**package** com.softwareag.platform.management.is.monitoring.impl;

**public** **class** OSGIISRuntimeMonitorImpl **extends** BaseConfigurationManager **implements** RuntimeMonitor {

**private** RuntimeStatus getCurrentStatus(RuntimeComponentEx component) **throws** PlatformManagerException {

Case 1: shutdown.anchor exists

**if** (isShutdownLockExist(component))

**check for** profiles\SPM\bin\shutdown.anchor

{

Read The process id from the shutdown.anchor file

Check from the pid if process is running then - RuntimeStatus.***STOPPING***;

Else - RuntimeStatus.***FAILED***;

In case of any exception - **return** RuntimeStatus.***UNKNOWN***;

}

Case2: When shutdown.anchor doesn’t exist

**if** (isWrapperLockExist(component))

**check for** profiles\SPM\bin\wrapper.anchor

{

//WRAPPER\_LOCK file exist but process is not running. **return** ***FAILED***;

In case of any exception - **return** RuntimeStatus.***UNKNOWN***;

//if process is running wrapperLockFile

Check for activePorts and open\_port\_counts.

If open port == 0 implies The process is running but there is no open ports, return ***UNRESPONSIVE.***

**openPorts < activePorts.size() //some of the ports are not opened yet.**

**return RuntimeStatus.*STARTING*;**

in all other case(The process is running and all ports are opened) **return** ***ONLINE***;

}

Case3: shutdown.anchor and wrapper.anchor both does not exists

**if** (openPorts == 0) {

// All ports are closed

**return** RuntimeStatus.***STOPPED***;

}

Else // There is no WRAPPER\_LOCK file but at least one port is open

**return** RuntimeStatus.***FAILED***;

}

**public** RuntimeStatus getRuntimeStatus(RuntimeComponentEx component) **throws** PlatformManagerException {

**if** (component.getCategory() == Category.***ENGINE***) {

**return** **getEngineRuntimeStatus**(component);

}

Category.***ENGINE =*** OSGI-IS\_default-DigitalEventServices, OSGI-IS\_default-WmMonitor

If component Category is not engine

In loop(for 3.5 sec in every 100ms) check for CurrentStatus of component by calling **above** getCurrentStatus(component) method

}

**private** **synchronized** RuntimeStatus **getEngineRuntimeStatus**(RuntimeComponentEx component) **throws** PlatformManagerException {

JMXConnector jmxConnector = **null**;

Check the **parent-status** of RuntimeComponent **component** if it is **not** **online** return this as status of the component. (This in turn again calls getCurrentStatus by passing parent)**---> This might be the case**

If parent is online

* Get JmxConnector of component.
  + In case of any **exception** or JmxConnector is **null** return **unknown** status
* Get the subsystem of component using JmxConnector
* Check the status of subsystem and return its status as component’s status.

}

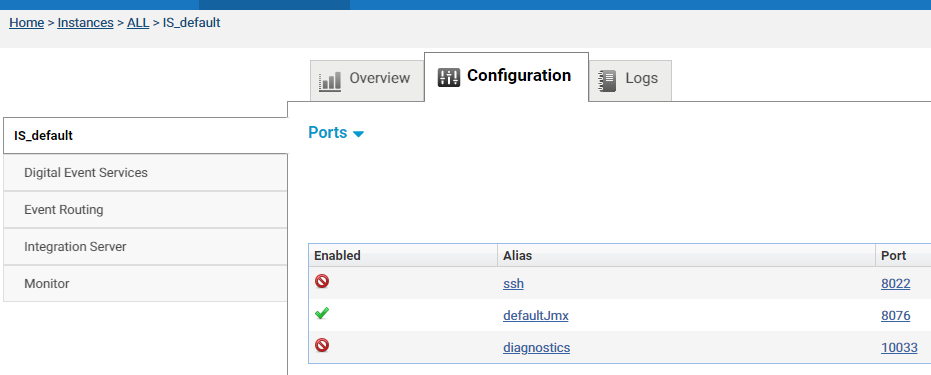
**private** **synchronized** List<Port> getActivePortConfigurations(RuntimeComponentEx component) {

List<Port> activePorts = **new** ArrayList<>();

If component’s ConfigurationTypeId is ***COMMON\_PORTS*** then only they are considered to be active and added to activePorts based on below checks.

* Port should be enabled
* Even if enabled, port-type should not be diagnostics

}



**Behavior at Cx env :**



Except IS all sub-components status are STARTING, including parent IS\_default.

Analysis : By keeping yellow highlight of above code-flow in mind

All runtime components are of type engine except IS, So they calls **getEngineRuntimeStatus**

Status of every components are set to STARTING as the parent has STARTING status.

IS type is not engine so it is not dependent on it’s parent status. IS status is evaluated separately and it is found ONLINE.

So **the root-cause** of above issue is – status of IS\_default is STARTING.

As per code there is only one case where this can happen (as highlighted in code flow analysis)

**openPorts < activePorts.size()**

active ports are jmx port (see below snap-shot), and looks like this port is not open.

How to validate this assumption???

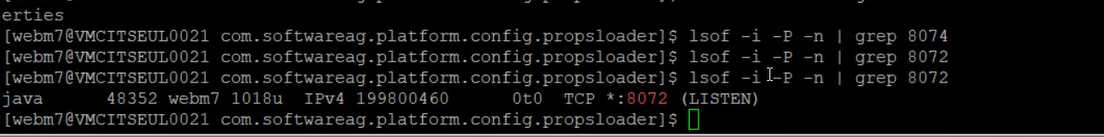
Ports configured at Cx Env:

A screenshot of a computer

Description automatically generated with medium confidence

When https port 8074 is disabled the issue got resolved.

So the above analysis holds true. Reason port 8074 was not open/free.



The main reason is TCP port is not listening to the configured port in https

Changing the port for https doesn't resolve the issue, but disabling the https port does resolve the issue.

We analyzed further and found that any port which is configured with https is not listening, this is verified by lsof command.

So there is something wrong with https port configuration, which is yet to be discovered.

Will update here as soon as we get any clue.

Further investigation has proved that no port is listening for https.

Further investigation has shown that any port is configured with https is not listening, this is verified by lsof command.

This is the root cause of this issue and I am investigating on this.

Will update here as soon as we get any clue. because if we configured any port with