SDN Community Contribution

(This is not an official SAP document.)

Disclaimer & Liability Notice

How to Handle the JCO Server Status

Applies To:

Implements the StatusChanged interface. The server keeps track of its own status in “human readable form” and notifies its controller on status change

Summary

There are two separate parts to keep track of the status…

One is in the controller. It acts as a pass through for the JCO notification but is a property change listener for the String property in readable form.

The server decodes the status and sets the String property and fires the propertyChangeOccured event

By: F.J. Brandelik

Company:

Date: June 28th 2005

Table of Contents

How to keep track of the state of your JCO Server

The Controller

The controller creates the server.

The controller subscribes to the JCO.ServerStateChanged event and is a pass-through for the server on this event.

It subscribes the server’s propertyChange event to allways carry the latest status…

It keeps track of the current and previous states and logs information when the service is interrupted or when the service resumes (presumably after a network interruption)

Controller Code

```
/* here is a controller setup
It will create the server, register it to maintain its state and register itself to be updated on state change:
*/
        server.addPropertyChangeListener(this);
        JCO.addServerStateChangedListener(this);

        /**
         * On server state change checks whether the state change
         * is for its controlled server and passes along the old and new
         * values to implement the change in human readable form.
         */
        public void serverStateChangeOccurred(
                JCO.Server theserver,
                int stateold,
                int statenew) {
                if (theserver == server) {
                        server.serverStateChangeOccurred(theserver, stateold, statenew);
                        oldstate = stateold;
                        newstate = statenew;

                } //endif

        } //end method
        /**
         * Uses the property change method to update the state.
         * Logs if the change is due to loss of connectivity.
         * Logs if the connectivity is restored
         * @see java.beans.PropertyChangeListener#propertyChange(PropertyChangeEvent)
         */
        public void propertyChange(PropertyChangeEvent evt) {
                String newstate = curstate;
                if (evt.getSource() == server
                        && (evt.getPropertyName().equals("sapIdocServerStatus"))) {
                        newstate = server.getSapIdocServerStatus();
                        if ((newstate.indexOf("Disconnected") >= 0
                                        || newstate.indexOf("Suspended") >= 0)
                                        && curstate.indexOf("Connected") >= 0) {
                                //log state change
                                String msg =
                                        server.poolname
                                                + " Service interrupted for SID "
                                                + server.SID;
                                server.logger.warning(msg);
                        } //endif
                        if ((curstate.indexOf("Disconnected")>=0
                                || curstate.indexOf("Suspended")>=0)
                                && newstate.indexOf("Connected")>=0){
```

```
                                        //log state change
                                        String msg = server.poolname + " Service resumed for SID " + server.SID;
                                        server.logger.warning(msg);
                                }//endif
        } //endif
        curstate = newstate;

} //end method

/**
 * Returns the current state of the server.
 * @return String
 */
public String getCurstate() {
        return curstate;
}
```

The Server

The JCO Server, in this case an Idoc server, formats the Idoc as xml, and dumps it into a queue in WebSphereMQ. The server keeps track of its own status, in human readable format and notifies all of it's property listeners to update them of any status change.

The server translates the binary status into a status that is humanly readable (String)…

The Server side code

/* now on how to handle the state change on the server...
Remember the server keeps track of its state in human readable form

```
/**
 * serverStateChangeOccurred method comment.
 */
public void serverStateChangeOccurred(
        JCO.Server arg1,
        int oldstate,
        int state) {
        JCO.Server myserver = null;
        if (getMyserverthread() != null)
                myserver = getMyserverthread().getServer();
        if (arg1 == this || arg1 == myserver) {

                String mystatus = "";

                if (statecheck(state, JCO.STATE_DISCONNECTED))
                        mystatus = mystatus + "Disconnected ";
                if (statecheck(state, JCO.STATE_CONNECTED))
                        mystatus = mystatus + "Connected ";
                if (statecheck(state, JCO.STATE_BUSY))
                        mystatus = mystatus + "Busy ";
                if (statecheck(state, JCO.STATE_LISTENING))
```

```
                    mystatus = mystatus + "Listening ";
                if (statecheck(state, JCO.STATE_SUSPENDED))
                    mystatus = mystatus + "Suspended ";
                if (statecheck(state, JCO.STATE_TRANSACTION))
                    mystatus = mystatus + "Transaction";
                if (statecheck(state, JCO.STATE_USED))
                    mystatus = mystatus + "Used";

                if (mystatus.equals(""))
                    mystatus = "Not Initialized";

                setSapIdocServerStatus(mystatus);

        } //endif
} //end method

/**
* Insert the method's description here.
* Creation date: (07/13/2003 01:02:09 PM)
* @return byte
* @param currstate int
*/
private boolean statecheck(int currstate, byte checkstate) {

        boolean test = (currstate & checkstate) == checkstate;

        return test;
}

/**
 * Insert the method's description here.
 * Creation date: (07/13/2003 01:29:38 PM)
 * @param curstat java.lang.String
 */
private void setSapIdocServerStatus(String curstat) {
        String oldstat = getSapIdocServerStatus();
        String propname = "sapIdocServerStatus";
        fieldSapIdocServerStatus = curstat;
        getPropertyChange().firePropertyChange(propname, oldstat, curstat);

}

/**
 * Accessor for the propertyChange field.
 */
protected java.beans.PropertyChangeSupport getPropertyChange() {
        if (propertyChange == null) {
                propertyChange = new java.beans.PropertyChangeSupport(this);
        };
        return propertyChange;
}
```

Author Bio

The author has accumulated seven years of ABAP experience with one of the big five consultant companies. He is now active in the IT Architecture and J2EE world, working for one of the big media companies in the US.