth` method. Here is an example for the Bluemix object store:

```
providerOrApi: swift-v3
    containerName: BPMAnalyticArchive
    userId: 3eef42de3xxxxxxxxxxxx
    password: xxxxxxxx
    auth_url: https://identity.open.softlayer.com
    domainName: 136xxxx
    projectId: object_storage_1c8707e5_xxxx_xxxx_xxxx_9c9xxxxxxx1
```

For more information about the `openstack-swift` API, see the Auth System Openstack page.

How and when to enable archiving

If you decide to archive raw events to ObjectStore files, you activate this capability by setting the `archive/`**`enabled`** parameter to `true` and the `objectStoreConfiguration` parameters for the `swift temp auth` method. You can then start the EventSummaryAgent application with that configuration. From then on, the EventSummaryAgent application saves data to ObjectStore files as soon as the aggregation jobs, if any, are completed.

```
Jun 15, 2017 10:02:48 AM com.ibm.bpm.mon.oi.combine.EventSummaryAgent doJob
INFO: I: Start getting process/activity events from the data source.
```

```
Jun 15, 2017 10:02:49 AM com.ibm.bpm.mon.oi.archive.ArchiveHandler run
INFO: I: Archive process started.
Jun 15, 2017 10:02:49 AM com.ibm.bpm.mon.oi.archive.ArchiveHandler run
INFO: I: Archive process finished.
Jun 15, 2017 10:02:49 AM com.ibm.bpm.mon.oi.combine.EventSummaryAgent doJob
INFO: I: Start getting process/activity events from the data source.
Jun 15, 2017 10:02:49 AM com.ibm.bpm.mon.oi.archive.ArchiveHandler run
```

How to restore events to the Elasticsearch server

To start the restoring process, run the EventSummaryAgent application with the **`-restore`** parameter.
Important: Before you start the EventSummaryAgent application in restore mode, ensure that all the tasks to be restored are stored on the Elasticsearch index through the **`esTaskIndex`** parameter. Otherwise, if no restore task is defined, EventSummaryAgent switches to aggregation mode directly.

- On Linux:

  ```
  EventSummaryAgent.sh [-configFile <configFileLocation>] | -
  encodePassword <passwordValue>
  | -version | -restore
  ```

- On Windows:

  ```
  EventSummaryAgent.bat [-configFile <configFileLocation>] | -
  encodePassword <passwordValue>
  | -version | -restore
  ```

Remember: The destination server of the restoration process is controlled by the **`esSourceConfiguration`** parameter of the EventSummaryAgent configuration and by the **`esIndex`** parameter of the BPMEventEmitter configuration. Because the Elasticsearch source connection in the EventSummaryAgent configuration might use a read-only index, the restoration process uses the connection information from the EventSummaryAgent **`esSourceConfiguration`** parameter and the BPMEventEmitter index name. For more information about index aliases, see Maintaining indexes.
When the BPMEventEmitter application starts, it saves the configuration to the Elasticsearch index that is configured as follows. The restoration process reads the configuration in that index. You can double-check before you restore the archived events.

```
esTaskIndex: restore_task_index
```

The restoration process runs for the specified time range. You can also use the **`esTaskIndex`** parameter to specify a JSON task event of type `restore_task_type`. In the following example, all the events that occurred in June 2017 will be restored from the ObjectStore Elasticsearch server.

```
{“startTime”:”YYYY-MM”, “endTime”:”YYYY-MM”}
POST restore_task_index/restore_task_type
{
  "startTime":"2017-06",
  "endTime":"2017-06"
}
```

Only months are supported but you can specify months that straddle two years. In the following example, all the events that occurred in 2016 and in January 2017 are restored.

```
POST restore_task_index/restore_task_type
{
  "startTime":"2016-01",
  "endTime":"2017-01"
}
```

Example

In this example, the command starts the restoration process and specifies the name and location of the configuration file.

```
EventSummaryAgent.bat/sh -configFile <configurationFileLocation> -restore
IBM BPM Event Summary Agent

Jun 16, 2017 3:40:36 PM com.ibm.bpm.mon.oi.restore.RestoreWorker restore
INFO: I: Restore process is running.
Jun 16, 2017 3:40:41 PM com.ibm.bpm.mon.oi.restore.RestoreWorker restore
INFO: I: Restore process finished.
Jun 16, 2017 3:40:41 PM com.ibm.bpm.mon.oi.combine.EventSummaryAgent doJob
INFO: I: Start getting process/activity events from the data source.
```

After the tasks that are stored through the **`esTaskIndex`** are restored, the EventSummaryAgent application automatically returns to the normal aggregation process. This behavior saves system administrators the trouble of having to step in if the restoration process takes a long time.

Appendices

To help you work with IBM® BPM Analytics, you can use this complementary material.

- Security for IBM BPM Analytics
  It is of utmost importance that you enable security for all communication channels, including Elasticsearch, Apache Kafka, and ObjectStore, to prevent network attacks.
- Generating JSON native DEF events
  For better performance, you can use the provided IBM BPM script to configure the DEF to generate JSON native events. You can edit that script for multiple deployment environments and for your subscription preferences.
- Installing BPMEventEmitterWAR on the IBM BPM server
  After you have prepared the IBM BPM environment, you install the BPMEventEmitterWAR package and start the application.
- Optional: Disabling the Dynamic Event Framework (DEF) and skipping KPI data
  After you are finished with the IBM BPM Analytics, you can disable the Dynamic Event Framework (DEF) to avoid performance loss. You can also omit KPI data.
- Optional: Disabling task event generation
  By default, the Dynamic Event Framework (DEF) generates task event notification. To improve performance, you can choose to disable this feature by adding a configuration property.
- Generate XML DEF events (not recommended)
  Although this is not the preferred practice, you can generate XML DEF events.

Security for IBM BPM Analytics

It is of utmost importance that you enable security for all communication channels, including Elasticsearch, Apache Kafka, and ObjectStore, to prevent network attacks.

Security on the Elasticsearch server

Before you enable Elasticsearch security, you can leave all the security-related fields empty or remove them from the configuration file to disable those settings. After you have enabled Elasticsearch security, you might need to provide the correct values for the security-related fields. For example:

```
hosts: https://<some_address>:9200
# the following properties should be enabled when elastic search security is
on
username: elastic
password:<xor>cXxraGFIdw==
httpsTrustType: CRT
trustFileLocation: /opt/IBM/BPM/elastcisearch.crt
hostnameVerifier:false
```

This configuration means that the Elasticsearch server is protected with Transport Layer Security (TLS) and basic authentication. The path to the CRT certification file is `/opt/IBM/BPM/elasticsearch.crt` and the host/IP address to the server is correct.

Java 2 security

If you enabled Java 2 Security on your server, after you start the BPMEventEmitter application, you might receive an error message such as the following one.

```
SecurityManag W   SECJ0314W: Current Java 2 Security policy reported a
potential violation of Java 2 Security Permission. Refer to the InfoCenter
for further information.
Permission:
modifyThreadGroup : Access denied ("java.lang.RuntimePermission"
"modifyThreadGroup")
```

In this case, you must grant access to the application in the `was.policy` file to. For more information, see Configuring the was.policy file for Java 2 security in the WebSphere® Application Server documentation.

For the BPMEventEmitter application, you must update the `profile_root/config/cells/cell_name/applications/BPMEventEmitter_war.ear/deployments/BPMEventEmitter_war/META-INF/was.policy` file as follows:

```
//
// Template policy file for enterprise application.
// Extra permissions can be added if required by the enterprise application.
```

```
//
// NOTE: Syntax errors in the policy files will cause the enterprise
application FAIL to start.
//       Extreme care should be taken when editing these policy files. It is
advised to use
//       the policytool provided by the JDK for editing the policy files
//       (WAS_HOME/java/jre/bin/policytool).
//
grant codeBase "file:${application}" {
  permission java.lang.RuntimePermission "stopThread";
  permission java.lang.RuntimePermission "modifyThread";
  permission java.lang.RuntimePermission "modifyThreadGroup";
  permission java.lang.RuntimePermission "createSecurityManager";
};

grant codeBase "file:${jars}" {
};

grant codeBase "file:${connectorComponent}" {
};

grant codeBase "file:${webComponent}" {
};

grant codeBase "file:${ejbComponent}" {
};
```

Generating JSON native DEF events

For better performance, you can use the provided IBM® BPM script to configure the Dynamic Event Framework (DEF) to generate JSON native events. You can edit that script for multiple deployment environments and for your subscription preferences.

About this task

Dynamic Event Framework (DEF) events can be in JSON or in XML format. Normally, DEF events in native JSON format perform better than XML DEF events. Using native JSON DEF events improves overall performance.

IBM BPM includes the following script for you to configure the DEF to generate JSON native events:
`install_root_dir/BPM/Lombardi/tools/def/SampleConfigureJSONEventsToJMS.py`

This script has the following effects:

- Create the Java Message Service (JMS) queue connection factory in the cell scope. The factory default name is `monitorCF`.
- Create the JMS queue in the cell scope. The queue default name is `monitorQueue`.
- Create a JMS activation specification in the cell scope. The specification default name is `defAS`.
- Define a destination on the IBM BPM deployment environment bus. The destination default name is `monitorDestination`.
- Configure JSON native events to JMS.
- Reload event listeners.

Procedure

To generate JSON native DEF events, follow these steps.

1. Make sure that the deployment environment is created.
2. Start the deployment manager and the deployment environment.
3. Optional: If you have multiple deployment environments in one cell, edit the following fields before you run the `SampleConfigureJSONEventsToJMS.py` script.

   `deploymentEnvironmentName`
   A string value that specifies the name of deployment environment that you want to configure, for example: `de2`

   `ServiceIntegrationBusName`

   A string value that refers to the name of service integration bus to associate the JMS queue connection factory and JMS queue with, for example `de2.Bus`

jmsQueueConnectionFactory_AuthorizationAlias

A string value that refers to the authentication alias to use for the JMS queue connection factory. This is usually the authorization alias of the deployment environment, for example `DeAdminAlias` or `jmsQueueConnectionFactory_AuthorizationAlias = 'BPMAdminAlias_de2`

4. Optional: Specify your register interest in receiving events.

   Before you run the sample script, edit the subscription section. Each subscription is a single string with a / separator for each of the seven-part keys. Subscriptions are separated by commas. The seven-part keys are Application Name / Version / ComponentType / Component Name / Element Type / Element Name / Nature.

   To listen on every event for all applications, use the wildcard character as in the following example:

   ```
   subscriptions=[
   '*/*/*/*/*/*/*'
   ]
   ```

   The following example shows how you might register to receive events for the Hiring Sample.

   ```
   subscriptions=[
   'HSS/*/BPD/*/PROCESS/*/*',
   'HSS/*/BPD/*/ACTIVITY/*/*',
   'HSS/*/BPD/*/GATEWAY/*/*',
   'HSS/*/BPD/*/EVENT/*/*'
   ]
   ```

5. From the install_root/profiles/Dmgr01/bin directory, run the following command.

   ```
   wsadmin -lang jython -f
   <Install_Root>/BPM/Lombardi/tools/def/SampleConfigureJSONEventsToJMS.py
   ```

6. Restart the deployment environment.
7. Check that Dynamic Event Framework (DEF) was enabled successfully.
   a. Start a process in Process Portal.
   b. Select **Service Integration > Service Integration Bus Browser**.
   c. Click the bus destination that DEF used.

   The queue depth must be greater than 0.

Installing BPMEventEmitterWAR on the IBM BPM server

After you have prepared the IBM® BPM environment, you install the BPMEventEmitterWAR package and start the application.

Procedure

1. Log in to the WebSphere® Application Server administration console.

   The default address is `http://<Server_Address>:9060/admin`.

2. In the navigator panel, expand **Applications > Application Types > WebSphere enterprise** applications, and then click **Install**.
3. Select the `BPMEventEmitter.war` file from the local disk or specify the remote address for that file, and click **Next**.
4. Select **Fast Path** and click **Next**.
   a. Step 1: Select installation options, if you do not have any special access control on your server, click **Next**.
   b. Step 2: Map modules to servers, for a single cluster environment, keep the default value. For a golden topology with three clusters, you can map the application to the support cluster only.
   c. Step 3: Map context roots for Web modules, click **Next**.
   d. Step 4: Metadata for modules, click **Next**.
   e. Step 5: Summary, click **Finish**.
5. After the installation is successful, click the **Save** link that is displayed.

   The application is installed and is listed in a table row.

6. Proceed with the following post-installation actions.
   a. Click the application name to open the configuration page.
   b. In the Modules section, click **Manage Modules**.
   c. Click **BPMEventEmitter** to open the module configuration page.
7. In the configuration page, from the **Class loader order** menu, select **Parent last**.
8. Click **OK** in the next page, and then click **Save** to save the change to the main WebSphere Application Server configuration.
9. Optional: If you do not use the default application specification and queue JNDI and you have not updated those values before installing the BPMEventEmitter application, do it now in the **Enterprise Applications > BPMEventEmitter_war > Message Driven Bean listener bindings** panel.
10. Optional: Edit the `was.policy` file.

    For more information, see Security for IBM BPM Analytics.

11. After logging in, expand **Applications > Application Types > WebSphere enterprise applications** in the navigator panel, then select **BPMEventEmitter_war** and click **Start**.

Results

After a successful start, you can see status messages that are similar to the following example in SystemOut.log.

```
[5/18/17 12:15:16:466 IST] 00000152 LifeCycleMana I Application Name:IBM BPM
Analytics
[5/18/17 12:15:16:466 IST] 00000152 LifeCycleMana I Application
Version:8.5.7.201703
[5/18/17 12:15:16:467 IST] 00000152 LifeCycleMana I Build Level:20170518_28
[5/18/17 12:15:49:509 IST] 0000b44b LifeCycleMana I   I:  DEF2ES MDB started.
[5/18/17 12:15:49:509 IST] 0000b44b ConfigConnect I   I: Kafka connection
disabled.
[5/18/17 12:15:49:727 IST] 0000b44b ConfigConnect I   I: ElasticSearch
connection created.
```

The application will retrieve messages from the Dynamic Event Framework (DEF) monitor event queue, transform them into JSON objects, and write to the Elasticsearch server.

If the application fails to start, see Troubleshooting IBM BPM Analytics.

Optional: Disabling the Dynamic Event Framework (DEF) and skipping KPI data

After you are finished with IBM® BPM Analytics, you can disable the Dynamic Event Framework (DEF) to avoid performance loss. You can also omit KPI data.

About this task

You can improve performance by disabling DEF. But analysis also shows that part of performance decrease is due to KPI data retrieval.

Procedure

1. To delete the generated DEF configuration, open the `install_root/profiles/<Dmgr_Profile>/config/cells/<cell_name>/defconfig.xml` file and delete the `<defListener></defListener>` and `</defProducer></defProducer>` configurations.
2. Run the `SampleReloadDEF.py` script to reload DEF.

   From a command line, go to the `bin` directory under the home directory of your deployment manager profile and run the following command:

   ```
   Wsadmin -lang jython -f <Script_Location>/SampleReloadDEF.py
   ```

3. From the WebSphere® Application Server administration console, select **System administration > Nodes**, select the nodes that you want to synchronize, and click **Full Resynchronize**.
4. Optional: If you do not need the KPI data to be added to DEF generated messages, add the following property to the `100Custom.xml` configuration file.

   ```
   <common>
     <skip-kpi-data merge="replace">true</skip-kpi-data>
   </common>
   ```

   The location of the `100Custom.xml` file depends on the type of the server that you are configuring: performance-data-warehouse, process-center, or process-server.

5. Synchronize the node and restart the IBM BPM server for the changes to take effect.

Optional: Disabling task event generation

By default, the Dynamic Event Framework (DEF) generates task event notification. To improve performance, you can choose to disable this feature by adding a configuration property.

About this task

If you use the REST API or the REST API test tool or the JavaScript API to modify human tasks in a business process definition (BPD), TASK_FIELD_CHANGED or TASK_SUBJECT_CHANGED messages are generated in the monitored queue.

```
<mon:monitorEvent mon:id="fd44650ea9c51792162189">
  <mon:eventPointData>
    <mon:kind mon:version="2010-11-11">wle:TASK_FIELD_CHANGED</mon:kind>
```

or

```
<mon:monitorEvent mon:id="x17ce6a4cfa9c51792162189">
  <mon:eventPointData>
    <mon:kind mon:version="2010-11-11">wle:TASK_SUBJECT_CHANGED</mon:kind>
```

These notification messages are likely to reduce performance. To keep up performance, you can disable notification by adding a server configuration property. You can edit server configuration properties in the `100Custom.xml` file to override the default values. The location of the `100Custom.xml` file depends on the type of the server that you are configuring: performance-data-warehouse, process-center, or process-server. For example, for Process Center, the path is `install_root/profiles/Dmgr01/config/cells/testServerCell01/nodes/testServerNode01/servers/BPM.AppCluster.testServerNode01.0/process-center/config/100Custom.xml`

The applicable rules are described in The 100Custom.xml file and configuration.

For more information about APIs, see the following pages:

- REST APIs
- Test APIs
- JavaScript APIs

Procedure

1. Open the appropriate `100custom.xml` file and add the following configuration property.

```
<common>
   <monitor-event-emission>
       <enable-task-api-def merge="replace">false</enable-task-api-def>
   </monitor-event-emission>
</common>
```

2. Synchronize the node and restart the IBM® BPM server for the changes to take effect.

Generate XML DEF events (not recommended)

Although this is not the preferred practice, you can generate DEF events in XML format.

About this task

IBM® BPM provides scripts that you can use to configure the DEF to generate XML events. You can find these scripts in the *install_root*/BPM/Lombardi/tools/def directory.

Remember: Using JSON native DEF events improves overall performance. See Generating JSON native DEF events.

- Use the SampleConfigureEventsToJMS.py script to define the queue connection factory, event subscriptions, and authentication alias.
- Use the SampleReloadDEF.py script to make the DEF refresh its configuration dynamically.

If you want to configure the DEF to receive XML DEF events, you must create WebSphere® Application Server resources manually before you run each script.

- Create a JMS queue connection factory
  You start with creating a Java Message Service (JMS) connection factory.
- Defining a destination on the bus
  After you have created a queue connection factory, you define a destination on the IBM BPM deployment environment bus.
- Creating a JMS queue
  After you have defined a destination on the IBM BPM deployment environment, you create a Java Message Service (JMS) queue.
- Editing and running the configuration script
  After you have created a queue connection factory, defined a destination bus, and created a queue, you update the configuration script with this information, run this script, reload the Dynamic Event Framework (DEF), and check that it was successfully enabled.

Create a JMS queue connection factory

You start with creating a Java Message Service (JMS) connection factory.

Procedure

1. In the WebSphere® Application Server administration console, click **Resources > JMS > Queue connection factories**.
2. Set the scope to cluster, such as `Cluster=<Cluster_Name>`.

   The queue connection factory is used by the Dynamic Event Framework (DEF) emitter that belongs to the IBM® BPM engine. Therefore, the scope of the queue connection factory is either the application cluster or the cell.

3. Click **New** to create a JMS queue connection factory.
4. Select a JMS resource provider, such as **Default messaging provider** and click **OK**.
5. On the next configuration page, enter a name for the connection factory, such as `monitorcf`, and a JNDI name, such as `jms/monitorcf`, and under **Connection > Bus name**, select a bus for the DEF, such as **BP.BPM.Bus**.
6. Under **Security settings > Container-managed authentication alias**, select an authentication alias, such as **DeAdminAlias**.

What to do next

Next, you define a destination on the IBM BPM deployment environment bus.

Defining a destination on the bus

After you have created a queue connection factory, you define a destination on the IBM® BPM deployment environment bus.

Procedure

1. In the WebSphere® Application Server administration console, click **Service integration > Buses** and select the bus.
2. Click **Destinations** and create a queue destination.
3. Select **Queue** as the destination type.
4. Define a queue identifier, such as `MonitorDES`.
5. Select the bus member that you defined in step 2 of Create a JMS queue connection factory.

What to do next

Next, you create a JMS queue.

Creating a JMS queue

After you have defined a destination on the IBM® BPM deployment environment, you create a Java Message Service (JMS) queue.

Procedure

1. In the WebSphere® Application Server administration console, click **Resources > JMS > Queues** and set the scope to **Cell**, such as `Cell=<Cell_Name>`.

   You must create the queue at the cell scope because the queue is shared by the Dynamic Event Framework (DEF) emitter as the write destination and the BPMEventEmitter application as the read source. For a three-cluster topology, to avoid resource conflicts, install the BPMEventEmitter application on the support cluster. By creating the queue at the cell scope, you ensure that the queue is accessible by both the DEF emitter and BPMEventEmitter.

2. Click **New** to create a JMS queue, and select **Default messaging provider** as the JMS resource provider.
3. On the configuration page, enter a name, such as `monQueue` and a JNDI name, such as `jms/monQueue`.
4. Under **General properties > Connection**, select the same bus name as in step 5 of Create a JMS queue connection factory and select **monitorDES** as the queue name.

What to do next

Now, you update the `SampleConfigureEventsToJMS.py` script.

Editing and running the configuration script

After you have created a queue connection factory, defined a destination bus, and created a queue, you update the configuration script with this information, run this script, reload the Dynamic Event Framework (DEF), and check that it was successfully enabled.

About this task

You can find the `SampleConfigureEventsToJMS.py` script file in the `install_root/BPM/Lombardi/tools/def` directory.

Procedure

1. Edit the fields to reflect the settings that you defined for the connection factory, destination bus, and queue.

   `defListenterId`
   Enter a character string that uniquely identifies this event listener.

   `eventQueueJndiName`

   Enter as a character string the JNDI name of the queue resource that you created in the WebSphere® Application Server administration console.

   `eventQueueCFJndiName`

   Enter as a character string the JNDI name of the queue connection factory resource that you created in the WebSphere Application Server administration console.

   `eventQueueCF_AuthorizationAlias`

   Enter as a character string the authorization alias that you created in the WebSphere Application Server administration console.

   This example illustrates these settings.

   ```
   defListenerId = ' jmsListenerTHREE'
   eventQueueJndiName = 'jms/monQueue'
   eventQueueCFJndiName = 'jms/monitorcf'
   eventQueueCF_AuthorizationAlias = 'DeAdminAlias'
   ```

2. Edit the subscription section as instructed in step 4 of Generating JSON native DEF events.
3. Make sure that the support cluster where the Dynamic Event Framework (DEF) runs is started.

4. Go to the `bin` directory under your deployment manager profile home directory and run the `SampleConfigureEventsToJMS.py` script on a command line.

   ```
   wsadmin -lang jython -f <Script_Location>/SampleConfigureEventsToJMS.py
   ```

5. From the same bin directory, run the `SampleReloadDEF.py` script to reload the DEF.

   ```
   wsadmin -lang jython -f <Script_Location>/SampleReloadDEF.py
   ```

   The command creates the `defconfig.xml` file in the `dmgr_profile_home/config/cells/cellName` directory.

6. From the WebSphere Application Server administration console, select **System administration > Nodes**, select the node, and click **Full Synchronize**.
7. Restart the deployment environment.
8. Check that Dynamic Event Framework (DEF) was enabled successfully.
   a. Start a process in Process Portal.
   b. Select **Service Integration > Service Integration Bus Browser**.
   c. Click the bus destination that DEF used.

   The queue depth must be greater than 0.