Exhibit J

Comparison of Java (J2SE 1.5) and Android versions of PolicyNodeImpl.java

PolicyNodeImpl.java (Java version)
[comments removed and spacing adjusted for comparison]

```
final class PolicyNodeImpl implements PolicyNode {

    private static final String ANY_POLICY = "2.5.29.32.0";
    private PolicyNodeImpl mParent;
    private HashSet mChildren;
    private String mValidPolicy;
    private HashSet mQualifierSet;
    private boolean mCriticalityIndicator;
    private HashSet mExpectedPolicySet;
    private boolean mOriginalExpectedPolicySet;
    private int mDepth;
    private boolean isImmutable = false;

    PolicyNodeImpl(PolicyNodeImpl parent, String validPolicy,
        Set qualifierSet,boolean criticalityIndicator,
        Set expectedPolicySet,boolean generatedByPolicyMapping) {

        mParent = parent;
        mChildren = new HashSet();
        if (validPolicy != null)
            mValidPolicy = validPolicy;
        else
            mValidPolicy = "";

        if (qualifierSet != null)
            mQualifierSet = new HashSet(qualifierSet);
        else
            mQualifierSet = new HashSet();

        mCriticalityIndicator = criticalityIndicator;
        if (expectedPolicySet != null)
            mExpectedPolicySet = new HashSet(expectedPolicySet);
        else
            mExpectedPolicySet = new HashSet();

        mOriginalExpectedPolicySet = !generatedByPolicyMapping;
        if (mParent != null) {
            mDepth = mParent.getDepth() + 1;
            mParent.addChild(this);
        } else {
            mDepth = 0;
        }
    }
```

PolicyNodeImpl.java (Android version)
[spacing adjusted for comparison]

```
public class PolicyNodeImpl implements PolicyNode {

    private static final String ANY_POLICY = "2.5.29.32.0";
    private PolicyNodeImpl mParent;
    private HashSet mChildren;
    private String mValidPolicy;
    private HashSet mQualifierSet;
    private boolean mCriticalityIndicator;
    private HashSet mExpectedPolicySet;
    private boolean mOriginalExpectedPolicySet;
    private int mDepth;
    private boolean isImmutable;

    public PolicyNodeImpl(PolicyNodeImpl policynodeimpl, String s,
        Set set, boolean flag,
        Set set1, boolean flag1) {
        isImmutable = false;
        mParent = policynodeimpl;
        mChildren = new HashSet();
        if(s != null) {
            mValidPolicy = s;
        } else {
            mValidPolicy = "";
        }
        if(set != null) {
            mQualifierSet = new HashSet(set);
        } else {
            mQualifierSet = new HashSet();
        }
        mCriticalityIndicator = flag;
        if(set1 != null) {
            mExpectedPolicySet = new HashSet(set1);
        } else {
            mExpectedPolicySet = new HashSet();
        }
        mOriginalExpectedPolicySet = !flag1;
        if(mParent != null) {
            mDepth = mParent.getDepth() + 1;
            mParent.addChild(this);
        } else {
            mDepth = 0;
        }
    }
```

Comparison of Java (J2SE 1.5) and Android versions of PolicyNodeImpl.java

PolicyNodeImpl.java (Java version) [comments removed and spacing adjusted for comparison]	PolicyNodeImpl.java (Android version) [spacing adjusted for comparison]

`PolicyNodeImpl(PolicyNodeImpl parent,`
`PolicyNodeImpl node) {`
`this(parent, node.mValidPolicy, node.mQualifierSet,`
`node.mCriticalityIndicator, node.mExpectedPolicySet, false);`
`}`
 	
`public PolicyNode getParent() {`
`return mParent;`
`}`
 	
`public Iterator<PolicyNodeImpl> getChildren() {`
`return Collections.unmodifiableSet(mChildren).iterator();`
`}`
 	
`public int getDepth() {`
`return mDepth;`
`}`
 	
`public String getValidPolicy() {`
`return mValidPolicy;`
`}`
 	
`public Set<PolicyQualifierInfo> getPolicyQualifiers() {`
`return Collections.unmodifiableSet(mQualifierSet);`
`}`
 	
`public Set<String> getExpectedPolicies() {`
`return Collections.unmodifiableSet(mExpectedPolicySet);`
`}`
 	
`public boolean isCritical() {`
`return mCriticalityIndicator;`
`}`
 	
`public String toString() {`
`StringBuffer buffer = new StringBuffer(this.asString());`
 	
`Iterator it = getChildren();`
`while (it.hasNext()) {`
`buffer.append((PolicyNodeImpl)it.next());`
`}`
`return buffer.toString();`
`}`
	`PolicyNodeImpl(PolicyNodeImpl policynodeimpl,`
	`PolicyNodeImpl policynodeimpl1) {`
	`this(policynodeimpl, policynodeimpl1.mValidPolicy, ((Set)`
	`(policynodeimpl1.mQualifierSet)), policynodeimpl1.mCriticalityIndicator,`
	`((Set) (policynodeimpl1.mExpectedPolicySet)), false);`
	`}`
	
	`public PolicyNode getParent() {`
	`return mParent;`
	`}`
	
	`public Iterator getChildren() {`
	`return Collections.unmodifiableSet(mChildren).iterator();`
	`}`
	
	`public int getDepth() {`
	`return mDepth;`
	`}`
	
	`public String getValidPolicy() {`
	`return mValidPolicy;`
	`}`
	
	`public Set getPolicyQualifiers() {`
	`return Collections.unmodifiableSet(mQualifierSet);`
	`}`
	
	`public Set getExpectedPolicies() {`
	`return Collections.unmodifiableSet(mExpectedPolicySet);`
	`}`
	
	`public boolean isCritical() {`
	`return mCriticalityIndicator;`
	`}`
	
	`public String toString() {`
	`StringBuffer stringbuffer = new StringBuffer(asString());`
	
	`for(Iterator iterator = getChildren(); iterator.hasNext();`
	`stringbuffer.append((PolicyNodeImpl)iterator.next()));`
	
	`return stringbuffer.toString();`
	`}`

Comparison of Java (J2SE 1.5) and Android versions of PolicyNodeImpl.java

PolicyNodeImpl.java (Java version)
[comments removed and spacing adjusted for comparison]

```
boolean isImmutable() {
        return isImmutable;
    }

    void setImmutable() {
        if (isImmutable)
            return;
        Iterator it = mChildren.iterator();
        while (it.hasNext()) {
            PolicyNodeImpl node = (PolicyNodeImpl) it.next();
            node.setImmutable();
        }
        isImmutable = true;
    }

    private void addChild(PolicyNodeImpl child) {
        if (isImmutable) {
            throw new IllegalStateException("PolicyNode is immutable");
        }
        mChildren.add(child);
    }



   void addExpectedPolicy(String expectedPolicy) {
        if (isImmutable) {
            throw new IllegalStateException("PolicyNode is immutable");
        }
        if (mOriginalExpectedPolicySet) {
            mExpectedPolicySet.clear();
            mOriginalExpectedPolicySet = false;
        }
        mExpectedPolicySet.add(expectedPolicy);
    }

    void prune(int depth) {
        if (isImmutable)
            throw new IllegalStateException("PolicyNode is immutable");
        if (mChildren.size() == 0)
            return;
        Iterator it = mChildren.iterator();
        while (it.hasNext()) {
            PolicyNodeImpl node = (PolicyNodeImpl) it.next();
            node.prune(depth);
            if ((node.mChildren.size() == 0) && (depth > mDepth + 1))
                it.remove();
        }
    }
```

PolicyNodeImpl.java (Android version)
[spacing adjusted for comparison]

```
    boolean isImmutable() {
        return isImmutable;
    }

    void setImmutable() {
        if(isImmutable)  return;
        PolicyNodeImpl policynodeimpl;
        for(Iterator iterator = mChildren.iterator(); iterator.hasNext();
policynodeimpl.setImmutable())
            policynodeimpl = (PolicyNodeImpl)iterator.next();

        isImmutable = true;
    }


    private void addChild(PolicyNodeImpl policynodeimpl) {
        if(isImmutable) {
            throw new IllegalStateException("PolicyNode is immutable");
        } else {
            mChildren.add(policynodeimpl);
            return;
        }
    }

    void addExpectedPolicy(String s) {
        if(isImmutable)
            throw new IllegalStateException("PolicyNode is immutable");

        if(mOriginalExpectedPolicySet) {
            mExpectedPolicySet.clear();
            mOriginalExpectedPolicySet = false;
        }
        mExpectedPolicySet.add(s);
    }

    void prune(int i) {
        if(isImmutable)
            throw new IllegalStateException("PolicyNode is immutable");
        if(mChildren.size() == 0)
            return;
        Iterator iterator = mChildren.iterator();
        do {
            if(!iterator.hasNext())  break;
            PolicyNodeImpl policynodeimpl = (PolicyNodeImpl)iterator.next();
            policynodeimpl.prune(i);
            if(policynodeimpl.mChildren.size() == 0 && i > mDepth + 1)
                iterator.remove();
        } while(true);
    }
```

Comparison of Java (J2SE 1.5) and Android versions of PolicyNodeImpl.java

PolicyNodeImpl.java (Java version)
[comments removed and spacing adjusted for comparison]

```
void deleteChild(PolicyNode childNode) {
    if (isImmutable) {
        throw new IllegalStateException("PolicyNode is immutable");
    }
    mChildren.remove(childNode);
}



PolicyNodeImpl copyTree() {
    return copyTree(null);
}

private PolicyNodeImpl copyTree(PolicyNodeImpl parent) {
    PolicyNodeImpl newNode = new PolicyNodeImpl(parent, this);

    Iterator it = mChildren.iterator();
    while (it.hasNext()) {
        PolicyNodeImpl node = (PolicyNodeImpl) it.next();
        node.copyTree(newNode);
    }

    return newNode;
}

Set getPolicyNodes(int depth) {
    Set set = new HashSet();
    getPolicyNodes(depth, set);
    return set;
}

private void getPolicyNodes(int depth, Set set) {
    if (mDepth == depth) {
        set.add(this);
    } else {
        Iterator it = mChildren.iterator();
        while (it.hasNext()) {
            PolicyNodeImpl node = (PolicyNodeImpl) it.next();
           node.getPolicyNodes(depth, set);
        }
    }
}
```

PolicyNodeImpl.java (Android version)
[spacing adjusted for comparison]

```
    void deleteChild(PolicyNode policynode) {
        if(isImmutable) {
            throw new IllegalStateException("PolicyNode is immutable");
        } else {
            mChildren.remove(policynode);
            return;
        }
    }

    PolicyNodeImpl copyTree() {
        return copyTree(null);
    }

    private PolicyNodeImpl copyTree(PolicyNodeImpl policynodeimpl) {
        PolicyNodeImpl policynodeimpl1 = new PolicyNodeImpl(policynodeimpl,
this);
        PolicyNodeImpl policynodeimpl2;
        for(Iterator iterator = mChildren.iterator(); iterator.hasNext();
policynodeimpl2.copyTree(policynodeimpl1))
            policynodeimpl2 = (PolicyNodeImpl)iterator.next();

        return policynodeimpl1;
    }


    Set getPolicyNodes(int i) {
        HashSet hashset = new HashSet();
        getPolicyNodes(i, ((Set) (hashset)));
        return hashset;
    }

    private void getPolicyNodes(int i, Set set) {
        if(mDepth == i) {
            set.add(this);
        } else {
            PolicyNodeImpl policynodeimpl;
            for(Iterator iterator = mChildren.iterator();
iterator.hasNext(); policynodeimpl.getPolicyNodes(i, set))
                policynodeimpl = (PolicyNodeImpl)iterator.next();
        }
    }
```

Comparison of Java (J2SE 1.5) and Android versions of PolicyNodeImpl.java

PolicyNodeImpl.java (Java version)
[comments removed and spacing adjusted for comparison]

```
    Set getPolicyNodesExpected(int depth, String expectedOID, boolean
matchAny)
    {
        if (expectedOID.equals(ANY_POLICY)) {
            return getPolicyNodes(depth);
        } else {
            return getPolicyNodesExpectedHelper(depth, expectedOID,
matchAny);
        }
    }

    private Set getPolicyNodesExpectedHelper(int depth, String expectedOID,
        boolean matchAny)
    {
        HashSet set = new HashSet();

        if (mDepth < depth) {
            Iterator it = mChildren.iterator();
            while (it.hasNext()) {
                PolicyNodeImpl node = (PolicyNodeImpl) it.next();
                set.addAll(node.getPolicyNodesExpectedHelper(depth,
                                                             expectedOID,
                                                             matchAny));
            }
        } else {
            if (matchAny) {
                if (mExpectedPolicySet.contains(ANY_POLICY))
                    set.add(this);
            } else {
                if (mExpectedPolicySet.contains(expectedOID))
                    set.add(this);
            }
        }

        return set;
    }

    Set getPolicyNodesValid(int depth, String validOID) {
        HashSet set = new HashSet();
        if (mDepth < depth) {
            Iterator it = mChildren.iterator();
            while (it.hasNext()) {
                PolicyNodeImpl node = (PolicyNodeImpl) it.next();
                set.addAll(node.getPolicyNodesValid(depth, validOID));
            }
        } else {
            if (mValidPolicy.equals(validOID))
                set.add(this);
        }
        return set;
    }
```

PolicyNodeImpl.java (Android version)
[spacing adjusted for comparison]

```
    Set getPolicyNodesExpected(int i, String s, boolean flag) {


        if(s.equals("2.5.29.32.0"))
            return getPolicyNodes(i);
        else
            return getPolicyNodesExpectedHelper(i, s, flag);
    }



    private Set getPolicyNodesExpectedHelper(int i, String s, boolean flag)

{
        HashSet hashset = new HashSet();

        if(mDepth < i) {
            PolicyNodeImpl policynodeimpl;
            for(Iterator iterator = mChildren.iterator();
iterator.hasNext();
hashset.addAll(policynodeimpl.getPolicyNodesExpectedHelper(i, s, flag)))
                policynodeimpl = (PolicyNodeImpl)iterator.next();


        } else if(flag) {

            if(mExpectedPolicySet.contains("2.5.29.32.0"))
                hashset.add(this);
        } else if(mExpectedPolicySet.contains(s)) {

            hashset.add(this);
        }


        return hashset;
    }

    Set getPolicyNodesValid(int i, String s) {
        HashSet hashset = new HashSet();
        if(mDepth < i) {
            PolicyNodeImpl policynodeimpl;
            for(Iterator iterator = mChildren.iterator();
iterator.hasNext(); hashset.addAll(policynodeimpl.getPolicyNodesValid(i,
s)))
                policynodeimpl = (PolicyNodeImpl)iterator.next();

        } else if(mValidPolicy.equals(s)) {
            hashset.add(this);
        }
        return hashset;
    }
```

Comparison of Java (J2SE 1.5) and Android versions of PolicyNodeImpl.java

PolicyNodeImpl.java (Java version)
[comments removed and spacing adjusted for comparison]

```
    private static String policyToString(String oid) {
        if (oid.equals(ANY_POLICY)) {
            return "anyPolicy";
        } else {
            return oid;
        }
    }

    String asString() {
        if (mParent == null) {
            return "anyPolicy  ROOT\n";
        } else {
            StringBuffer sb = new StringBuffer();
            for (int i = 0, n = getDepth(); i < n; i++) {
                sb.append("  ");
            }
            sb.append(policyToString(getValidPolicy()));
            sb.append("  CRIT: ");
            sb.append(isCritical());
            sb.append("  EP: ");
            for (Iterator t = getExpectedPolicies().iterator(); t.hasNext();
) {
                String policy = (String)t.next();
                sb.append(policyToString(policy));
                sb.append(" ");
            }
            sb.append(" (");
            sb.append(getDepth());
            sb.append(")\n");
            return sb.toString();
        }
    }
}
```

PolicyNodeImpl.java (Android version)
[spacing adjusted for comparison]

```
    private static String policyToString(String s) {
        if(s.equals("2.5.29.32.0")) {
            return "anyPolicy";
        } else {
            return s;
        }
    }

    String asString() {
        if(mParent == null)
            return "anyPolicy  ROOT\n";

        StringBuffer stringbuffer = new StringBuffer();
        int i = 0;
        for(int j = getDepth(); i < j; i++)
            stringbuffer.append("  ");
        stringbuffer.append(policyToString(getValidPolicy()));
        stringbuffer.append("  CRIT: ");
        stringbuffer.append(isCritical());
        stringbuffer.append("  EP: ");
        for(Iterator iterator = getExpectedPolicies().iterator();
iterator.hasNext(); stringbuffer.append(" ")) {

            String s = (String)iterator.next();
            stringbuffer.append(policyToString(s));
        }
        stringbuffer.append(" (");
        stringbuffer.append(getDepth());
        stringbuffer.append(")\n");
        return stringbuffer.toString();
    }
}
```