Monitoring Processes and Services in IBM Business Process Manager Versions 7.5.1.2 & 8.0.1.2

Contents

Monitoring Processes and Services in IBM Business Process Manager Versions 7.5.1.2 & 8.0.1.2

Capturing performance data

The Process Admin Console includes an Instrumentation monitor to help identify performance bottlenecks in Process Server and to capture instrumentation data that you can use to further analyze any performance issues.

Before you begin

You must log in to the Process Admin Console.

Procedure

- To access the Instrumentation monitor and display the most recent data:
  1. In the Server Admin area of the Process Admin Console, click the indicator next to **Monitoring** to list the available monitoring options.
  2. Click the **Instrumentation** option.
  3. Click the **Refresh** button.
  4. To automatically refresh the displayed data, select the time interval that you want from the drop-down menu.
  5. To reset all values to 0, click the **Reset** button. This enables you to monitor performance as data is collected.
- To log instrumentation data to an external .dat file:
  1. In the Server Admin area of the Process Admin Console, click the indicator next to **Monitoring** to list the available monitoring options.
  2. Click the **Instrumentation** option.
  3. Click the **Start Logging** button. The Instrumentation monitor displays the path and file to which the data is saved. The file is created and stored on the host of the IBM® Business Process Manager server that you are currently monitoring.
  4. Click the **Stop Logging** button to end data capture to the log file.

  CAUTION:
  Logging instrumentation data is an expensive operation that impacts both the processor and memory consumption of the system, and the logging file is large. Limit the use of logging to investigate performance issues with the server only.

Monitoring processes and services in the Process Admin Console

To identify performance issues with your process application, view the performance data available in the Process Monitor page of the Process Admin Console. Identify process applications that have bottlenecks, drill into the process application to identify the steps that are expensive, and learn how long it takes to run services.

Before you begin

In a network deployment environment, the Process Monitor is server-specific. The monitor data is only kept in memory, and it is specific to the Java virtual machine (JVM) process. To ensure that you are looking at Process Monitor for the correct server, connect directly to the IBM BPM server http or https port, instead of connecting to an http server that might route you to any one of the underlying IBM BPM servers.

About this task

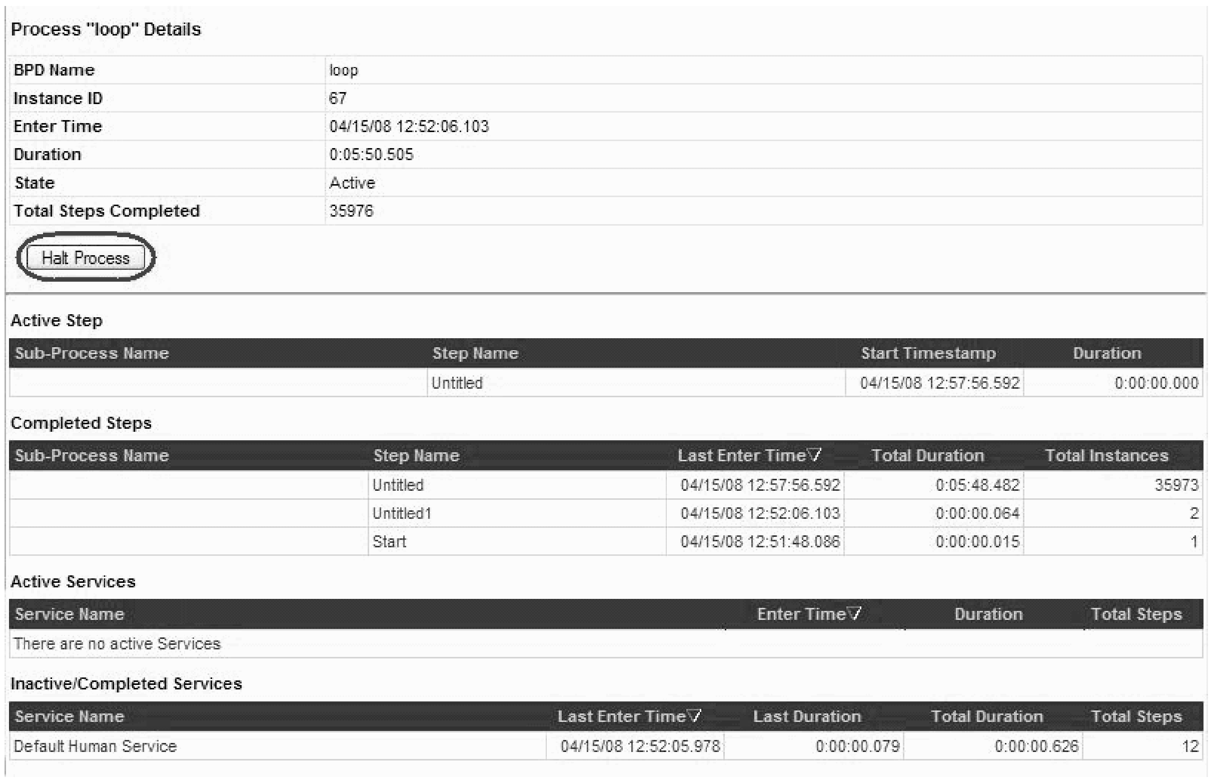
Limitations

- All monitor and instrumentations data are kept in memory and only show information about a particular cluster member. Restarting the server clears the data. To view information across different cluster members, connect to the deployment manager through a JMX console. For more information, see "Monitoring MBeans with JConsole" on page 6
- The haltProcess() and haltService() methods might not always be able to stop a process instance or service. The instance or service stops only if it is currently being run by the process and service engine, and is not stuck inside a service implementation (for example, in the middle of calling a web Service or running JavaScript).

Procedure

To view the performance information for your process apps and services:

- Log in to the Process Admin console, by entering the url. For example: **http://servername:9080/ProcessAdmin**, where *servername* is the name of your server and 9080 is the default port.
- In the Server Admin area of the Process Admin console, expand **Monitoring** to list the available monitoring options.
- Click the **Process Monitor** option.
- Switch to the Summary page, which provides an overview of active and most expensive processes and services.
  1. To view details of a particular process app, click the process app. The Processes page opens showing the details of the process app. You can view the duration of each step in the process, including the type of activity, such as event, gateway. You can also view the list of services that are running and the total duration of each service. You can identify a service to investigate, for example you might look at the service that is taking the longest time.
  2. To view details of all active and completed process apps, click **Processes**.
- Switch to the Services page and see a list of all the service steps and their activity types. Here you can identify the step, for example a coach, that is taking a long time. You can now try to determine why that particular step has a long duration. For example, you might notice that a particular coach is taking a long time to complete, and when you re run the process, the time might be significantly less, indicating that the performance issue is most likely due to the initial loading of the model. On further analysis, you might notice that there are numerous calls to stand-alone Ajax services, which might affect the scalability and performance of the coaches, and rework the process app so that the number of such calls are minimized.
- To stop an active process:
  1. Click **Processes**.
  2. Under **Active Processes Currently Executing**, click the name of the process that you want to stop.
  3. Click **Halt Process**.



Process "loop" Details

BPD Name	loop
Instance ID	67
Enter Time	04/15/08 12:52:06.103
Duration	0:05:50.505
State	Active
Total Steps Completed	35976

Active Step

Sub-Process Name	Step Name	Start Timestamp	Duration
	Untitled	04/15/08 12:57:56.592	0:00:00.000

Completed Steps

Sub-Process Name	Step Name	Last Enter Time▽	Total Duration	Total Instances
	Untitled	04/15/08 12:57:56.592	0:05:48.482	35973
	Untitled1	04/15/08 12:52:06.103	0:00:00.064	2
	Start	04/15/08 12:51:48.086	0:00:00.015	1

Active Services

Service Name	Enter Time▽	Duration	Total Steps
There are no active Services			

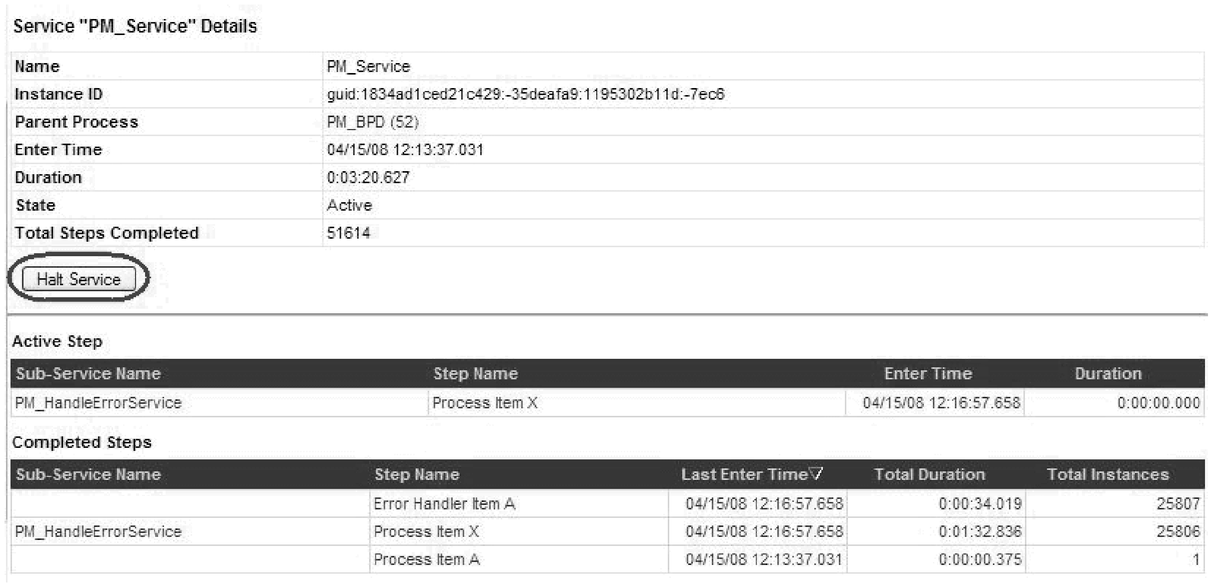
Inactive/Completed Services

Service Name	Last Enter Time▽	Last Duration	Total Duration	Total Steps
Default Human Service	04/15/08 12:52:05.978	0:00:00.079	0:00:00.626	12

Note: The **Halt Process** button appears only if the process is currently running.

The halted process now appears in the Active Processes Not Currently Executing list.

- To stop an active service:
  1. Click the **Services** option.
  2. Under **Active Services Currently Executing**, click the name of the service that you want to stop.
  3. Click **Halt Service**.



Service "PM_Service" Details

Name	PM_Service
Instance ID	guid:1834ad1ced21c429:-35deafa9:1195302b11d:-7ec6
Parent Process	PM_BPD (52)
Enter Time	04/15/08 12:13:37.031
Duration	0:03:20.627
State	Active
Total Steps Completed	51614

Active Step

Sub-Service Name	Step Name	Enter Time	Duration
PM_HandleErrorService	Process Item X	04/15/08 12:16:57.658	0:00:00.000

Completed Steps

Sub-Service Name	Step Name	Last Enter Time▽	Total Duration	Total Instances
	Error Handler Item A	04/15/08 12:16:57.658	0:00:34.019	25807
PM_HandleErrorService	Process Item X	04/15/08 12:16:57.658	0:01:32.836	25806
	Process Item A	04/15/08 12:13:37.031	0:00:00.375	1

Note: The **Halt Service** button appears only if the service is currently running.

The halted service now appears in the Active Services Not Currently Executing/Completed Services list.

Identify infinite loops in process applications

An infinite loop in a process app or service may cause the application to consume a large amount of processor resources. You can use the monitor page in the Process Admin console to look at the number of times certain steps and services are run in a process app. If you suspect an infinite loop, you can then open the corresponding BPD or service diagram to view the logic.

The following are some examples of infinite loops in a process application:

- An infinite loop between activities in a BPD. An infinite loop can occur in a BPD. For example, if there is conditional logic that is implemented by a gateway where the process flow loops back to a system task based on the value of a variable, and the exit condition never occurs, the loop might not exit. After you identify the process that is running for a long time, you can view the process steps, and if you see a step that has a large number of total steps, you might have an infinite loop within a BPD. You can identify the process instance in the Process Inspector by using the process id, and terminate the process.
- An infinite loop between service steps in an integration service. If a process is running with a large number of steps within a particular service, you might have an infinite loop within the service. Go to the Services page and use the **Halt Service** button to interrupt the service.
- An infinite loop within a server script step. For example, a JavaScript block that is in a continuous loop, which is usually indicated by a single JavaScript activity that is taking a lot of time.

  Note: The **Halt Process** button cannot interrupt a currently running JavaScript.
- An infinite loop between and event producer and an event consumer. For example, a process that loops between message send and message receive events. This situation might be indicated by many instances of a process being created in a very short duration.

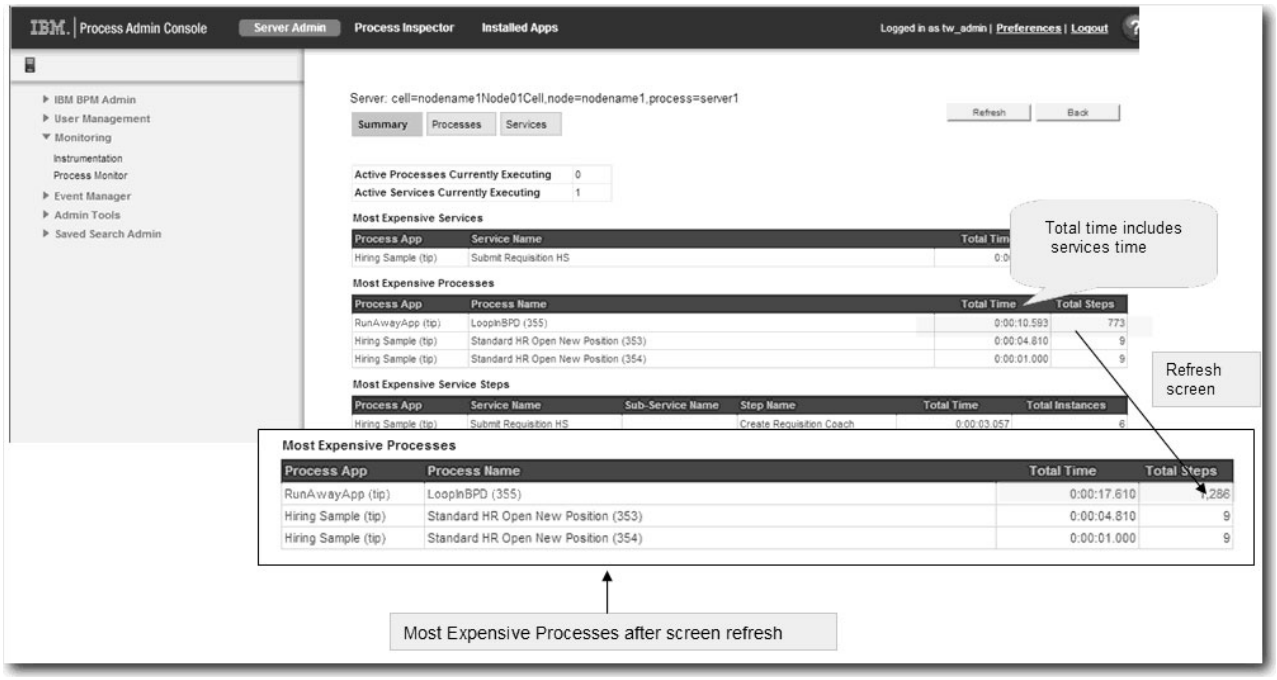
Example: Infinite loop in a BPD

An infinite loop can occur in a BPD. For example, if there is conditional logic that is implemented by a gateway where the process flow loops back to a system task based on the value of a variable, and the exit condition never occurs, the loop might not exit.

To detect the cause of the infinite loop, log in to the Process Admin console and open the Process Monitor. You can use the information that is displayed here to identify the BPD step that is looping:

- From the **Most Expensive Processes** list in the **Summary** page, you can identify the process app that is taking the longest time. The list shows the total time for each process, which includes the time that is taken by the services within the process.
- You can drill into a process to view the time that is taken by each step in the process (total duration), and the number of times a step has been executed (total instances).

The following image shows an example of a summary page, which shows that a process has been running for over 10 seconds. The system administrator refreshes the screen, and sees that the process has now been running for 17 seconds.



Server: cell=nodename1Node01Cell,node=nodename1,process=server1

Active Processes Currently Executing	0
Active Services Currently Executing	1

Most Expensive Services

Process App	Service Name	Total Tim
Hiring Sample (tip)	Submit Requisition HS	0:0

Most Expensive Processes

Process App	Process Name	Total Time	Total Steps
RunAwayApp (tip)	LoopInBPD (355)	0:00:10.593	773
Hiring Sample (tip)	Standard HR Open New Position (353)	0:00:04.810	9
Hiring Sample (tip)	Standard HR Open New Position (354)	0:00:01.000	9

Most Expensive Service Steps

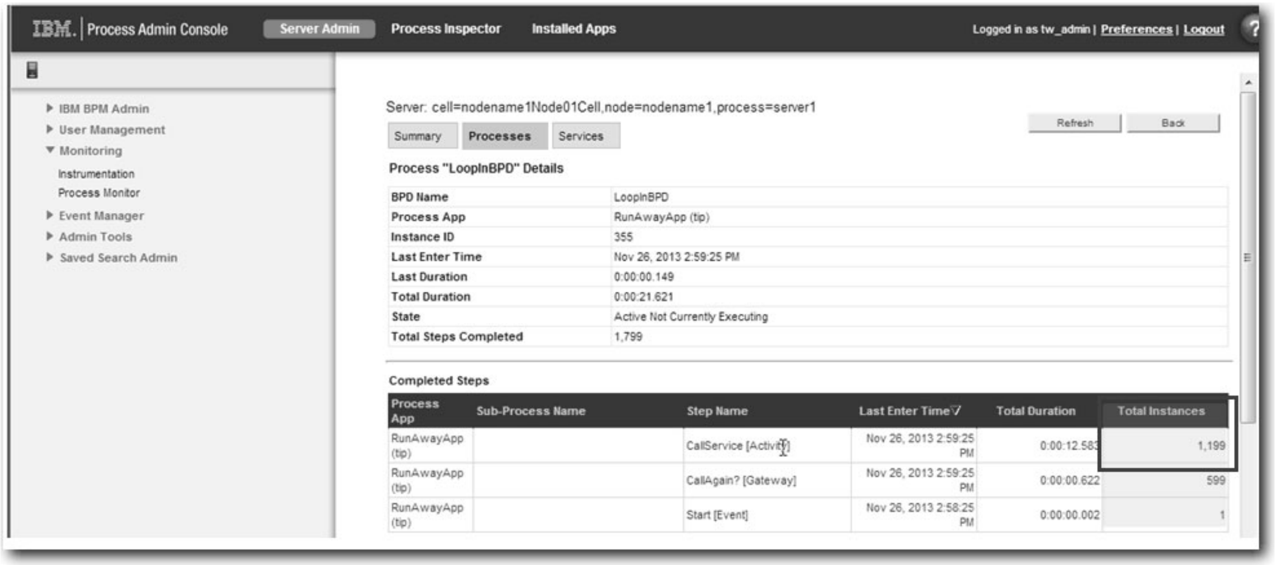
Process App	Service Name	Sub-Service Name	Step Name	Total Time	Total Instances
Hiring Sample (tip)	Submit Requisition HS		Create Requisition Coach	0:00:03.057	6

Most Expensive Processes

Process App	Process Name	Total Time	Total Steps
RunAwayApp (tip)	LoopInBPD (355)	0:00:17.610	1,286
Hiring Sample (tip)	Standard HR Open New Position (353)	0:00:04.810	9
Hiring Sample (tip)	Standard HR Open New Position (354)	0:00:01.000	9

The system administrator drills into the process details, and sees that a service has run over 1000 times, which is usually an indication that the process is looping.

Note: By default, the system keeps track of the last 100 completed processes and the last 500 completed services. To change these settings, edit `00static.xml`.



Server: cell=nodename1Node01Cell,node=nodename1,process=server1

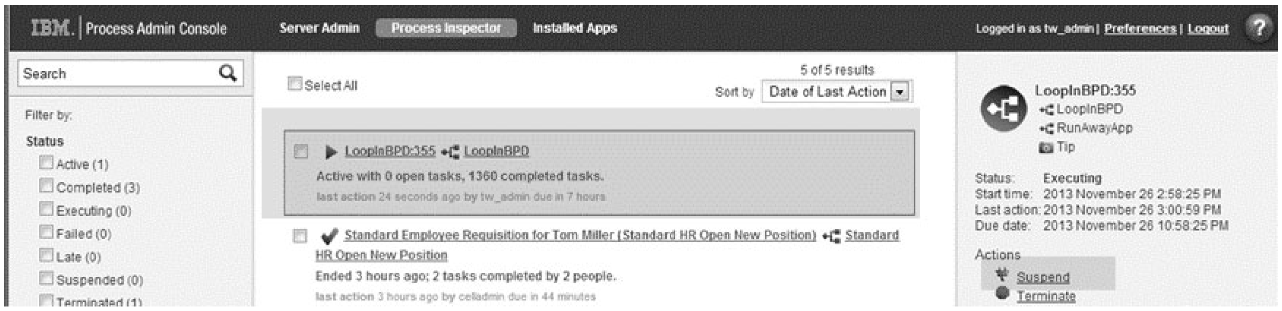
Process "LoopInBPD" Details

BPD Name	LoopInBPD
Process App	RunAwayApp (tip)
Instance ID	355
Last Enter Time	Nov 26, 2013 2:59:25 PM
Last Duration	0:00:00.149
Total Duration	0:00:21.621
State	Active Not Currently Executing
Total Steps Completed	1,799

Completed Steps

Process App	Sub-Process Name	Step Name	Last Enter Time▽	Total Duration	Total Instances
RunAwayApp (tip)		CallService [Activity]	Nov 26, 2013 2:59:25 PM	0:00:12.583	1,199
RunAwayApp (tip)		CallAgain? [Gateway]	Nov 26, 2013 2:59:25 PM	0:00:00.622	599
RunAwayApp (tip)		Start [Event]	Nov 26, 2013 2:58:25 PM	0:00:00.002	1

The system administrator switches to the **Process Inspector** and uses the instance ID (355) to find the looping process, and terminates the process.



5 of 5 results

LoopInBPD:355 LoopInBPD
Active with 0 open tasks, 1360 completed tasks.
last action 24 seconds ago by tw_admin due in 7 hours

Standard Employee Requisition for Tom Miller (Standard HR Open New Position) Standard HR Open New Position
Ended 3 hours ago; 2 tasks completed by 2 people.
last action 3 hours ago by celladmin due in 44 minutes

LoopInBPD:355
LoopInBPD
RunAwayApp
Tip

Status: Executing
Start time: 2013 November 26 2:58:25 PM
Last action: 2013 November 26 3:00:59 PM
Due date: 2013 November 26 10:58:25 PM

Tracking changes in the number of instances created and service requests

You can use the Instrumentation page in the Process Monitor to log and track the number of instances that are created and the number of service requests. You can then investigate changes in the data to determine whether there is a problem. For example, if there is a spike in the number of new instances that are created, you can use the data to understand whether the spike is a result of a normal business situation, an application error or a denial of service attack.

To view the number of process instances that were created for each BPD since the server started:

1. Log in to the Process Admin console, by entering the url. For example: **http://servername:9080/ProcessAdmin**, where *servername* is the name of your server and 9080 is the default port.
2. In the Server Admin area of the Process Admin console, expand **Monitoring** to list the available monitoring options.
3. Click **Instrumentation**. You can see how many process instances were created for each BPD since the server started. You can also see how many service requests were made to each process app, for services such as integration service, human service, Ajax service.
4. Click **Save** to save the instrumentation data as an XML file.

You can also perform these tasks by using the Process Monitor:

- Retrieve instrumentation data as XML by using the **InstrumentationManager** MBean
- See the rate of change by monitoring the **BPDInstancesStarted** MBean

For more information, see:

- "Monitoring MBeans with JConsole"
- "Process Monitor MBeans reference" on page 16

Monitoring MBeans with JConsole

The Process Monitor data in the Process Admin console shows you only data for the cluster member to which it is connected. You can view information for different cluster members by using a JMX-compliant tool such as JConsole. Each cluster member hosts an instance of the MBean, which can be queried individually to get the full picture.

You can run a script to start JConsole and then view the JMX MBeans that are available for the Process Monitor, and navigate to the MBean for each cluster member to view the data. To view the MBeans for all cluster members, connect JConsole to the Deployment Manager. To view Process Monitor MBeans from JConsole:

1. Start JConsole by using a script.
   - Example Linux script (*filename.sh*)

     ```
     #!/bin/bash
     ################################################################
     # This script is assumed to run on the same host where the BPM
     # server is running.
     ```

```
# Change the variable( $HOST, $PROFILE_NAME, $WAS_VER, $PORT...)
# if your environment is different
#
# To get $WAS_VER, go to $WAS_HOME/runtimes directory.
# The version number is the 3 digit number that is part of the
# jar file name. Eg com.ibm.jaxrs.thinclient_8.0.0.jar
#################################################################

PROFILE_NAME=StandAloneProfile
WAS_VER=$WAS_VER
PORT=2809


HOST=$(hostname)
WAS_HOME=$HOME/main/deploy2/AppServer
JAVA_HOME=$WAS_HOME/java

PROFILE_HOME=$WAS_HOME/profiles/$PROFILE_NAME
BPM_HOME=$WAS_HOME
PROVIDER=-Djava.naming.provider.url=corbaname:iiop:$HOST:$PORT

PROPs_HOME=$PROFILE_HOME/properties

FILE_SAS=$PROPs_HOME/sas.client.props
FILE_SSL=$PROPs_HOME/ssl.client.props

CLIENTSAS=-Dcom.ibm.CORBA.ConfigURL=file://$FILE_SAS
CLIENTSSL=-Dcom.ibm.SSL.ConfigURL=file://$FILE_SSL

echo "sas file: $FILE_SAS"
echo "ssl file: $FILE_SSL"

PROPs=
PROPs="$PROPs $CLIENTSAS"
PROPs="$PROPs $CLIENTSSL"
PROPs="$PROPs $PROVIDER"
echo $PROPs

CLASSPATH=
CLASSPATH=$CLASSPATH:$JAVA_HOME/lib/tools.jar
CLASSPATH=$CLASSPATH:$BPM_HOME/runtimes/com.ibm.ws.admin.client_$WAS_VER.jar
CLASSPATH=$CLASSPATH:$BPM_HOME/runtimes/com.ibm.ws.ejb.thinclient_$WAS_VER.jar
CLASSPATH=$CLASSPATH:$BPM_HOME/runtimes/com.ibm.ws.orb_$WAS_VER.jar
CLASSPATH=$CLASSPATH:$BPM_HOME/plugins/javax.j2ee.management.jar
CLASSPATH=$CLASSPATH:$JAVA_HOME/lib/jconsole.jar

URL=service:jmx:iiop://$HOST:$PORT/jndi/JMXConnector

JARS=$(echo $CLASSPATH |sed s/:/\ /g)

FILE_SETUPCMD=$PROFILE_HOME/bin/setupCmdLine.sh

#detect the require file:
for file in $FILE_SAS $FILE_SSL $JARS
do
    if [ ! -e $file ]
 then
 echo "file doesn't exist: $file"
 exit
    fi
done

#exit
. $FILE_SETUPCMD

$JAVA_HOME/bin/java -classpath $CLASSPATH $PROPs  sun.tools.jconsole.JConsole $URL
```

- Example windows script (*filename.bat*)

```
@echo off
REM ############################################################
REM # This script is assumed to run on the same host where the
REM # BPM server is running.
REM # Change the variable( HOST, PROFILE_NAME, WAS_VER, PORT...)
REM # if your environment is different
REM #
REM # To get %WAS_VER%, go to %BPM_HOME%\runtimes directory.
REM # The version number is the 3 digit number that is part of the
REM # jar file name. Eg com.ibm.jaxrs.thinclient_8.0.0.jar
REM ############################################################

set PROFILE_NAME=
set WAS_VER=%WAS_VER%
set PORT=2809


set HOST=

REM ************************************************************
REM PD_HOME is the Process Designer installation path, to reference
REM to the properties file for connection.
REM ************************************************************
set PD_HOME=
set JAVA_HOME=

REM ******************************************************************
REM BPM_HOME is a directory that stores the necessary jar files copied
REM from the server
REM ******************************************************************
set BPM_HOME=
set PROVIDER=-Djava.naming.provider.url=corbaname:iiop:%HOST%:%PORT%

set PROPs_HOME=%PD_HOME%/resources

set FILE_SAS=%PROPs_HOME%/sas.client.props
set FILE_SSL=%PROPs_HOME%/ssl.client.props

set CLIENTSAS=-Dcom.ibm.CORBA.ConfigURL=file:///%FILE_SAS%
set CLIENTSSL=-Dcom.ibm.SSL.ConfigURL=file:///%FILE_SSL%

echo "sas file: %FILE_SAS%"
echo "ssl file: %FILE_SSL%"

set PROPs=%CLIENTSAS%
set PROPs=%PROPs% %CLIENTSSL%
set PROPs=%PROPs% %PROVIDER%

echo "props: " %PROPs%

REM ********************************************************************************
REM the following jar files in the directory of BPM_HOME should be copied
REM from the server
REM ********************************************************************************
set CLASSPATH=%JAVA_HOME%\lib\jconsole.jar
set CLASSPATH=%CLASSPATH%;%JAVA_HOME%\lib\tools.jar
set CLASSPATH=%CLASSPATH%;%BPM_HOME%\com.ibm.ws.admin.client_%WAS_VER%.jar
set CLASSPATH=%CLASSPATH%;%BPM_HOME%\com.ibm.ws.ejb.thinclient_%WAS_VER%.jar
set CLASSPATH=%CLASSPATH%;%BPM_HOME%\com.ibm.ws.orb_%WAS_VER%.jar
set CLASSPATH=%CLASSPATH%;%BPM_HOME%\javax.j2ee.management.jar


set URL=service:jmx:iiop://%HOST%:%PORT%/jndi/JMXConnector


REM FILE_SETUPCMD=$PROFILE_HOME/bin/setupCmdLine.sh
REM $FILE_SETUPCMD
echo "class path: "  %CLASSPATH%


REM %JAVA_HOME%\bin\java -classpath %JAVA_HOME%/lib/jconsole.jar;
REM                 %java_home%/lib/tools.jar sun.tools.jconsole.JConsole %URL%
REM %JAVA_HOME%\bin\java -classpath %CLASSPATH% sun.tools.jconsole.JConsole %URL%
%JAVA_HOME%\bin\java -classpath %CLASSPATH% %PROPs% sun.tools.jconsole.JConsole %URL%
```

2. In the MBeans tab, locate ProcessMonitor and navigate to the server that you want.
3. You can see the MBeans operations that are available. Choose the operation that you want to run, enter parameter values, and click the operation to run it.

You can also use MBeans programmatically to extract information in the process instrumentation and monitor pages. For more information, see “Process Monitor MBeans reference” on page 16.

Data displayed in the Process Monitor

The Process Admin Console includes a Process Monitor that enables administrators to view information about the processes and services that are running on Process Server. You can use the information to identify any problematic processes or services.

The process monitor displays information only for the current cluster member. You can see this information at the top of the Process Monitor page. For example:

```
Server: cell=nodename1Node01Cell,node=nodename1,process=server1
```

To see data for all cluster members at the same time, you must use JMX. For more information, see “Monitoring MBeans with JConsole” on page 6

Summary

The summary shows you how many active services and processes are currently consuming processor resources in the current cluster. You can view the total time, total number of instances, and the total number of steps that are needed to run a service or process; and identify the services and processes that are the most expensive. For example, a service that takes a long time to run, or a process that has a large number of steps.

Data displayed	Description
Active Processes Currently Executing	Total number of process instances currently running on this server.
Active Services Currently Executing	Total number of services currently running on this server that are potentially problematic.
Most Expensive Services	Name, total running time, and the number of steps that are required for each service that is deemed most costly on this server.
Most Expensive Processes	Process name, which includes the instance ID, total running time (which includes service execution time), and the number of steps that are required for each process that is deemed most costly on this server.
Most Expensive Service Steps	Service name, step name, total running time, and total number of instances that are required to run each step that is deemed most costly on this server. (If any sub-services are associated with the step, the Process Monitor displays those sub-service names as well.)
Most Expensive Process Steps	Process name, step name, total running time, and total number of instances that are required to run each step that is deemed most costly on this server. (If any subprocesses are associated with the step, the Process Monitor displays those subprocess names as well.)

Processes

The Processes page shows the following data for all processes on the server, in the current cluster.

Data displayed	Description
Active Processes Currently Executing	Process name, which includes the instance ID, enter time (start time), duration (running time), and total number of steps for each process instance currently running on this server. In addition to processes that are running smoothly, this category also includes currently running processes that might have problems. For example, if a process instance is stuck in a repeating loop, it is shown in this list.
Active Processes Not Currently Executing	Name, last enter time (most recent start time), last duration (running time from most recent execution), total duration (cumulative running time), and total number of steps for processes that were previously started, but not currently active, on this server. This category includes process instances that are active but not running at this moment. For example, if a process instance is waiting for an event, it is included in this category.
Completed Processes	Name, last enter time (most recent start time), last duration (running time from most recent execution), total duration (cumulative running time), and total number of steps for processes that ran successfully on this server.

Services

The **Services** page shows the following data for all services on this server:

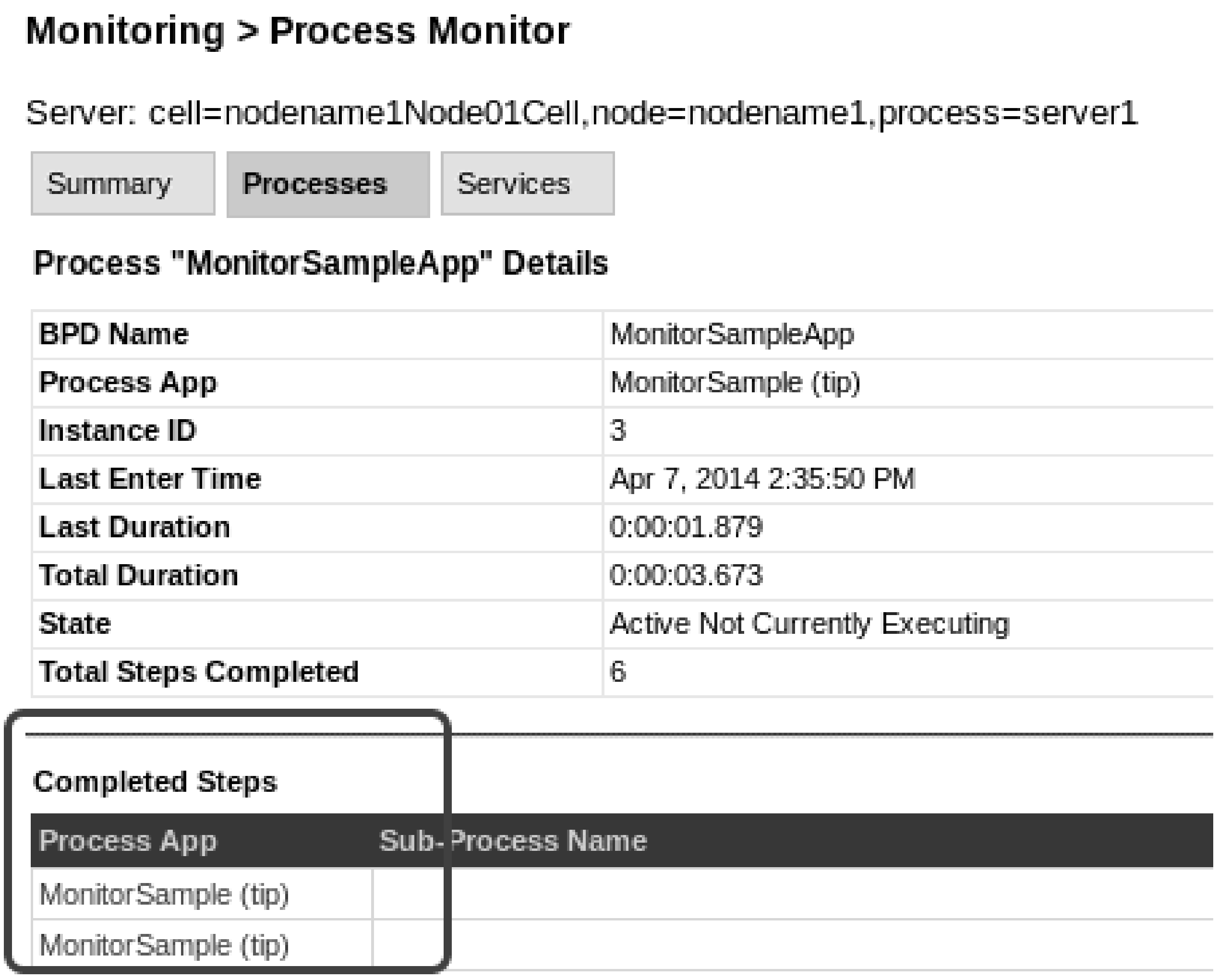
Data displayed	
Active Services Currently Executing	Name, enter time (start time), duration (running time), and total number of steps for each service currently running on this server. In addition to services that are running smoothly, this category also includes currently running services that might have problems. For example, if a service is stuck in a repeating loop, it is shown in this list.
Active Services Not Currently Executing/Completed Services	Name, last enter time (most recent start time), last duration (running time from most recent execution), total duration (cumulative running time), and total number of steps for services that were previously started, but not currently active, on this server and for services that ran successfully on this server. This category includes two types of services: • Services that completed successfully • Services that were previously started but are not currently running. For example, if a service is waiting for an event, it is included in this category.

Interpreting Process Monitor data

There are common assumptions people make when they look at Process Monitor data that might lead them to interpret their process performance incorrectly. Learn about the areas of Process Monitor where users are most likely to misinterpret what the data represents.

Completed Steps column

The completed steps column represents the current execution state of the process or service step that is received from the process engine. It means only that the process or service engine has completed execution of the step, but it does not mean the step itself is actually completed, as the step might be waiting for a response from a user or the system.



Monitoring > Process Monitor

Server: cell=nodename1Node01Cell,node=nodename1,process=server1

Process "MonitorSampleApp" Details

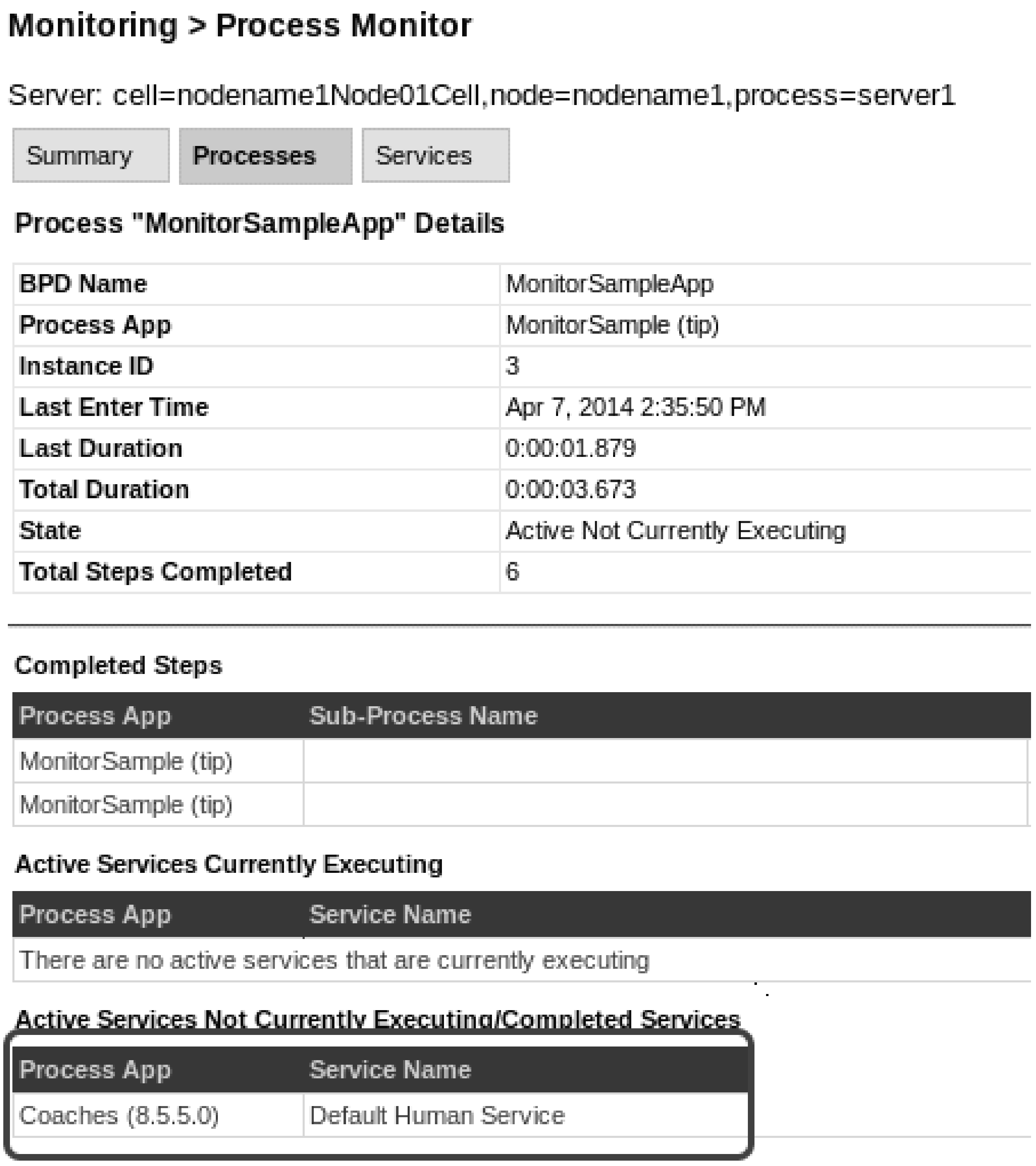
BPD Name	MonitorSampleApp
Process App	MonitorSample (tip)
Instance ID	3
Last Enter Time	Apr 7, 2014 2:35:50 PM
Last Duration	0:00:01.879
Total Duration	0:00:03.673
State	Active Not Currently Executing
Total Steps Completed	6

Completed Steps

Process App	Sub-Process Name
MonitorSample (tip)	
MonitorSample (tip)	

Process App column

The Process App column in the Active services table lists the process app or toolkit that contains the service at design time (in the Process Center); it is not the runtime process application. For example, in the following image, although the service Default Human Service is a task of the BPD instance at run time, it is shown as part of *Coaches(8.5.5.0)* because it is in the *Coaches(8.5.5.0)* toolkit.


Monitoring > Process Monitor

Server: cell=nodename1Node01Cell,node=nodename1,process=server1

Process "MonitorSampleApp" Details

BPD Name	MonitorSampleApp
Process App	MonitorSample (tip)
Instance ID	3
Last Enter Time	Apr 7, 2014 2:35:50 PM
Last Duration	0:00:01.879
Total Duration	0:00:03.673
State	Active Not Currently Executing
Total Steps Completed	6

Completed Steps

Process App	Sub-Process Name
MonitorSample (tip)	
MonitorSample (tip)	

Active Services Currently Executing

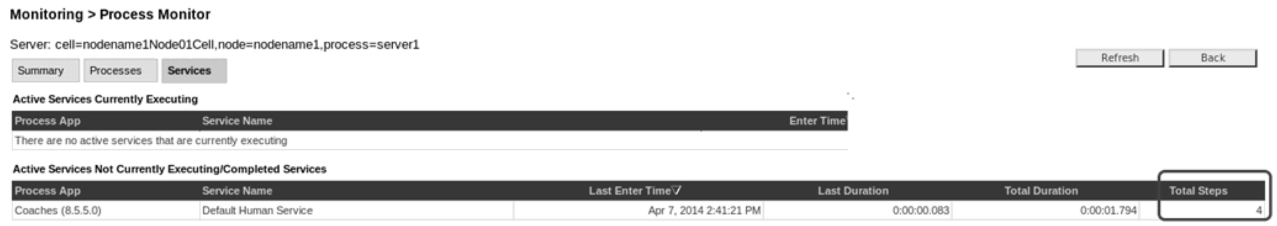
Process App	Service Name
There are no active services that are currently executing	

Active Services Not Currently Executing/Completed Services

Process App	Service Name
Coaches (8.5.5.0)	Default Human Service

Total Steps count

The total steps that are shown in the Process Monitor do not correspond to the total number of steps as defined in the business process definition. The total steps count includes the number of times the service is generated, even when the user does not actually run the service. For example, if a Process Portal user starts a task that is a human service, the corresponding coach page appears. If the user closes the coach page immediately without clicking anything else, the step total increases by one.



Server: cell=nodename1Node01Cell,node=nodename1,process=server1

Active Services Currently Executing

Process App	Service Name	Enter Time
There are no active services that are currently executing		

Active Services Not Currently Executing/Completed Services

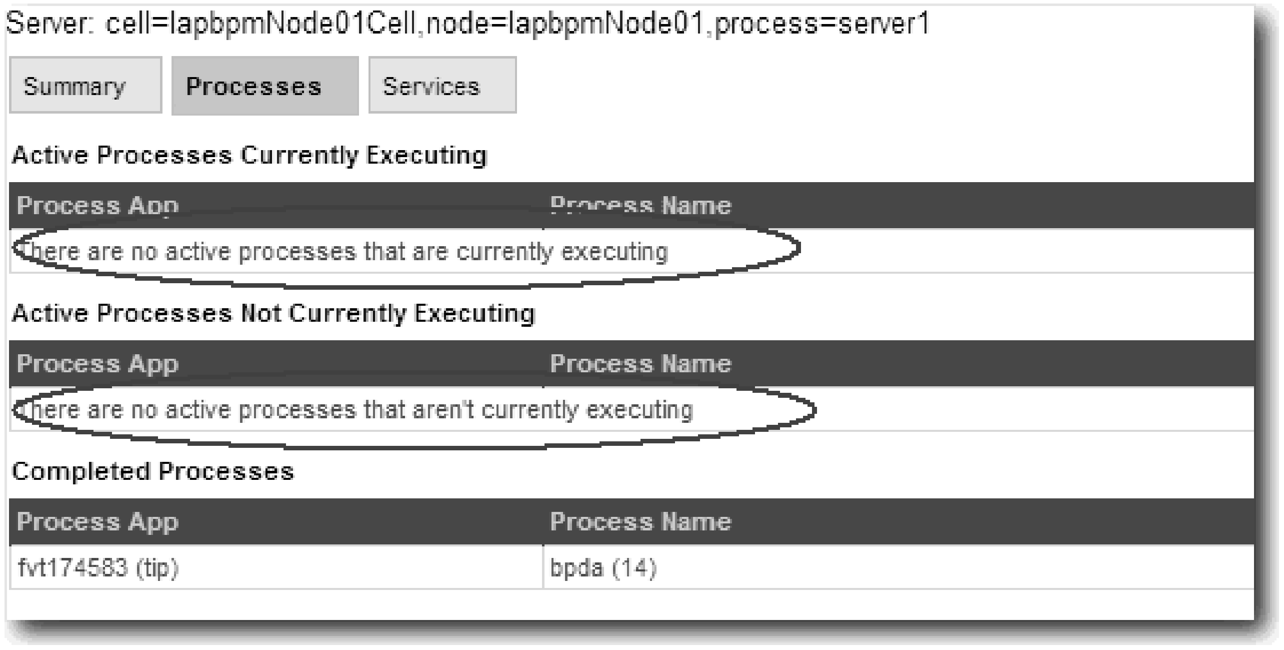
Process App	Service Name	Last Enter Time	Last Duration	Total Duration	Total Steps
Coaches (8.5.5.0)	Default Human Service	Apr 7, 2014 2:41:21 PM	0:00:00.083	0:00:01.794	4

Halt process and halt service buttons

It is typically difficult to see the halt process and halt service buttons, as they appear only when the service is actually being executed by the engine. When you see these buttons, the process or service is typically running in a loop, or the service is taking a long time to run.

Inconsistent monitoring data

In certain situations, you might see data that seems contradictory. For example, the total number of currently executing processes or services might be zero, but there are running processes or services.



Server: cell=lapbpmNode01Cell,node=lapbpmNode01,process=server1

Active Processes Currently Executing

Process App	Process Name
There are no active processes that are currently executing	

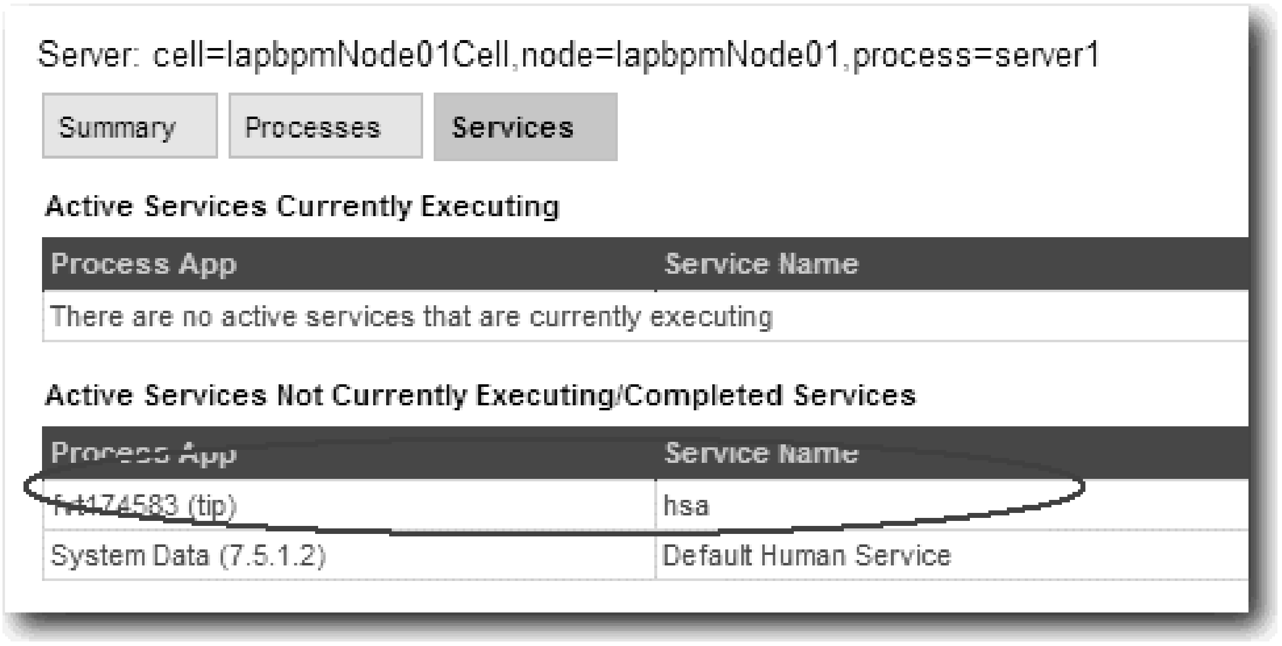
Active Processes Not Currently Executing

Process App	Process Name
There are no active processes that aren't currently executing	

Completed Processes

Process App	Process Name
fvt174583 (tip)	bpda (14)

Figure 1. Processes page


Server: cell=lapbpmNode01Cell,node=lapbpmNode01,process=server1

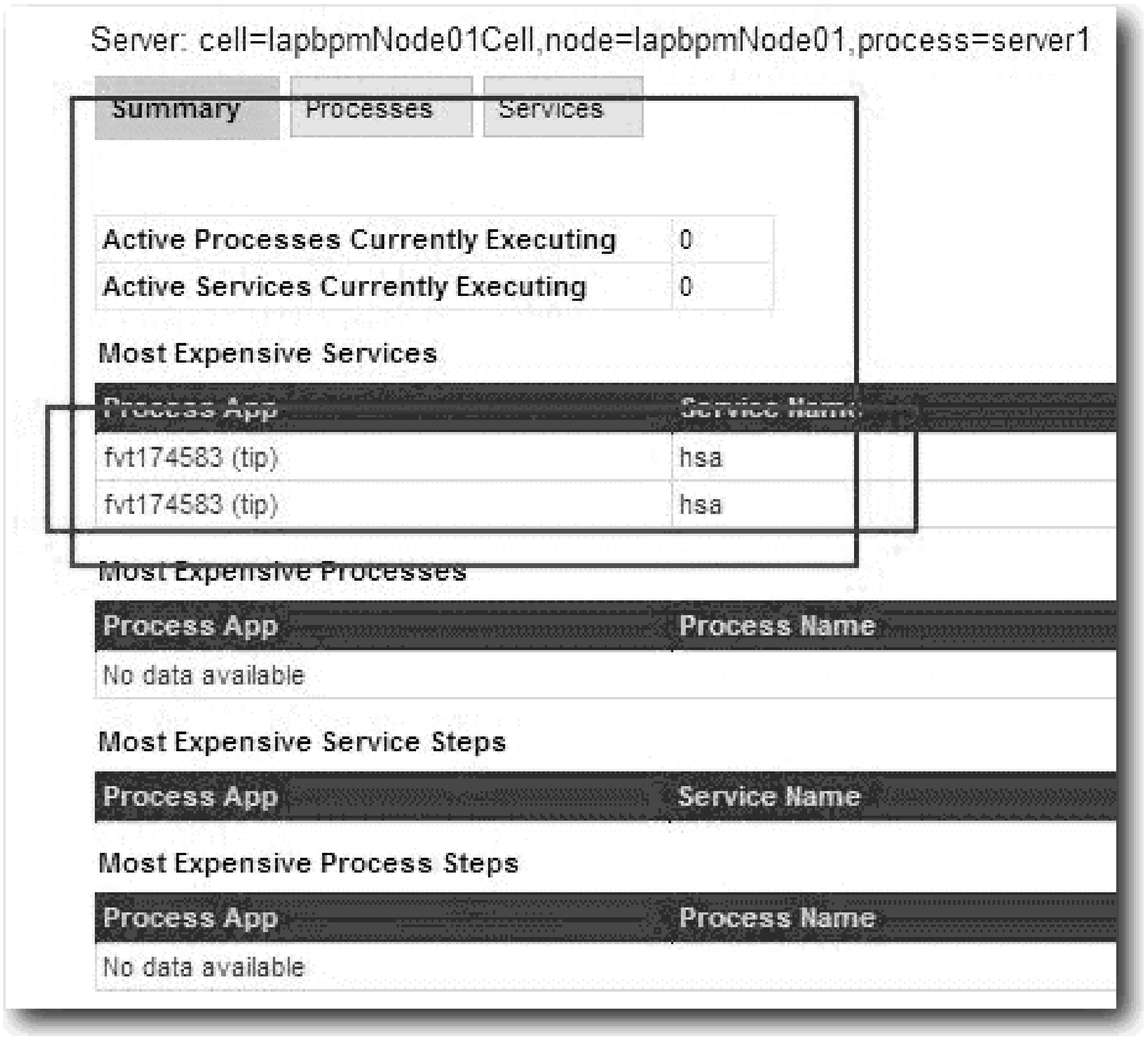
Active Services Currently Executing

Process App	Service Name
There are no active services that are currently executing	

Active Services Not Currently Executing/Completed Services

Process App	Service Name
hl174583 (tip)	hsa
System Data (7.5.1.2)	Default Human Service

Figure 2. Services page



Server: cell=lapbpmNode01Cell,node=lapbpmNode01,process=server1

Active Processes Currently Executing	0
Active Services Currently Executing	0

Most Expensive Services

Process App	Service Name
fvt174583 (tip)	hsa
fvt174583 (tip)	hsa

Most Expensive Processes

Process App	Process Name
No data available	

Most Expensive Service Steps

Process App	Service Name

Most Expensive Process Steps

Process App	Process Name
No data available	

Figure 3. Summary page

For example, the Processes page shows no active processes (Figure 1), but there are active processes in Process Portal, and the Services and Summary pages (Figures 2 and 3) show active process apps in the table.

These results might occur in the following scenario:
1. The system is not monitoring when the process instance is created.
2. The system starts monitoring.
3. The task in the process instance starts running.

A Process Monitor record is created when the event occurs. The process instance record is created when the instance is created. In the preceding scenario, when the process gets created, the corresponding record is not created, since the system is not monitoring. But the service event occurred after the system started monitoring. So there are service records but no process record. This situation might also occur when the server restarts.

Exit time calculation

If a process or service is still executing when it is exported, there is no exit time, and the system uses the current time as the exit time to calculate the duration. As you refresh the information on screen or through the JMX api, the duration time is updated. This explains why when a process or service that is executing is exported more than once, the duration time that shown in the exported file can vary.

Process Monitor MBeans reference

IBM Business Process Manager provides MBeans that you can use to access the Process Monitor data through a JMX compliant console such as JConsole.

InstrumentationManager MBean

IBM Business Process Manager provides a programming interface for accessing instrumentation data.

ObjectName (for Performance Data Warehouse):

```
com.lombardisoftware:Scope=ENVIRONMENT_PERFORMANCE_SERVER,Name=InstrumentationManager,cell=[cell],
                              Type=InstrumentationManager,node=[node],process=[server]
```

ObjectName (for Process Server):

```
com.lombardisoftware:Scope=ENVIRONMENT_SERVER,Name=InstrumentationManager,cell=[cell],
                     Type=InstrumentationManager,node=[node],process=[server]
```

MBean **InstrumentationManager**

The MBean interface to the instrumentation manager.

Data retrieved complies to the following schema:

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema attributeFormDefault="unqualified" elementFormDefault="qualified"
 targetNamespace="http://www.ibm.com/bpm/v1/instrumentations" xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:tns="http://www.ibm.com/bpm/v1/instrumentations">
 <xsd:element name="instrumentations" type="tns:instrumentationType">
 </xsd:element>

 <xsd:complexType name="instrumentationType">
  <xsd:sequence>
   <xsd:element name="logFilePath" type="xsd:string" maxOccurs="1" minOccurs="0" />
   <xsd:element name="inst" maxOccurs="unbounded" minOccurs="0" type="tns:instType"/>
  </xsd:sequence>
  <xsd:attribute type="xsd:boolean" name="isLogging" />
  <xsd:attribute type="xsd:string" name="server" />
  <xsd:attribute type="xsd:string" name="id" />

 </xsd:complexType>

 <xsd:complexType name="instType" mixed="true">
  <xsd:sequence>
   <xsd:element type="xsd:string" name="description" minOccurs="0"/>
   <xsd:element type="xsd:string" name="description-key" minOccurs="0" />
   <xsd:element type="xsd:long" name="value" minOccurs="0" />
   <xsd:element type="xsd:long" name="count" minOccurs="0" />
   <xsd:element type="xsd:int" name="inProcess" minOccurs="0" />
   <xsd:element type="xsd:decimal" name="averageDuration" minOccurs="0" />
   <xsd:element type="xsd:decimal" name="movingAverageDuration" minOccurs="0" />
   <xsd:element type="xsd:decimal" name="total" minOccurs="0" />
  </xsd:sequence>
  <xsd:attribute type="xsd:string" name="name" />
  <xsd:attribute type="xsd:string" name="id" />
  <xsd:attribute type="xsd:string" name="type" />
  <xsd:attribute type="xsd:int" name="depth" />
 </xsd:complexType>

</xsd:schema>
```

Table 1. Attribute summary

Attribute Summary	
java.lang.String	DisplayName
java.lang.String	LogFilePath Return the path to the current log file
boolean	Logging Returns true if the manager is currently logging.
javax.management.ObjectName	ParentObjectName Provides support for a parent/child structure over the beans.

Table 2. Operation summary

Operation Summary	
java.lang.String	retrieveInstrumentationAll() Return the set of instrumentation data as XML
java.lang.String	retrieveInstrumentationAllAsJSON () Return the set of instrumentation data as JSON
java.lang.String	retrieveInstrumentationByFilter(boolean showVisibleOnly) Return a selected set of instrumentation data as XML
java.lang.String	retrieveInstrumentationByFilterAsJSON (boolean showVisibleOnly) Return a selected set of instrumentation data as JSON
void	startLogging() Start logging to a new log file.
void	stopLogging() If currently logging stop, otherwise do nothing.

Attribute Detail

Logging
: **public boolean Logging**

  Returns true if the manager is currently logging.

LogFilePath
: **public java.lang.String LogFilePath**

  Return the path to the current log file

DisplayName
: **public java.lang.String DisplayName**

ParentObjectName
: **public javax.management.ObjectName ParentObjectName**

  Provides support for a parent/child structure over the beans

Operation Detail

startLogging

```
public void startLogging()
```

Start logging to a new log file. If currently logging close the exisitng file and open a new one.

stopLogging

```
public void stopLogging()
```

If currently logging stop, otherwise do nothing.

retrieveInstrumentationByFilter

```
public java.lang.String retrieveInstrumentationByFilter(boolean showVisibleOnly)
                                                 throws AdminException
```

Return a selected set of instrumentation data as XML.

Parameters

showVisibleOnly - show user visible instrumentations only (only instrumentations seen in Process Admin)

Returns

the set of instrumentation data as XML

Throws

AdminException

retrieveInstrumentationAll

```
public java.lang.String retrieveInstrumentationAll()
                                                 throws AdminException
```

Return the set of instrumentation data as XML.

Returns

the set of instrumentation data as XML

Throws

AdminException

retrieveInstrumentationByFilterAsJSON

```
public java.lang.String retrieveInstrumentationByFilterAsJSON(boolean showVisibleOnly)
                                                 throws AdminException
```

Return a selected set of instrumentation data as JSON.

Parameters

showVisibleOnly - show user visible instrumentations only (only instrumentations seen in Process Admin)

Returns

the set of instrumentation data as JSON

Throws

AdminException

retrieveInstrumentationAllAsJSON

```
public java.lang.String retrieveInstrumentationAllAsJSON()
                                                 throws AdminException
```

Return the set instrumentation data as JSON.

Returns

the set of instrumentation data as JSON

Throws
AdminException

ProcessMonitor MBean

IBM Business Process Manager provides a programming interface for accessing process monitor data.

ObjectName:

```
com.lombardisoftware:Scope=ENVIRONMENT_SERVER,Name=ProcessMonitor,cell=[cell],
                              Type=ProcessMonitor,node=[node],process=[server]
```

MBean **ProcessMonitor**

Information on the management interface of the MBean

Data retrieved complies to the following schema:

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema attributeFormDefault="unqualified" elementFormDefault="qualified"
 targetNamespace="http://www.ibm.com/bpm/v1/process_monitor" xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:tns="http://www.ibm.com/bpm/v1/process_monitor">
 <xsd:element name="processMonitor" type="tns:processMonitorType" />

 <xsd:complexType name="processMonitorType">
  <xsd:sequence>
   <xsd:element name="process" type="tns:processType" minOccurs="0" maxOccurs="unbounded"/>
   <xsd:element name="service" type="tns:serviceType"  minOccurs="0" maxOccurs="unbounded"/>
  </xsd:sequence>
  <xsd:attribute type="xsd:string" name="server" />
  <xsd:attribute type="xsd:string" name="id" />
  <xsd:attribute type="xsd:boolean" name="showDetails" />
  <xsd:attribute type="xsd:long" name="durationExceededInSeconds" />
  <xsd:attribute type="xsd:long" name="stepExceeded" />
 </xsd:complexType>

 <xsd:complexType name="stepType">
  <xsd:attribute type="xsd:string" name="name"/>
  <xsd:attribute type="xsd:string" name="type"/>
  <xsd:attribute type="xsd:dateTime" name="lastEnterTime" />
  <xsd:attribute type="xsd:string" name="lastDuration"/>
  <xsd:attribute type="xsd:string" name="totalDuration"/>
  <xsd:attribute type="xsd:long" name="steps"/>
 </xsd:complexType>

 <xsd:complexType name="serviceType">
  <xsd:sequence>
   <xsd:element name="step" type="tns:stepType" minOccurs="0" maxOccurs="unbounded"/>
  </xsd:sequence>
  <xsd:attribute type="xsd:string" name="processApplication"/>
  <xsd:attribute type="xsd:string" name="name" />
  <xsd:attribute type="xsd:string" name="state"  />
  <xsd:attribute type="xsd:string" name="contextID"  />
  <xsd:attribute type="xsd:dateTime" name="lastEnterTime"/>
  <xsd:attribute type="xsd:string" name="duration"/>
  <xsd:attribute type="xsd:string" name="lastDuration"/>
  <xsd:attribute type="xsd:string" name="totalDuration"/>
  <xsd:attribute type="xsd:long" name="steps"/>
 </xsd:complexType>

 <xsd:complexType name="processType">
  <xsd:sequence>
   <xsd:element name="step" type="tns:stepType" minOccurs="0" maxOccurs="unbounded"/>
   <xsd:element name="service" type="tns:serviceType" minOccurs="0" maxOccurs="unbounded"/>
  </xsd:sequence>
  <xsd:attribute type="xsd:string" name="processApplication"/>
  <xsd:attribute type="xsd:string" name="name"/>
  <xsd:attribute type="xsd:long" name="instanceID"/>
  <xsd:attribute type="xsd:string" name="state"/>
  <xsd:attribute type="xsd:dateTime" name="lastEnterTime"/>
  <xsd:attribute type="xsd:string" name="totalDuration"/>
  <xsd:attribute type="xsd:long" name="steps"/>
 </xsd:complexType>
</xsd:schema>
```

Table 3. Attribute summary

Attribute Summary	
java.lang.String	**DisplayName**
boolean	**Monitoring**
javax.management.ObjectName	**ParentObjectName**

Table 4. Operation summary

Operation Summary	
void	**haltProcess(java.lang.String processInstanceID)** Attempt to halt running of a process instance.
void	**haltProcessAndAssociatedServices(java.lang.String processInstanceID)** Attempt to halt running of a process instance.
void	**haltService(java.lang.String serviceContextID)** Attempt to halt running of a service execution.
java.lang.String	**retrieveMonitorAll()** Return all process monitor data
java.lang.String	**retrieveMonitorAllAsJSON()** Return all process monitor data
java.lang.String	**retrieveMonitorByFilter(boolean showDetails, long durationExceededInSeconds, long stepExceeded)** Return a selected set of process monitor data
java.lang.String	**retrieveMonitorByFilterAsJSON(boolean showDetails, long durationExceededInSeconds, long stepExceeded)** Return a selected set of process monitor data
void	**startMonitoring()**
void	**stopMonitoring()**

Table 5. Operations inherited from class java.lang.Object

Operations inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Attribute Detail

Monitoring
public boolean Monitoring

DisplayName
public java.lang.String DisplayName

ParentObjectName
public javax.management.ObjectName ParentObjectName

Operation Detail

startMonitoring

```
public void startMonitoring()
```

stopMonitoring

```
public void stopMonitoring()
```

retrieveMonitorAll

```
public java.lang.String retrieveMonitorAll()
                     throws AdminException
```

Return all process monitor data

Returns

all process monitor data as XML

Throws

AdminException

retrieveMonitorByFilter

```
public java.lang.String retrieveMonitorByFilter(boolean showDetails,
                                        long durationExceededInSeconds,
                                        long stepExceeded)
                                    throws AdminException
```

Return a selected set of a selected set process monitor data.

Parameters

showDetails - show information about individual steps in process instance/service

durationExceededInSeconds - only show process instance/service that exceeded a certain time in seconds

stepExceeded - only show process instance/service that exceeded a certain steps

Returns

all process monitor data as XML

Throws

AdminException

retrieveMonitorAllAsJSON

```
public java.lang.String retrieveMonitorAllAsJSON()
                               throws AdminException
```

Return all process monitor data

Returns

all process monitor data as JSON

Throws

AdminException

retrieveMonitorByFilterAsJSON

```
public java.lang.String retrieveMonitorByFilterAsJSON(boolean showDetails,
                                                     long durationExceededInSeconds,
                                                     long stepExceeded)
                                                 throws AdminException
```

Return a selected set of process monitor data

Parameters

showDetails - show information about individual steps in process instance/service

durationExceededInSeconds - only show process instance/service that exceeded a certain time in seconds

stepExceeded - only show process instance/service that exceeded a certain steps

Returns
all process monitor data as JSON

Throws
AdminException

haltProcess

```
public void haltProcess(java.lang.String processInstanceID)
                 throws AdminException
```

Attempt to halt running of a process instance. This will fail if the process is not active, or is in the middle of executing a task.

Throws
AdminException

haltProcessAndAssociatedServices

```
public void haltProcessAndAssociatedServices(java.lang.String processInstanceID)
                                      throws AdminException
```

Attempt to halt running of a process instance. This will fail if the process is not active, or is in the middle of executing a task.

Throws
AdminException

haltService

```
public void haltService(java.lang.String serviceContextID)
                 throws AdminException
```

Attempt to halt running of a service execution. This will fail if the service is not active, or is in the middle of executing a step.

Throws
AdminException