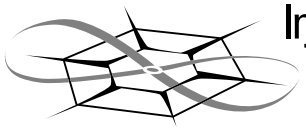
The University of Kansas



Information and Telecommunication Technology Center

Technical Report

ACE Connection Interface Specification Version 0.9

James Mauro, Leon Searl, and Gary Minden

ITTC-FY2001-TR-23150-07

January 2001

Project Sponsor:
U.S. Air Force and the Defense Advanced Research Projects Agency under contract no. F30602-00-2-0581
and The National Science Foundation
under grant EIA-9972843

1 Introduction

The ACE Connection interface contains a set of Java methods and C functions needed for communications between services within the ACE Environment. These services communicate through two separate types of channels. The first channel, the Command Channel, is low bandwidth, reliable and bi-directional. The other channel, the Data Channel, is high bandwidth, unreliable, and unidirectional. These two channels comprise all communications between two services with the ACE Environment.

2 Specifications

2.1 Problem Domain

All services within the ACE Domain talk to each other via a network. The services have been implemented in such a way that the implementation for the underlying network is located in one place. This allows for the underlying network to be changed at any time. The services within the ACE Domain communicate on two different types of network connections, Command connections and Data connections. Command connections are bi-directional, error and ordering correcting. They are similar to TCP connections. The Data connections are unidirectional and do not correct for errors or lost packets. They are similar to UDP packets. Both sets of connections are generated from a Connection Factory. This factory handles setup of the connections and returns an "established" connection.

2.2 Connection Factory

Java:

```
public class ACEConnectionFactory
{
      public ACEConnectionFactory( boolean EncryptionOn )
      public ACEConnectionFactory( )
      public boolean isEncrypted()
      public ACECommandConnection getCommandConnection(String
            ACEAddress)
      public ACEServerCommandConnection getACEServerCommandConnection()
      public ACEDataRecieveConnection getDataRecieveConnection(String
            ACEAddress)
      public ACEDataSendConnection getDataSendConnection(String
            ACEAddress)

}
```

C functions:

```
int GetACEDataRecieveConnection( const char * address,
                                 ACEDataConnection ** connection )
int GetACEDataSendConnection( const char * address, ACEDataConnection
                                 ** connection )
```

2.2.1 Query Interface Specifications

S-2.2.1.1 Method isEncrypted()

The Connection factory provides a method to tell if it is currently creating encrypted connections or if the connections are in the clear. The isEncrypted method specifies the system. Encryption is turned on or off by the constructor. By default, encryption is turned on. The encryption state cannot be changed after it has started. This method is for debugging only.

```
public boolean isEncrypted()
```

Arguments:

None

Exceptions:

None

Returns:

true if the connection is encrypted on creation, false otherwise.

2.2.2 Behavioral Rule Interface Specification

None

2.2.3 Mapping Interface Specification

None

2.2.4 Operation Specification

S-2.2.4.1 ACEConnectionFactory Constructor

A new ACE Connection factory can be created with the constructor. The constructor takes one argument, a Boolean value to tell if encryption is to be used. A second constructor, without an argument is used to create a default object. The default constructor is the same as calling the first connection with true passed to it.

```
public ACEConnectionFactory( boolean useEncryption )

public ACEConnectionFactory()
```

S-2.2.4.2 getCommandConnection

A Command connection can be requested with the getCommandConnection method. This method returns an ACECommandConnection object which can be used to communicate with another ACE Service. The ACECommandConnection must be initialized before the return to the caller.

```
public ACECommandConnection getCommandConnection( String ACEAddress )
```

Arguments:

ACEAddress - a string representation that specifies where to connect.

Exceptions:

IOException, PermissionsException, ConnectException

Returns:

A connected ACECommandConnection to the address specified.

S-2.2.4.3 getDataRecieveConnection

A Data connection can be requested with the getDataRecieveConnection method. This method returns an ACEDataConnection object, which can be used to communicate with another ACEService. The ACEDataConnection is connectionless.

Java:

```
public ACEDataRecieveConnection getDataRecieveConnection(String
ACEAddress)
```

Arguments:

ACEAddress – a string representation that specifies from what service to listen to

the packets from.

Exceptions:

IOException, PermissionsException

Returns:

An ACEDataRecieveConnection that sends packets to the specified port.

C:

```
int GetACEDataRecieveConnection( const char * address, ACEDataConnection
                                 ** connection )
```

Arguments:

address – The ACE Address to receive packets from.

connection – A pointer to an object that contain a new connection object. The function initializes the connection object. The connection object is initialized to only receive packets.

Returns:

ACECONNECTION_INVALID_ADDRESS – An invalid address was requested

ACECONNECTION_PERMISSION_DENIED – Permission was denied to send the packet.

ACECONNECTION_SUCCESS – The operation succeeded

S-2.2.4.4 getDataSendConnection

A Data connection can be requested with the getDataSendConnection method. This method returns an ACEDataSendConnection object, which can be used to communicate with another ACEService. The ACEDataSendConnection is connectionless.

Java:

```
public ACEDataSendConnection getDataSendConnection(String ACEAddress)
```

Arguments:

ACEAddress – a string representation that specifies where to send the packets to.

Exceptions:

IOException, PermissionsException

Returns:

An ACEDataSendConnection that sends packets to the specified port.

C:

```
int GetACEDataSendConnection( const char * address, ACEDataConnection **
                                connection )
```

Arguments:

address – The ACE Address to send packets to.

connection – A pointer to an object contains a new connection. The function initializes the connection object. The connection object is initialized to only send data out.

Returns:

ACECONNECTION_PERMISSION_DENIED – Permission was denied to send the packet.

ACECONNECTION_SUCCESS – The operation succeeded

S-2.2.4.5 getACEServiceCommandConnection

A server socket style connection can be returned with getACEServerCommandConnection. This returns an ACEServerCommandConnection that can receive new ACECommandConnections that connect from outside sources.

```
public ACEServerCommandConnection getServerCommandConnection()
```

Arguments:

None

Exceptions:

IOException, PermissionsException, ConnectException

Returns: A connected ACEServerCommandConnection to the address specified.

2.3 Command Connections

Java:

```
public class ACECommandConnection
{
        public String getAddress()
        public void send( ACECmdLine command )
        public ACECmdLine receive( ACECmdTable MyTable )
        public ACECmdLine sendWithResponse( ACECmdLine command,
                          ACECmdTable MyTable)
}
```

2.3.1 Query Interface Specifications

S-2.3.1.1 getAddress

The connected address can be retrieved from the ACECommandConnection. The returned address is a string and it represents a valid ACEAddress.

```
public String getAddress()
```

Arguments:

None

Exceptions:

None

Returns:

A string value of the ACEAddress.

2.3.2 Behavioral Rule Interface Specification

None

2.3.3 Mapping Interface Specification

None

2.3.4 Operation Specification

S-2.3.4.1 Send

An ACECmdLine can be sent to another service over the ACECommandInterface. The command object is sent as a whole over the network. The send command blocks, until it is finished sending.

```
public void send( ACECmdLine command )
```

Arguments:

command – The command object to send over the network.

Exceptions:

ServiceUnavailableException

Returns:

None

S-2.3.4.2 Receive

An ACECmdLine can be read from the current ACE Command Connection. It returns a complete ACECmdLine. The receive method blocks until a complete command object has been received.

```
public ACECmdLine receive( ACECmdTable MyTable)
```

Arguments:

MyTable – An ACECmdTable that provides ACESemantics for parsing comamnds.

Exceptions:

ServiceUnavaibleException, ParserException

Returns:

A valid ACECmdLine that has been read from the input stream.

S-2.3.4.3 SendWithResponse

An ACECmdLine can be sent to another service, with another ACECmdLine as a response to the command. This method blocks until a response command has been received.

```
public ACECmdLine sendWithResponse( ACECmdLine command,
      ACECmdTable MyTable )
```

Arguments:

command - An ACECmdLine to send to the client.

MyTable – An ACECmdTable that provides ACESemantics for parsing commands

Exceptions:

ServiceUnavaibleException, ParserException

Returns:

A valid ACECmdLine that has been read from the input stream.

2.4 Server Command Connections

Java:

```
public class ACECommandConnection
{
      public ACECommandConnection accept( );
      public ACECommandConnection accept( String ACEAddress );
}
```

2.4.1 Query Interface Specifications

None

2.4.2 Behavioral Rule Interface Specification

None

2.4.3 Mapping Interface Specification

None

2.4.4 Operation Specification

S-2.4.4.1 Accept

The accept command returns a connected ACE Command Connection. The accept can receive connections from any source.

```
public ACECommandConnection accept()
```

Arguments:

None

Exceptions:

IOException

Returns:

A connected ACE Connection.

S-2.4.4.2 Accept with Address

The accept method that takes one argument specifies to only accept connections from one specific source. This source for the connection is specified the ACE Address arguments. Connections from any other services are denied.

```
public accept(String ACEAddress)
```

Arguments:

ACEAddress – The only source to accept incoming connections from

Exceptions:

IOException

Returns:

A connected ACE Connection.

2.5 Data Connections

Java:

```
public class ACEDataConnection
{
        public String getRemoteAddress()
        public void setCryptKey( String Key )
        public String  getLocalAddress()
}

public class ACEDataSendConnection extends ACEDataConnection
{
        public void send( byte[] packet )
}

public class ACEDataRecieveConnection extends ACEDataConnection
{
        public byte[] receive( )
}
```

C Functions:

```
        int GetACEDataRemoteAddress( ACEDataConnection * connection, char
              ** address)
        int GetACEDataLocalAddress( ACEDataConnection * connection, char
              ** address)
        int ACEDataConnectionSend( ACEDataConnection * connection, char *
              data, int size, int * sizeSent )
        int ACEDataConnectionRecieve( ACEDataConnection * connection,
              char * data, int size, int * sizeSent )
        int ACEDataConnectionSetCryptKey( ACEDataConnection * connection,
              char * key, int size )
        int ACEDataConnectionClose( ACEDataConnection * connection )
```

2.5.1 Query Interface Specifications

S-2.5.1.1 getRemoveAddress and GetACEDataRemoteAddress

The specified address that the ACEDataSendConnections sends packets to and the ACEDataRecieveConnection receives packets from can be retreived with the getRemoteAddress method.

Java:

```
public String getRemoteAddress()
```

Arguments:

None

Exceptions:

None

Returns:

A valid ACE address.

C:

```
int GetACEDataRemoteAddress( ACEDataConnection * connection, char **
                            address)
```

Arguments:

connection - A pointer to the Connection object.

address – A pointer to a string array. The memory for the array is created within the function. It must be freed after completion.

Returns:

ACECONNECTION_PERMISSION_DENIED – Permission was denied to send the packet.

ACECONNECTION_SUCCESS – The operation succeeded

S-2.5.1.2 getLocalAddress and GetDataLocalAddress

The local port that the socket uses to send or receive packets can be retrieved using the getLocalAddress() method.

Java:

```
public String  getLocalAddress()
```

Arguments:

None

Exceptions:

None

Returns:

A valid ACE address.

C:

```
int GetACEDataLocalAddress( ACEDataConnection * connection, char **
                                    address)
```

Arguments:

connection - A pointer to the Connection object.

address – A pointer to a string array. The memory for the array is created within the function. It must be freed after completion.

Returns:

ACECONNECTION_PERMISSION_DENIED – Permission was denied to send the packet.

ACECONNECTION_SUCCESS – The operation succeeded

2.5.2 Behavioral Rule Interface Specification

None

2.5.3 Mapping Interface Specification

None

2.5.4 Operation Specification

S-2.5.4.1 Send and ACEDataSendConnection

An ACEDataSendConnection can send an array of bytes to the other end using the send method or the ACEDataConnectionSend function. This functions blocks until the byte package has been sent.

Java:

```
public void send( byte[] packet )
```

Arguments:

packet – An array of bytes to be sent to the client.

Exceptions:

IOException

Returns:

None

C:

```
int ACEDataConnectionSend( ACEDataConnection * connection, char * data,
                    int size, int * sizeSent )
```

Arguments:

connection – the location to send the bytes to.

data – the array of bytes containing the data to be sent

size – size of the array of bytes.

sizeSent – number of bytes in the array that were actually sent.

Returns:

ACECONNECTION_PERMISSION_DENIED – Permission was denied to send the packet.

ACECONNECTION_SUCCESS – The operation succeeded

S-2.5.4.2 Receive and ACEDataReceiveConnection

4.5.2 – An ACEDataRecieveConnection can receive a number of bytes from an incoming connection. The receive method or the ACEDataConnectionRecieve method is used to perform this function. This function blocks until a datagram packet has been received.

Java:

```
public byte[] recieve( )
```

Arguments:

None

Exceptions:

IOException

Returns:

Any array of bytes from the last packet sent.

C:

```
int ACEDataConnectionRecieve( ACEDataConnection * connection, char *
                    data, int size, int * sizeSent )
```

Arguments:

connection – the location to send the bytes to.

data – the array of bytes to write the new data to.

size – size of the array of bytes.

sizeSent – number of bytes in the array that have been written into the array.

Returns:

ACECONNECTION_PERMISSION_DENIED – Permission was denied to send the packet.

ACECONNECTION_SUCCESS – The operation succeeded

S-2.5.4.3 setCryptKey

If the communications are encrypted, the communications between the data in and data out connections must share a symmetric key between them. The key exchange occurs with the help of the client and Command Connections. Both data in and data out connections share this method and this function.

Java:

```
public void setCryptKey( String Key )
```

Arguments:

Key – sets the key inside the connection.

Exceptions:

InvalidKeyException

Returns:

None

S-2.5.4.4 ACEDataConnectionClose

Close connection. The connection must be closed to free up memory and to release the socket. The function is only needed under C and C++, because Java performs the operation automatically when the finalize( ) method is called.

```
int ACEDataConnectionClose( ACEDataConnection * connection )
```

Arguments:

connection – the connection to close

Returns:

ACECONNECTION_PERMISSION_DENIED – Permission was denied to send the packet.

ACECONNECTION_SUCCESS – The operation succeeded

3 Design Constraints

None

4 Deprecated Specifications

5 Glossary

6 Change Log

Version	Date	Changes	Changes By
0.1	Jan 11, 2001	Initial Working Version	James Mauro
1.0	<date>	Initial Released Version	Initial Released Version

7 Notes

-