Chapter 1

Assignment

In this chapter we will present the class that has been assigned to us. First, the lines of code to be analysed are presented without any comment. Afterwards, a brief description of the class role and function is presented. Obviously this is made by us and it is based only on the code and on the documentation provided by the authors.

```
                getCorbaIORDescSet().add(iorDesc2);
            }
            getCorbaIORDescSet().add(iorDesc);

        } catch(Exception e) {
            _logger.log(Level.SEVERE,"iiop.Exception",e);
        }
    }

    public ConnectionContext getClientConnectionContext() {
        Hashtable h = ConnectionExecutionContext.getContext();
        ConnectionContext scc =
            (ConnectionContext) h.get(CLIENT_CONNECTION_CONTEXT);
        return scc;
    }

    public void setClientConnectionContext(ConnectionContext scc) {
        Hashtable h = ConnectionExecutionContext.getContext();
        h.put(CLIENT_CONNECTION_CONTEXT, scc);
    }

    /**
     * This method determines if SSL should be used to connect to the
     * target based on client and target policies. It will return null if
```

```
     * SSL should not be used or an SocketInfo containing the SSL port
     * if SSL should be used.
     */
    public SocketInfo getSSLPort(IOR ior, ConnectionContext ctx)
    {
        SocketInfo info = null;
        CompoundSecMech mechanism = null;
        try {
             mechanism = selectSecurityMechanism(ior);
        } catch(SecurityMechanismException sme) {
            throw new RuntimeException(sme.getMessage());
        }
        ctx.setIOR(ior);
        ctx.setMechanism(mechanism);

        TLS_SEC_TRANS ssl = null;
        if ( mechanism != null ) {
            ssl = getCtc().getSSLInformation(mechanism);
        }

        if (ssl == null) {
            if (isSslRequired()) {
                // Attempt to create SSL connection to host, ORBInitialPort
                IIOPProfileTemplate templ = (IIOPProfileTemplate)
                    ior.getProfile().getTaggedProfileTemplate();
                IIOPAddress addr = templ.getPrimaryAddress();
                info = IORToSocketInfoImpl.createSocketInfo(
                        "SecurityMechanismSelector1",
                        "SSL", addr.getHost(), orbHelper.getORBPort(orbHelper.getORB())
                            );
                return info;
            } else {
                return null;
            }
        }

        int targetRequires = ssl.target_requires;
        int targetSupports = ssl.target_supports;

        /*
         * If target requires any of Integrity, Confidentiality or
         * EstablishTrustInClient, then SSL is used.
         */
        if (isSet(targetRequires, Integrity.value) ||
                isSet(targetRequires, Confidentiality.value) ||
                isSet(targetRequires, EstablishTrustInClient.value)) {
            if (_logger.isLoggable(Level.FINE)) {
                 _logger.log(Level.FINE, "Target␣requires␣SSL");
            }
```

```
            ctx.setSSLUsed(true);
            String type = "SSL";
            if(isSet(targetRequires, EstablishTrustInClient.value)) {
                type = "SSL_MUTUALAUTH";
                ctx.setSSLClientAuthenticationOccurred(true);
            }
            short sslport = ssl.addresses[0].port;
            int ssl_port = Utility.shortToInt(sslport);
            String host_name = ssl.addresses[0].host_name;

            info = IORToSocketInfoImpl.createSocketInfo(
                    "SecurityMechanismSelector2",
                    type, host_name, ssl_port);

            return info;
        } else if (isSet(targetSupports, Integrity.value) ||
                    isSet(targetSupports, Confidentiality.value) ||
                    isSet(targetSupports, EstablishTrustInClient.value)) {
            if (_logger.isLoggable(Level.FINE)) {
                _logger.log(Level.FINE, "Target␣supports␣SSL");
            }

            if ( isSslRequired() ) {
                if (_logger.isLoggable(Level.FINE)) {
                    _logger.log(Level.FINE, "Client␣is␣configured␣to␣require␣SSL␣for␣
                        the␣target");
                }

                ctx.setSSLUsed(true);
                short sslport = ssl.addresses[0].port;
```