



## APPENDIX **A**

# Sample Client Program

---

This appendix provides a sample client program using the API.

### **Example A-1 Zone Manager Web Services API**

```
package com.cisco.dcbu.smis.jaxws.test;

import static com.cisco.dcbu.smis.jaxws.test.ZoneWsUtil.*;

import java.io.BufferedReader;
import java.io.InputStreamReader;
import com.cisco.dcbu.smis.jaxws.ep.Fabric;
import com.cisco.dcbu.smis.jaxws.ep.OperationStatus;
import com.cisco.dcbu.smis.jaxws.ep.Vsan;
import com.cisco.dcbu.smis.jaxws.ep.WwnKey;
import com.cisco.dcbu.smis.jaxws.ep.Zone;
import com.cisco.dcbu.smis.jaxws.ep.ZoneMember;
import com.cisco.dcbu.smis.jaxws.ep.ZoneSet;
import com.cisco.dcbu.smis.jaxws.ep._switch;
public class ZoneWsClient {

    /**
     * Default constructor
     */
    public ZoneWsClient() {
        input_ = new BufferedReader(new InputStreamReader(System.in));
    }

    /**
     * Main method is the starting point of execution.
     * @param args
     * @throws Exception
     */
    public static void main(String[] args) throws Exception {
        menu();
    }

    /**
     * This method show all available zone webservice as user menu.
     * @throws Exception
     */
    public static void menu() throws Exception {
        ZoneWsClient zwsClient = new ZoneWsClient();
        int choice = 0;

        for (;;) {
```

## Send documentation comments to [dcnm-san-docfeedback@cisco.com](mailto:dcnm-san-docfeedback@cisco.com)

```

        header("====Zone WS
Client====");
        System.out.println("Available Zone services:\n\n " + "\t1:Discover Zones in
fabric\n " + "\t2:Discover Zonesets in fabric\n " + "\t3:Discover Zonemembers in fabric\n
"
        + "\t4:Create new Zone in fabric\n " + "\t5:Create new ZoneSet in
fabric\n " + "\t6:Add Zonemember to existing Zone in fabric\n " + "\t7:Add existing Zone
to existing Zoneset in fabric\n "
        + "\t8:Activate/Deactivate a ZoneSet in fabric\n " + "\t9:Manual Zone
Cache Invalidation\n " + "\t10:Exit ");
        System.out.println("\n\tPlease enter your Choice: ");
        try {
            choice = Integer.parseInt(input_.readLine());
        } catch (NumberFormatException e) {
            System.out.println("Enter Integer choices only...");
            continue;
        }

        switch (choice) {
        case 1:
            zwsClient.displayZone();
            break;
        case 2:
            zwsClient.displayZonesets();
            break;
        case 3:
            zwsClient.displayZonemembers();
            break;
        case 4:
            zwsClient.createZone();
            break;
        case 5:
            zwsClient.createZoneSet();
            break;
        case 6:
            zwsClient.addZoneMember();
            break;
        case 7:
            zwsClient.addZone();
            break;
        case 8:
            zwsClient.activateAndDeactivateZoneSet();
            trailer("ACTIVATE/DEACTIVATE ZONESET");
            break;
        case 9:
            zwsClient.zoneCacheInvalidation();
            break;
        case 10:
            System.out.println("Exiting ZoneWSClient..");
            trailer("Zone WS CLIENT");
            System.exit(1);
        default:
            System.out.println("Wrong Choice entered.");
        }
    }

}

/**
 * This method invalidates zonedata cache by making use of
 * ZoneManagerWS: invalidateZoneDataCache()
 */
public void zoneCacheInvalidation() {
    try {

```

**Send documentation comments to [dcnm-san-docfeedback@cisco.com](mailto:dcnm-san-docfeedback@cisco.com)**

```

        System.out.println("Invalidation of Zone data cache started");
        zoneManager.invalidateZoneDataCache();
        System.out.println("Succesfully Invalidated Zone data cache");
    } catch (Exception e) {
        exception(e);
    }finally{
        trailer("Zone Cache Invalidation");
    }
}

/**
 * This method takes prompts user to enter already existing zoneset name to
 * activate/deactivate.
 * ZoneManagerWS: ActivateZoneset()
 * ZoneManagerWS: deActivateZoneset()
 */
public void activateAndDeactivateZoneSet() throws Exception {
    int flag = 0,choice=0;
    do{
        try {
            System.out.println("Enter your choice\n 1:Activate a ZoneSet\n 2:Deactivate a
ZoneSet\n 3:Main menu");
            choice = Integer.parseInt(input_.readLine());
            OperationStatus oper = null;
            if(choice==3)return;
            if (choice != 1 && choice != 2 && choice !=3) {
                System.out.println("Enter right Choice: 1 or 2 or 3");
                flag = 0;
                continue;
            }
            oper = activation(choice);
            if (oper != null){
                System.out.println("Zoneset activation/deactivation is done successfully.\n
Please wait for 1-5 minutes to get reflected.");
                flag =1;
            }
            else if (oper == null){
                System.out.println("Zoneset activation/deactivation is not successfull");
                flag =1;
            }
            break;
        } catch(NumberFormatException ne){
            System.out.println("NumberFormatException: Integer type expected.");
            continue;
        }
    } catch (Exception e) {
        exception(e);
        flag=0;
        continue;
    }
    }while(choice !=3||flag==0);
}

private OperationStatus activation(int choice) throws Exception {
    OperationStatus oper = null;
    System.out.println("Enter the ZoneSet name to be activated/deactivated ");
    String zName = input_.readLine();
    Fabric[] fabrics = getFabrics();
    if(fabrics!=null){
        for (Fabric fab : fabrics) {
            _switch[] sws = getSwitchesByFabric(fab);
            Vsan[] vsans = getVsans(fab);
            if(sws!=null){

```

**Send documentation comments to [dcnm-san-docfeedback@cisco.com](mailto:dcnm-san-docfeedback@cisco.com)**

```

        for (_switch sw : sws) {
            if(vsans!=null){
                for (Vsan vsan : vsans) {
                    ZoneSet enfZoneset = zoneManager.getEnfZoneSetDO(vsan.getKey());
                    ZoneSet[] zoneSets = zoneManager.getZoneSets(vsan.getKey(),new
WwnKey(sw.getWwn()));
                    if(zoneSets==null){
                        System.out.println("Zoneset is null for VSan
"+vsan.getKey().getVsanID());
                        continue;
                    }
                    for (ZoneSet zs : zoneSets) {
                        if (enfZoneset != null) {
                            if (enfZoneset.getName().equals(zName) && choice == 1 &&
enfZoneset.isActive()) {
                                System.out.println("Selected ZoneSet is already activated");
                                return null;
                            }
                            else if (!enfZoneset.getName().equals(zName) && choice == 2 &&
zs.getName().equals(zName)) {
                                System.out.println("Selected ZoneSet is already
deactivated");
                                return null;
                            }
                            } else if (enfZoneset == null && choice == 2) {
                                System.out.println("No ZoneSet is active to deactivate");
                                return null;
                            } else if (zs.getName().equals(zName) && choice == 2 &&
!zs.isActive()) {
                                System.out.println("Selected ZoneSet is already deactivated");
                                return null;
                            }
                            if (zs.getName().equals(zName)) {
                                if (choice == 1)
                                    oper = zoneManager.activateZoneset(vsan.getKey(), new
WwnKey(sw.getWwn()), zs.getName());
                                else if (choice == 2)
                                    oper = zoneManager.deActivateZoneset(vsan.getKey(), new
WwnKey(sw.getWwn()), zs.getName());
                                return oper;
                            }
                        }
                    }
                }
            }
        }

        System.out.println("Invalid Zoneset is selected");
    }
}
return oper;
}

/**
 * This method prompts user to enter a unique zone name and creates the
 * zone in default vsan.
 * ZoneManagerWS:createZone()
 */
public void createZone() {
    try {
        String zName=null;
        do{
            System.out.println("Enter the new Zone name");

```

**Send documentation comments to [dcnm-san-docfeedback@cisco.com](mailto:dcnm-san-docfeedback@cisco.com)**

```

zName= input_.readLine();
}while(zName.length()<1);
Fabric[] fabrics = getFabrics();
if(fabrics!=null){
for (Fabric fab : fabrics) {
    _switch[] sws = getSwitchesByFabric(fab);
    Vsan[] vsans = getVsans(fab);
    if(sws!=null){
    for (_switch sw : sws) {
        if(vsans!=null){
            for (Vsan vsan : vsans) {
                int qosPriority = -1;
                boolean readOnly = false;
                boolean broadcast = false;
                boolean qos = false;
                Zone zone1 = zoneManager.createZone(vsan.getKey(),new
WwnKey(sw.getWwn()), zName, readOnly, broadcast, qos, qosPriority);
                if (zone1 == null)
                    System.out.println("Failed to create Zone: " + zName);
                else
                    System.out.println("Zone " + zName + " created successfully in
Vsan:" + vsan.getKey().getVsanID() + ",Switch: " + sw.getName());
                return;
            }
        }
    }
}
} catch (Exception e) {
    exception(e);
}finally{
    trailer("CREATE ZONE");
}

}

/**
 * This method prompts user to enter an existing zone,zonememberid and zonemembertype
and creates a zonemember in zone provided.
 * ZoneManagerWS:addZoneMemberToZone
 */
public void addZoneMember() {
    try {
        for (;;) {
            System.out.println("\nEnter existing zonename to which member has to be
added\n");

            String zoneName = input_.readLine();
            Fabric[] fabrics = getFabrics();
            if(fabrics!=null){
            for (Fabric fab : fabrics) {
                _switch[] sws = getSwitchesByFabric(fab);
                Vsan[] vsans = getVsans(fab);
                if(sws!=null){
                for (_switch sw : sws) {
                    if(vsans!=null){
                        for (Vsan vsan : vsans) {
                            Zone existingzones[] = zoneManager.getZones(vsan.getKey(), new
WwnKey(sw.getWwn()));

                            if(existingzones==null){
                                continue;
                            }
                            boolean flag=false;
                            for (Zone zs : existingzones) {
                                if (zs.getName().equals(zoneName)) {

```

**Send documentation comments to [dcnm-san-docfeedback@cisco.com](mailto:dcnm-san-docfeedback@cisco.com)**

```

        flag=true;
        System.out.println("Enter zonemember type which is in
number format. \n 1. For WWN ");
        int zonememtype = Integer.parseInt(input_.readLine());
        String hexstr = null;
        byte[] zonememberid = null;
        if(zonememtype==1){
            System.out.println("Enter zonemember id which is in
hexa format Eg: 1122334455667788 or 11:22:33:44:55:66:77:88");
            hexstr=input_.readLine();
            if (hexstr.contains(":"))
                zonememberid = ZoneWsUtil.fromHexString(hexstr,
true);
            else
                zonememberid = ZoneWsUtil.fromHexString(hexstr,
false);
        }
        else{
            System.out.println("Demo ZoneWsClient do not support
other member types\n");
            continue;
        }

        ZoneMember zoneMemberAdded =
zoneManager.addZoneMemberToZone(vsan.getKey(),new WwnKey(sw.getWwn()),zoneName,
zonememtype, -1,-1, -1, zonememberid, null);
        if (zoneMemberAdded == null)
            System.out.println("Failed to add Zonemember to
specified zone");
        else
            System.out.println("ZoneMember added succesfully to
zone "+ zoneName);
        return;
    }
}
if(flag==false)
    System.out.println("Zone " + zoneName + " not found. Please
enter an existing zone");
    }}
    }}
}
} catch(NumberFormatException ne){
    System.out.println("NumberFormatException: Please check the input
type/format");
}
catch (Exception e) {
    exception(e);
}finally{
    trailer("ADD ZONE");
}
}

/**
 * This method prompts user to enter existing zoneset, existing zone
 * and adds zone to zoneset.
 * ZoneManagerWS:addZoneToZoneset()
 *
 */
public void addZone() {
    try {
        for (;;) {

```

**Send documentation comments to [dcnm-san-docfeedback@cisco.com](mailto:dcnm-san-docfeedback@cisco.com)**

```

        System.out.println("Enter already existing zoneset to which the zone has to
be added: ");
        String zonesetName = input_.readLine();
        Fabric[] fabrics = getFabrics();
        if(fabrics!=null){
        for (Fabric fab : fabrics) {
            _switch[] sws = getSwitchesByFabric(fab);
            Vsan[] vsans = getVsans(fab);
            if(sws!=null){
            for (_switch sw : sws) {
                if(vsans!=null){
                for (Vsan vsan : vsans) {
                    ZoneSet existingzonesets[] =
zoneManager.getZoneSets(vsan.getKey(), new WwnKey(sw.getWwn()));
                    for (ZoneSet zset : existingzonesets) {
                        if (zset.getName().equals(zonesetName)) {
                            System.out.println("Enter existing zone to be added to
zoneset: ");
                                String zonename = input_.readLine();
                                Zone existingzones[] =
zoneManager.getZones(vsan.getKey(),new WwnKey(sw.getWwn()));
                                for (Zone zone : existingzones) {
                                    if (zone.getName().equals(zonename)) {
                                        ZoneSet znst =
zoneManager.addZoneToZoneset(vsan.getKey(),new WwnKey(sw.getWwn()),zset, zone);
                                        if (znst == null)
                                            System.out.println("Failed to add Zone to
specified Zoneset");
                                            else
                                                System.out.println("Zone "+ zonename+ " added
succesfully to zoneset "+ zonesetName);
                                            return;
                                        }
                                    }
                                }
                                System.out.println("Zone "+ zonename+ " not found.
Please enter an existing zone");
                            }
                        }
                    }
                }
            }
        }
    }
}

        System.out.println("ZoneSet " + zonesetName+ " not found. Please enter an
existing zoneset");
    }
} catch (Exception e) {
    exception(e);
}finally{
    trailer("ADD ZONE");
}
}

/**
 * This method prompts user to enter zoneset name which will be created
 * in default vsan.
 * ZoneManagerWS:createZoneSet()
 */
public void createZoneSet() {
    try {
        System.out.println("Enter the new unique Zoneset name");
        String zonsetName =input_.readLine();
        Fabric[] fabrics = getFabrics();
    }
}

```

**Send documentation comments to [dcnm-san-docfeedback@cisco.com](mailto:dcnm-san-docfeedback@cisco.com)**

```

        if(fabrics!=null){
        for (Fabric fab : fabrics) {
            _switch[] sws = getSwitchesByFabric(fab);
            Vsan[] vsans = getVsans(fab);
            if(sws!=null){
                for (_switch sw : sws) {
                    if(vsans!=null){
                        for (Vsan vsan : vsans) {

                            ZoneSet newzoneset = zoneManager.createZoneSet(vsan.getKey(), new
WwnKey(sw.getWwn()), zonsetName);
                            if (newzoneset == null)
                                System.out.println("Failed to create Zoneset: " + zonsetName);
                            else
                                System.out.println("ZoneSet " + zonsetName+ " created in
Vsan:"+ vsan.getKey().getVsanID() + ",Switch: " + sw.getName());
                            return;
                        }
                    }
                }
            }
        } catch (Exception e) {
            exception(e);
        }finally{
            trailer("CREATE ZONESET");
        }
    }

/**
 * This method displays the Zone member setting data in the fabric.
 * ZoneManagerWS:getZoneMembers()
 */
public void displayZonemembers() {
    try {
        Fabric[] fabrics = getFabrics();
        if(fabrics!=null){
            for (Fabric fab : fabrics) {
                _switch[] sws = getSwitchesByFabric(fab);
                Vsan[] vsans = getVsans(fab);
                if(vsans!=null){
                    for (Vsan vsan : vsans) {
                        Zone[] enfZones = getEnfZones(vsan);
                        if (enfZones != null) {
                            for (Zone ezone : enfZones) {
                                ZoneMember[] zms = getZoneMembers(ezone);
                                for (ZoneMember zm : zms)
                                    System.out.println("vsan id:"+ vsan.getKey().getVsanID()+
",vsan wwn:"+ vsan.getKey().getPWwn().getValue()+ ",Zone Name:" + ezone.getName()
                                        + ", ZoneMemberType: " + zm.getType()+ ",
ZoneMemberId: " + ZoneWsUtil.toHexString(zm.getId()+ ",Status:" + ezone.isActive());
                            }
                        }
                    }
                }
            }
            if(sws!=null){
                for (_switch sw : sws) {
                    if(vsans!=null){
                        for (Vsan vsan : vsans) {
                            Zone[] zones = getZones(vsan, sw);
                            if (zones == null)
                                return;
                            for (Zone zone : zones) {
                                ZoneMember[] zms = getZoneMembers(zone);
                                for (ZoneMember zm : zms)
                                    System.out.println("vsan id:"+ vsan.getKey().getVsanID()+
",vsan wwn:"+ vsan.getKey().getPWwn().getValue()+ ",Zone Name:" + zone.getName()

```



**Send documentation comments to [dcnm-san-docfeedback@cisco.com](mailto:dcnm-san-docfeedback@cisco.com)**

```

        + ", ZoneMemberType: " + zm.getType() + ",
ZoneMemberId: "+ ZoneWsUtil.toHexString(zm.getId())+ ",Status:" + zone.isActive());
    }
    }}
    }}
    } catch (Exception e) {
        exception(e);
    }finally{
        trailer("Zone Members");
    }

}

private ZoneMember[] getZoneMembers(Zone zone) {
    header("Now getting ZoneMembers for Zone " + zone.getName());
    ZoneMember[] zoneMemz = null;
    try {
        zoneMemz = zone.getMembers();
        if (zoneMemz != null)
            System.out.println(zoneMemz.length+ " ZoneMembers found for the Zone " +
zone.getName());
        else
            System.out.println("No ZoneMembers found for the Zone "+ zone.getName());
        return zoneMemz;
    } catch (Exception e) {
        exception(e);
    }
    return null;
}

/**
 * This method displays the active and local zonesets in the fabric.
 * ZoneManagerWS:getZoneSets()
 */
public void displayZonesets() {
    try {
        Fabric[] fabrics = getFabrics();
        if(fabrics!=null){
            for (Fabric fab : fabrics) {
                _switch[] sws = getSwitchesByFabric(fab);
                Vsan[] vsans = getVsans(fab);
                if(vsans!=null){
                    for (Vsan vsan : vsans) {
                        getEnfZonesets(vsan);
                    }
                }
                if(sws!=null){
                    for (_switch sw : sws) {
                        if(vsans!=null){
                            for (Vsan vsan : vsans) {
                                ZoneSet[] zoneSets = zoneManager.getZoneSets(vsan.getKey(), new
WwnKey(sw.getWwn()));
                                header("Now getting local ZoneSets for Vsan "+
vsan.getKey().getVsanID());
                                if (zoneSets == null) {
                                    System.out.println("No Zoneset found for switch"+ sw.getName()
+ " Vsan "+ vsan.getKey().getVsanID());
                                }
                                else{
                                    System.out.println(zoneSets.length+ " Zonesets found for the switch
"+ sw.getName() + " Vsan "+ vsan.getKey().getVsanID());
                                    for (ZoneSet zoneset : zoneSets) {

```

**Send documentation comments to [dcnm-san-docfeedback@cisco.com](mailto:dcnm-san-docfeedback@cisco.com)**

```

        System.out.println("VsanId:" + vsan.getKey().getVsanID() +
",Vsan WWN:" + vsan.getKey().getPwwn().getValue() + ",ZonesetName:" + zoneset.getName()+
",Status:" + zoneset.isActive());
    }
    }
    }}
    }}
} catch (Exception e) {
    exception(e);
}finally{
    trailer("ZONESETS");
}

}

private static Fabric[] getFabrics() {
    try {
        Fabric[] fabrics = san.getFabrics();
        if (fabrics == null) {
            System.out.println("fabrics is null" + fabrics.length);
            trailer("No Fabric - Return to Main Menu");
            menu();
        }
        return fabrics;
    } catch (Exception e) {
        exception(e);
    }
    return null;
}

private static _switch[] getSwitchesByFabric(Fabric fabric) {
    try {
        _switch[] sws = san.getSwitchesByFabric(fabric.getFabricKey());
        if (sws == null)
            System.out.println("No switches found for fabric");
        else {
            header(sws.length + " switch(es) found for fabric");
            for (_switch sw : sws) {
                System.out.println("Name:" + sw.getName() + ",IP:" + sw.getIpAddress() +
",WWN:" + sw.getPwwn().getValue());
            }
            return sws;
        } catch (Exception e) {
            exception(e);
        }
        return null;
    }

private static Vsan[] getVsans(Fabric fabric) {
    try {
        Vsan[] vsans = san.getVsans(fabric.getFabricKey());
        if (vsans == null)
            System.out.println("No vsans found for fabric");
        else {
            header(vsans.length + " vsan(s) found for fabric");
            for (Vsan vsan : vsans) {
                System.out.println("VanID:" + vsan.getKey().getVsanID()+ ",VsanWWN:" +
vsan.getKey().getPwwn().getValue());
            }
        }
    }
}

```

**Send documentation comments to [dcnm-san-docfeedback@cisco.com](mailto:dcnm-san-docfeedback@cisco.com)**

```

        return vsans;
    } catch (Exception e) {
        exception(e);
    }
    return null;
}

/**
 * This method displays all the active and local zones in the fabric.
 * ZoneManagerWS:getEnfZones()
 * ZoneManagerWS:getZones()
 */
public void displayZone() throws Exception {

    Fabric[] fabrics=getFabrics();
    if(fabrics!=null){
        for (Fabric fabric : fabrics) {
            System.out.println("Discovered the Fabric with WWN:" +
fabric.getSeedSwWwn().getValue());
            _switch[] sws = getSwitchesByFabric(fabric);
            Vsan[] vsans = getVsans(fabric);
            if(vsans!=null){
                for (Vsan vsan : vsans) {
                    Zone[] enfZones = getEnfZones(vsan);
                    if (enfZones != null) {
                        for (Zone ezone : enfZones) {
                            System.out.println("vsan id:"+ vsan.getKey().getVsanID() + ",vsan
wnn:"+ vsan.getKey().getPwwn().getValue()+ ",Zone Name:" + ezone.getName() + ",Status:"+
ezone.isActive());
                        }
                    }
                }
            }
            if(sws!=null){
                for (_switch sw : sws) {
                    if(vsans!=null){
                        for (Vsan vsan : vsans) {
                            Zone[] zones = getZones(vsan, sw);
                            if (zones == null) continue;
                            for (Zone zone : zones) {
                                System.out.println("vsan id:"+ vsan.getKey().getVsanID() + ",vsan
wnn:"+ vsan.getKey().getPwwn().getValue()+ ",Zone Name:" + zone.getName() + ",Status:"+
zone.isActive());
                            }
                        }
                    }
                }
            }
            trailer("ZONES");
        }

private Zone[] getEnfZones(Vsan vsan) throws Exception {
    header(" Getting active zones for Vsan " + vsan.getKey().getVsanID());
    Zone[] enfZones = null;
    try {
        enfZones = zoneManager.getEnfZoneSet(vsan.getKey());
        if (enfZones != null)
            System.out.println(enfZones.length+ " Active zones found for the Vsan "+
vsan.getKey().getVsanID());
        else
            System.out.println("No Active zones found");
        return enfZones;
    } catch (Exception e) {

```

**Send documentation comments to [dcnm-san-docfeedback@cisco.com](mailto:dcnm-san-docfeedback@cisco.com)**

```

        exception(e);
    }
    return null;
}

private ZoneSet getEnfZonesets(Vsan vsan) throws Exception {
    header("Now getting active ZoneSets for Vsan " + vsan.getKey().getVsanID());
    ZoneSet enfZoneset = null;
    try {
        enfZoneset = zoneManager.getEnfZoneSetDO(vsan.getKey());
        if (enfZoneset != null)
            System.out.println("Active zoneset for the VsanId: "+
vsan.getKey().getVsanID() + ", Pwwn: "+ vsan.getKey().getPwwn().getValue()
                + ", ZonesetName: " + enfZoneset.getName()+ ", Status " +
enfZoneset.isActive());
        else
            System.out.println("Active zoneset is null");
        return enfZoneset;
    } catch (Exception e) {
        exception(e);
    }
    return null;
}

private static Zone[] getZones(Vsan vsan, _switch sw) {
    header("Now getting Local Zones for switch " + sw.getName() + " vsan "+
vsan.getKey().getVsanID());
    try {
        Zone[] zones = zoneManager.getZones(vsan.getKey(), new WwnKey(sw.getWwn()));
        if (zones != null)
            System.out.println("Found " + zones.length + " zones");
        else
            System.out.println("No zones found ");
        return zones;
    } catch (Exception e) {
        exception(e);
    }
    return null;
}

static {
    getCredentials();
    getLoginService();
    getLogin();
    getZoneManagerService();
    getSanService();
}
}

public class ZoneWsUtil {

    public static String url = null;

    public static String username = null;

    public static String password = null;

    public static String token = null;

    public static Logon logon = null;

    public static San san = null;

    public static ZoneManager zoneManager = null;

```

***Send documentation comments to [dcnm-san-docfeedback@cisco.com](mailto:dcnm-san-docfeedback@cisco.com)***

```

static BufferedReader input_ = null;

static byte[] fromHexString(String hexStr, boolean hasColon)
    throws NumberFormatException {
    hexStr = hexStr.toLowerCase();
    int len = hexStr.length();
    byte bytes[] = new byte[hasColon ? ((len / 3) + 1) : (len / 2)];
    int sPos = 0; // position in hexStr
    int bPos = 0; // position in bytes
    try {
        while (sPos < len) {
            char a = hexStr.charAt(sPos);
            char b = hexStr.charAt(sPos + 1);
            if (hasColon && (a == ':' || b == ':'))
                throw new NumberFormatException("bad Hex format");
            int v1 = Character.digit(a, 16);
            int v2 = Character.digit(b, 16);
            if (v1 < 0 || v2 < 0)
                throw new NumberFormatException("bad Hex format");
            int v3 = (int) (v1 * 16 + v2);
            bytes[bPos] = (byte) v3;
            sPos += hasColon ? 3 : 2;
            bPos++;
        }
    } catch (Exception ex) {
        throw new NumberFormatException("bad Hex format");
    }

    if (bPos < bytes.length)
        throw new NumberFormatException("bad Hex format");
    return bytes;
}

static String toHexString(byte[] inputByte) {
    char[] HEX_DIGIT = { '0', '1', '2', '3', '4', '5', '6', '7', '8', '9',
        'a', 'b', 'c', 'd', 'e', 'f' };
    if (inputByte == null || inputByte.length == 0)
        return null;
    StringBuffer outputstr = new StringBuffer(inputByte.length * 3);
    for (int i = 0; i < inputByte.length; i++) {
        int n = (int) (inputByte[i] & 0xFF);
        outputstr.append(HEX_DIGIT[(n >> 4) & 0x0F]);
        outputstr.append(HEX_DIGIT[n & 0x0F]);
        if (i + 1 < inputByte.length)
            outputstr.append(':');
    }
    return outputstr.toString();
}

static void header(String text) {
    System.out.println(" ");
    System.out.println(text);
}

static void trailer(String string) {
    System.out.println("\n***** END OF "
        + string + " *****");
}

/**
 * This static method is used read the login credentials from the properties
 * file named "configuration.properties"
 */
public static void getCredentials() {

```

**Send documentation comments to [dcnm-san-docfeedback@cisco.com](mailto:dcnm-san-docfeedback@cisco.com)**

```

Properties props = new Properties();
try {
    if(props!=null){
        System.out.println("Reading credentials from the 'configuration.properties'
file");
        props.load(new FileInputStream("configuration.properties"));
        url = props.getProperty("FMServer-ipadress");
        username = props.getProperty("UserName");
        password = props.getProperty("Password");
    }
}
catch (FileNotFoundException fe) {
    System.out.println("ZoneWsClient: 'configuration.properties' File Not
found!!");
    url="localhost";
    username="admin";
    password="cisco123";
}
catch (IOException e) {
    exception(e);
}
}

/**
 * This method is to check LogonWSService is available or not. If available
 * bind to the stub.
 */
public static void getLoginService() {

    try {
        String loginServiceAddr = "http://" + url
            + "/LogonWSService/LogonWS";

        LogonServiceLocator logonService = new LogonServiceLocator();
        logonService.setEndpointAddress("LogonPort", loginServiceAddr);

        logon = logonService.getLogonPort();

        if (logon == null)
            System.out.println("login service not available" + logon);
        else
            System.out.println("Logon success.");

    } catch (Exception e) {
        e.printStackTrace();
    }
}

/**
 * This method is to get login token.
 */
public static void getLogin() {

    try {
        if(logon!=null){
            token = logon.requestToken(username, password, 1000000000L);
            if (token == null)
                System.out.println("Token not found" + token);
            else {
                System.out.println("Token found for configured user. ");
            }
        }
    } catch (Exception e) {

```

**Send documentation comments to [dcnm-san-docfeedback@cisco.com](mailto:dcnm-san-docfeedback@cisco.com)**

```

        e.printStackTrace();
    }
}

/**
 * This method is to check SanWSService is available or not. If available
 * bind to the stub.
 */
public static void getZoneManagerService() {
    try {

        String zmServiceAddr = "http://" + url
            + "/ZoneManagerWSService/ZoneManagerWS";
        ZoneManagerServiceLocator zmService = new ZoneManagerServiceLocator();
        zmService.setEndpointAddress("ZoneManagerPort", zmServiceAddr);
        zoneManager = zmService.getZoneManagerPort();

        if (zoneManager == null) {
            System.out.println("zonemanager service not available");
            return;
        }
        SOAPHeaderElement hdrElement = setSoapHeader(token);
        ZoneManagerBindingStub zmStub = (ZoneManagerBindingStub) zoneManager;
        zmStub.setHeader(hdrElement);
    } catch (Exception e) {
        e.printStackTrace();
    }
}

/**
 * This method is to check ZoneManagerWSService is available or not. If
 * available bind to the stub.
 */
public static void getSanService() {
    try {

        String sanServiceAddr = "http://" + url + "/SanWSService/SanWS";
        SanServiceLocator sanService = new SanServiceLocator();
        sanService.setEndpointAddress("SanPort", sanServiceAddr);
        san = sanService.getSanPort();
        if (san == null) {
            System.out.println("service not available");
            return;
        }
        SOAPHeaderElement hdrElement = setSoapHeader(token);
        SanBindingStub sStub = (SanBindingStub) san;
        sStub.setHeader(hdrElement);
    } catch (Exception e) {
        e.printStackTrace();
    }
}

private static SOAPHeaderElement setSoapHeader(String token) {

    SOAPHeaderElement hdrElement = new SOAPHeaderElement(
        "http://ep.cisco.dcbu.cisco.com", "token");

    hdrElement.setPrefix("m");

    hdrElement.setMustUnderstand(false);

    hdrElement.setValue(token);
    return hdrElement;
}

```

***Send documentation comments to [dcnm-san-docfeedback@cisco.com](mailto:dcnm-san-docfeedback@cisco.com)***

```
static void exception(Exception e){
    header("ZonwWsWclient: Some exception occurred!! \n"+e.toString()+"\nEnter [y] to
view the stack trace or any key to continue...");
    try {
        String view=input_.readLine();
        if(view.equalsIgnoreCase("y")){
            e.printStackTrace();

        }
    } catch (IOException e1) {
        exception(e);
    }
}
}
```