Chapter 1

A Corrected Code Version

In this appendix, a possible corrected version of the code is provided for only the assigned methods. Three important clarifications follow. First of all, the number of the lines is not the same for two main reasons: first, here the numeration starts from 1 at the first assigned method (without considering all previous code); second, the numeration cannot be the same due to we have modified several lines of codes.
Then, we have added a new method to implement the logger. In fact, the code used to log some events is often duplicated in the assigned methods with a few differences. Hence, a better way to write that code is to define a new private method that includes the duplicated lines of code. In this way the code is more clear and readable and a change can be perform speedily and easily by a change on the method.
Finally, to make the following code more *nice for the eyes*, the space special characters are not shown, but obviously all tabs have been replaced with four spaces.

```
1     private Subject getSubjectFromSecurityCurrent()
2             throws SecurityMechanismException {
3         com.sun.enterprise.security.SecurityContext securityContext;
4         securityContext = com.sun.enterprise.security.SecurityContext.getCurrent();
5         if(securityContext == null) {
6             fineLevelLog(" SETTING GUEST ---");
7             securityContext = com.sun.enterprise.security.SecurityContext.init();
8         }
9         if(securityContext == null) {
10            throw new SecurityMechanismException("Could not find " +
```

```
                                                "security information");
        }
        Subject subject = securityContext.getSubject();
        if(subject == null) {
            throw new SecurityMechanismException("Could not find " +
                                                "subject information in the " +
                                                "security context.");
        }
        fineLevelLog("Subject in security current:" + subject);
        return subject;
    }

    public CompoundSecMech selectSecurityMechanism(IOR ior)
            throws SecurityMechanismException {
        CompoundSecMech[] mechList = getCtc().getSecurityMechanisms(ior);
        CompoundSecMech mech = selectSecurityMechanism(mechList);
        return mech;
    }

    /**
     * Select the security mechanism from the list of compound security
     * mechanisms.
     */
    private CompoundSecMech selectSecurityMechanism(CompoundSecMech[] mechList)
                                throws SecurityMechanismException {
        // We should choose from list of compound security mechanisms
        // which are in decreasing preference order. Right now we select
        // the first one.
        if(mechList == null || mechList.length == 0) {
            return null;
        }
        CompoundSecMech mech;
        for(int i = 0; i < mechList.length; i++) {
            mech = mechList[i];
            if( useMechanism(mech) ) {
                return mech;
            }
        }
        throw new SecurityMechanismException("Cannot use any of the " +
                                             "target's supported mechanisms");
    }

    private boolean useMechanism(CompoundSecMech mech) {
        TLS_SEC_TRANS tls = getCtc().getSSLInformation(mech);

        if ( (mech.sas_context_mech.supported_naming_mechanisms.length > 0 &&
              !isMechanismSupported(mech.sas_context_mech)) ||
              (mech.as_context_mech.client_authentication_mech.length > 0 &&
              !isMechanismSupportedAS(mech.as_context_mech))) {
```

```
60              return false;
61          }
62
63          if(tls == null) {
64              return true;
65          }
66          int targetRequires = tls.target_requires;
67          return ! (isSet(targetRequires, EstablishTrustInClient.value) && ! sslUtils.
                isKeyAvailable());
68      }
69
70      private boolean evaluateClientConformanceSsl(
71                          EjbIORConfigurationDescriptor iordesc,
72                          boolean  sslUsed,
73                          X509Certificate[] certchain) {
74
75          boolean sslRequired = false;
76          boolean sslSupported = false;
77          int sslTargetRequires = 0;
78          int sslTargetSupports = 0;
79
80          try {
81              fineLevelLog("SecurityMechanismSelector.evaluate_client_" +
82                           "conformance_ssl->:");
83
84              /*****************************************************************
85               * Conformance Matrix:
86               *
87               * |---------------|-----------------|-----------------|-----------|
88               * | SSLClientAuth | targetrequires. | targetSupports. | Conformant|
89               * |               |       ETIC      |      ETIC       |           |
90               * |---------------|-----------------|-----------------|-----------|
91               * |     Yes       |          0      |      1          |    Yes    |
92               * |     Yes       |          0      |      0          |    No     |
93               * |     Yes       |          1      |      X          |    Yes    |
94               * |     No        |          0      |      X          |    Yes    |
95               * |     No        |          1      |      X          |    No     |
96               * |---------------|-----------------|-----------------|-----------|
97               *
98               *****************************************************************/
99
100             // gather the configured SSL security policies.
101
102             sslTargetRequires = this.getCtc().getTargetRequires(iordesc);
103             sslTargetSupports = this.getCtc().getTargetSupports(iordesc);
104
105             sslRequired = (isSet(sslTargetRequires, Integrity.value) ||
106                            isSet(sslTargetRequires, Confidentiality.value) ||
107                            isSet(sslTargetRequires,EstablishTrustInClient.value));
```

```
108
109             sslSupported = ( sslTargetSupports != 0);
110
111             /* Check for conformance for using SSL usage.
112              *
113              * a. if SSL was used, then either the target must require or
114              *    support SSL. In the latter case, SSL is used because of client
115              *    policy.
116              * b. if SSL was not used, then the target must not require it
117              *    either. The target may or may not support SSL (it is
118              *    irrelevant).
119              */
120             fineLevelLog("SecurityMechanismSelector.evaluate_client_" +
121                          "conformance_ssl:" +
122                          " " + isSet(sslTargetRequires, Integrity.value) +
123                          " " + isSet(sslTargetRequires, Confidentiality.value) +
124                          " " +
125                          isSet(sslTargetRequires,EstablishTrustInClient.value) +
126                          " " + sslRequired +
127                          " " + sslSupported +
128                          " " + sslUsed);
129
130             if ((sslUsed && !(sslRequired || sslSupported)) || sslRequired) {
131                 return false;
132             }
133
134             /* Check for conformance for SSL client authentication.
135              *
136              * a. if client performed SSL client authentication, then the target
137              *    must either require or support SSL client authentication. If
138              *    the target only supports, SSL client authentication is used
139              *    because of client security policy.
140              *
141              * b. if SSL client authentication was not used, then the target must
142              *    not require SSL client authentcation either. The target may or may
143              *    not support SSL client authentication (it is irrelevant).
144              */
145
146             fineLevelLog("SecurityMechanismSelector.evaluate_client_" +
147                          "conformance_ssl:" +
148                          " " +
149                          isSet(sslTargetRequires,EstablishTrustInClient.value) +
150                          " " +
151                          isSet(sslTargetSupports,EstablishTrustInClient.value));
152
153             if ((certchain != null &&
154                  !(isSet(sslTargetRequires, EstablishTrustInClient.value) ||
155                  isSet(sslTargetSupports, EstablishTrustInClient.value))) ||
156                 (isSet(sslTargetRequires, EstablishTrustInClient.value))) {
```

```
157                     return false; // security mechanism did not match
158             }
159
160             fineLevelLog("SecurityMechanismSelector.evaluate_client_" +
161                             "conformance_ssl: true");
162
163             return true ; // mechanism matched
164         } finally {
165             fineLevelLog("SecurityMechanismSelector.evaluate_client_" +
166                             "conformance_ssl<-:");
167         }
168     }
169
170     //At the end of the class or into a specific class dedicated to the logger
171     private fineLevelLog (String s) {
172         if(_logger.isLoggable(Level.FINE)) {
173             _logger.log(Level.FINE, s);
174         }
175     }
```

Listing 1.1: "A corrected version of the code."