The University of Kansas

Information and Telecommunication Technology Center

Technical Report

Ambient Computing Environment: ACE Service Interface Specification

Leon Searl and Gary Minden

ITTC-FY2001-TR-23150-05

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

This document describes the API to the ACE Service object for both a service client and a service daemon. An ACE Service object is a class that represents a service provided by the ACE environment. A service is generally implemented as a daemon running on an ACE control server computer. The daemon communicates with other service daemons and applications to carry out a task. A daemon generally carries out one specialized task. The task may involve controlling a hardware device, processing a data stream, etc.

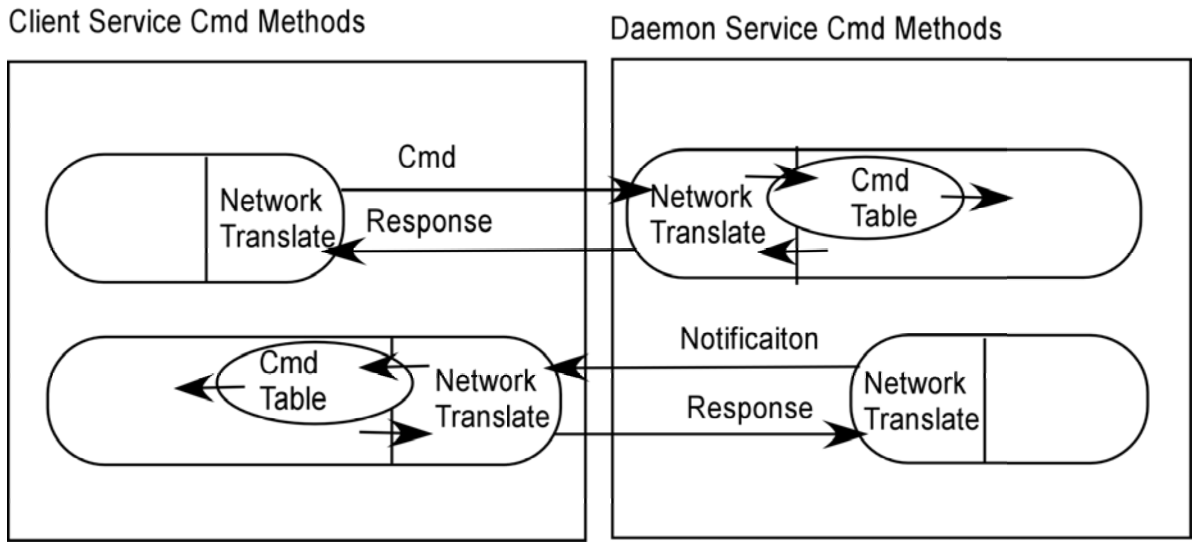
2 Specifications

The following specifications apply to the API of an ACE Service object.

2.1 *Problem Domain*

The ACE Service API is divided into two components, 1) Client side interface, 2) service daemon side interface. On the client side the client has an ACEService object that contains API methods that the client invokes to execute commands on the service. There is also a set of functions that the client must implement to process notifications from the service.

On the service daemon side there is a service object with a set of methods corresponding to the command set on the client side. A wrapper function is used to convert from the network format of the command to a method call. For daemon notifications sent to the client the daemon side of the service has a set of methods that send the notification to the client.



2.2 *Client Interface Specifications*

This section contains specifications that apply to the ACE Service client classes and methods. The Behavioral Rule Interface Specification section contains API specifications applying to methods of the service client class that change the behavior of the client. The

2.2.1 Query Interface Specifications

This section contains methods that the client uses to query the service daemon or the ACEService class. A summery of the methods is as follows:

class ACEService {

Array ACEServiceUserInfo getUsersList(void);
String getDataOutSocket();
String getDataInSocket();

}

S-2.2.1.1 Get Users List command: This command queries the service daemon to obtain the list of users and applications currently connected to the service daemon.

Java: class ACEService {

Array ACEServiceUserInfo getUsersList(void) }

Arguments: None

Return: An array of information of users currently connected to the service daemon. The class ACEServiceUserInfo has the following methods:

String getUserName();
String getApplicationName();
int getConnectedTimeSeconds();
boolean getWritePermissionP();

Exceptions: ServiceUnavailableException, CommandNotImplementedException, PermissionDeniedException

2.2.1.2 GetDataOutSocket Command: Gets the address for socket that data is transmitted out for from a daemon. This is the ACE address of the "UDP" socket. The socket may not be a UDP connection may not actually use UDP, but instead use a connection with similar characteristics, lack of error checking, no guaranteed ordering, no guaranteed delivery.

Java class ACEService: {

String getDataOutSocket() }

Return: The data socket address in the ACEAddress form. It will return a null if no outgoing socket is available.

Exceptions: ServiceUnavailableException, CommandNotImplementedException, PermissionDeniedException

2.2.1.3 GetDataInSocket Command: Gets the command in socket for transmitting data into the daemon. The address that the daemon is listening to is an ACE Address. The socket is an "UDP" type socket.

Java class ACEService: {

String getDataInSocket() }

Return The data socket address in ACEAddress form. It will return a null if no ingoing socket is available.

Exceptions: ServiceUnavailableException, CommandNotImplementedException,
PermissionDeniedException

2.2.2 Behavioral Rule Interface Specification

An ACE service client uses 'add' member functions of a service object to specify Listener objects with client methods that are to be executed when certain notifications are received from a connected service daemon.

```
class ACEService {
        void addShutdownCmdListener(…);
        void removeShutdownCmdListener(…);
        void addResetCmdListener(…);
        void removeResetCmdListener(…);
        void connectService( ACEService OutputService );
}
```

S-2.2.2.1 Deprecated

S-2.2.2.2 Deprecated

S-2.2.2.3 Deprecated

S-2.2.2.4 Connect Services Command: Connects the output data stream of another command interface into input stream of another command. Any connections or conversions that must be made to handle the connections are handled in the lower levels.

```
Java class ACEService: {

void connectService( ACEService OutputService );
```

Return: None

Exceptions: ServiceUnavailableException, CommandNotImplementedException, PermissionDeniedException

Arguments:

OutputService – The ACE Service to connect the output of to the input of the current service.

S-2.2.2.5 Add Shutdown Command Listener Command: This command tells the client side service object to send any shutdown notification commands to the specified listener object. The method executed is specified in Section 2.4.

Java:

```
class ACEService {
        void addShutdownCmdListener( ACEServiceListener listener );
```

Return: None

Exceptions: None

Arguments:

listener – The listener object listening for the shutdown notification command.

S-2.2.2.6 Remove Shutdown Command Listener Command: This command tells the client service object to no longer send any shutdown notification commands to the specified listener object (See section S-2.2.2.5).

Java:

```
class ACEService {
        void removeShutdownCmdListener( ACEServiceListener listener );
```

Return: None

Exceptions: None

Arguments:

listener – The listener object listening for the shutdown notification command.

S-2.2.2.7 Add Reset Command Listener Command: This command tells the client side service object to send any reset notification commands to the specified listener object. The method executed is specified in Section 2.4.

Java:

```
class ACEService {
        void addResetCmdListener( ACEServiceListener listener );
```

Return: None

Exceptions: None

Arguments:

listener – The listener object listening for the reset notification command.

S-2.2.2.8 Remove Reset Command Listener Command: This command tells the client service object to no longer send any reset notification commands to the specified listener object (See section S-2.2.2.5).

```
Java:
class ACEService {
        void removeShutdownCmdListener( ACEServiceListener listener );
```

Return: None

Exceptions: None

Arguments:

listener – The listener object listening for the shutdown notification command.

2.2.3 Mapping Interface Specification

This section contains the ACE Command Semantics of the network commands that clients send to service daemons.

S-2.2.3.1 Reset command: The client side ACEService->Reset method creates an ACE Command for network socket transmission to the server. The ACE Command semantics is as follows:

'Reset' – command name

Arguments: None.

The response from the service daemon to the command is as follows:

'ResetResult' – response name

Arguments:

'status' – Specifies the status, success or fail of the command.

Occurrences: CMDVAL_OCCURS_ONCE
Data Type: CMDVAL_STRING
Extent: { Success, Fail }
Mutually Dependent: None
Mutually Exclusive: None

'msg' – A message about the error that has occurred. It is more specific that an error number.

Occurrences: CMDVAL_OCCURS_ONCE
Data Type: CMDVAL_STRING
Extent: CMDVALEXTENT_ANY

Mutually Dependent: errorno, status
Mutually Exclusive: none

'errorno' – An error number for a failure of the command.

Occurrences: CMDVAL_OCCURS_ONCE
Data Type: CMDVAL_STRING
Extent: CMDVALEXTENT_ANY
Mutually Dependent: msg, status
Mutually Exclusive: none

S-2.2.3.2 Shutdown command: The client side ACEService->Shutdown method creates an ACE Command for network socket transmission to the server. The ACE Command semantics is as follows:

'Shutdown' – command name

Arguments: None.

The response from the service daemon to the command is as follows:

'ShutdownResult' – response name

Arguments:

'status' – Specifies the status, success or fail of the command.

Occurrences: CMDVAL_OCCURS_ONCE
Data Type: CMDVAL_STRING
Extent: { Success, Fail }
Mutually Dependent: None
Mutually Exclusive: None

'msg' – A message about the error that has occurred. It is more specific that an error number.

Occurrences: CMDVAL_OCCURS_ONCE
Data Type: CMDVAL_STRING
Extent: CMDVALEXTENT_ANY
Mutually Dependent: errorno, status
Mutually Exclusive: none

'errorno' – An error number for a failure of the command.

Occurrences: CMDVAL_OCCURS_ONCE
Data Type: CMDVAL_STRING
Extent: CMDVALEXTENT_ANY
Mutually Dependent: msg, status
Mutually Exclusive: none

S-2.2.3.3 Deprecated

S-2.2.3.4 Deprecated

S-2.2.3.5 List Users command: The client side ACEService->GetUserList method queries the service daemon to return a list of user information.

'GetUserList' – command name

Arguments: None:

The response from the service daemon to the command is as follows:

'GetUserListResult' – result name

Arguments:

'user' – An array of users connected to the service daemon.
Occurrences: CMDVAL_OCCURS_ONCE
Data Type: CMDVAL_STRING_ARRAY
Extent: CMDEXTENT_ANY
Mutually Dependent: application, connectionTime, writePermissionsP
Mutually Exclusive: None

'application' – An array of names of application connected to the service daemon. There is a one-to-one order correspondence of this array to the 'user' array.
Occurrences: CMDVAL_OCCURS_ONCE
Data Type: CMDVAL_STRING_ARRAY
Extent: CMDEXTENT_ANY
Mutually Dependent:
Mutually Exclusive: None

'connectionTime' – the number of seconds each user/application has been continuously connected to the service daemon.
Occurrences: CMDVAL_OCCURS_ONCE
Data Type: CMDVAL_INT_ARRAY
Extent: CMDEXTENT_ANY
Mutually Dependent: user, connectionTime, writePermissionsP
Mutually Exclusive: None

'writePermissionsP' – a predicate to indicate if the user has write permissions to the daemon.
Occurrences: CMDVAL_OCCURS_ONCE
Data Type: CMDVAL_WORD_ENUMARRAY
Extent: 'FALSE', 'TRUE'
Mutually Dependent: user, application, connectionTime
Mutually Exclusive: None

‘status’ – Specifies the status, success or fail of the command.

> Occurrences: CMDVAL_OCCURS_ONCE
> Data Type: CMDVAL_STRING
> Extent: { Success, Fail }
> Mutually Dependent: None
> Mutually Exclusive: None

‘msg’ – A message about the error that has occurred. It is more specific that an error number.

> Occurrences: CMDVAL_OCCURS_ONCE
> Data Type: CMDVAL_STRING
> Extent: CMDVALEXTENT_ANY
> Mutually Dependent: errorno, status
> Mutually Exclusive: none

‘errorno’ – An error number for a failure of the command.

> Occurrences: CMDVAL_OCCURS_ONCE
> Data Type: CMDVAL_STRING
> Extent: CMDVALEXTENT_ANY
> Mutually Dependent: msg, status
> Mutually Exclusive: none

S-2.2.3.6 Start Command Group command: See the StartCommandGroup method.

‘StartCommandGroup’ – command name

Arguments:

‘commands’ – The array of commands that were queued up.
> Occurrences: CMDVAL_OCCURS_ONCE
> Data Type: CMDVAL_CMDARRAY
> Extent: CMDEXTENT_ANY
> Mutually Dependent: None:
> Mutually Exclusive: None:

The response from the service daemon is as follows:

‘StartCommandGroupResult’ – response name

Arguments:

‘commandResults’ – an array of the command results in the same order as the commands were in from the client.
> Occurrences: CMDVAL_OCCURS_ONCE
> Data Type: CMDVAL_CMDARRAY
> Extent: CMDEXTENT_ANY

Mutually Dependent: None:
Mutually Exclusive: None:

S-2.2.3.7 GetDataOutSocket Command: The client calls the GetDataOutSocket from the client.

'GetDataOutSocket' – command name

Arguments: None:

The response from the service daemon to the command is as follows:

'GetDataOutSocketResult' – result name

Arguments:

'address' – A String representing the ACE Address of the data out socket.
Occurrences: CMDVAL_OCCURS_ONCE
Data Type: CMDVAL_STRING
Extent: CMDEXTENT_ANY
Mutually Dependent: None
Mutually Exclusive: None

'key' – A string representing the symmetric key to be used between the clients and the server.
Occurrences: CMDVAL_OCCURS_ONCE
Data Type: CMDVAL_STRING
Extent: CMDEXTENT_ANY
Mutually Dependent: address
Mutually Exclusive: None

'status' – Specifies the status, success or fail of the command.

Occurrences: CMDVAL_OCCURS_ONCE
Data Type: CMDVAL_ST RING
Extent: { Success, Fail }
Mutually Dependent: None
Mutually Exclusive: None

'msg' – A message about the error that has occurred. It is more specific that an error number.

Occurrences: CMDVAL_OCCURS_ONCE
Data Type: CMDVAL_STRING
Extent: CMDVALEXTENT_ANY
Mutually Dependent: errorno, status
Mutually Exclusive: none

'errorno' – An error number for a failure of the command.

Occurrences: CMDVAL_OCCURS_ONCE
Data Type: CMDVAL_STRING

Extent: CMDVALEXTENT_ANY
Mutually Dependent: msg, status
Mutually Exclusive: none

S-2.2.3.7 GetDataInSocket Command: The client calls the getDataInSocket from the client.

'GetDataInSocket' – command name

Arguments: None:

The response from the service daemon to the command is as follows:

'GetDataInSocketResult' – result name

Arguments:

'address' – A String representing the ACE Address of the data out socket.
Occurrences: CMDVAL_OCCURS_ONCE
Data Type: CMDVAL_STRING_ARRAY
Extent: CMDEXTENT_ANY
Mutually Dependent: application, connectionTime, writePermissionsP
Mutually Exclusive: None

'status' – Specifies the status, success or fail of the command.

Occurrences: CMDVAL_OCCURS_ONCE
Data Type: CMDVAL_STRING
Extent: { Success, Fail }
Mutually Dependent: None
Mutually Exclusive: None

'msg' – A message about the error that has occurred. It is more specific that an error number.

Occurrences: CMDVAL_OCCURS_ONCE
Data Type: CMDVAL_STRING
Extent: CMDVALEXTENT_ANY
Mutually Dependent: errorno, status
Mutually Exclusive: none

'errorno' – An error number for a failure of the command.

Occurrences: CMDVAL_OCCURS_ONCE
Data Type: CMDVAL_STRING
Extent: CMDVALEXTENT_ANY
Mutually Dependent: msg, status
Mutually Exclusive: none

S-2.2.3.8 ServiceConnect command: The client side command to connect two services together.

‘ServiceConnect’ – command name

Arguments:

‘address’ – the address of the input port to connect to.
    Data Type: CMDVAL_STRING
    Extent: CMDEXTENT_ANY
    Mutually Dependent: key
    Mutually Exclusive: None

‘key’ – the symmetric key to use in the transmission.
    Data Type: CMDVAL_STRING
    Extent: CMDEXTENT_ANY
    Mutually Dependent: key
    Mutually Exclusive: None

The response from the service daemon to the command is as follows:

‘ServiceConnectResult’ – result name

Arguments:

‘status’ – Specifies the status, success or fail of the command.

    Occurrences: CMDVAL_OCCURS_ONCE
    Data Type: CMDVAL_STRING
    Extent: { Success, Fail }
    Mutually Dependent: None
    Mutually Exclusive: None

‘msg’ – A message about the error that has occurred. It is more specific that an error number.

    Occurrences: CMDVAL_OCCURS_ONCE
    Data Type: CMDVAL_STRING
    Extent: CMDVALEXTENT_ANY
    Mutually Dependent: errorno, status
    Mutually Exclusive: none

‘errorno’ – An error number for a failure of the command.

Occurrences: CMDVAL_OCCURS_ONCE
    Data Type: CMDVAL_STRING
    Extent: CMDVALEXTENT_ANY
    Mutually Dependent: msg, status
    Mutually Exclusive: none

2.2.4 Operation Specification

The specifications in this section contain the API for clients to send commands to a service daemon. All of the methods in this section are 'void' return since they do not query the ACEService class or the service daemon.

A summary of the operation methods of the client ACEService class is as follows:

```
class ACEService {

        void resetServiceDaemon(…);
        void shutdownServiceDaemon(…);
        void startCommandGroup(…);
        void endCommandGroup(…);
}
```

S-2.2.4.1 An ACE client side Service object is declared as follows:

Java: abstract class ACEService extends ACEBaseCmd;

S-2.2.4.2 The ACE Client Service object is initialized as follows:

Java: constructor ACEService(String ACEAddress);

S.2.2.4.3 Reset command: The reset command commands the service to change its state to its state before just before the first client connection was created. This involves gracefully shutting down all client connections (doing the requesting connection last so that errors may be returned) and resting hardware for a device control daemon. Each client except the commanding client is sent a 'service reset' notification before its connection is closed.

The client must have administrative 'write' permissions on the service for this command to be successful.

```
Java: class ACEService  {
         void resetServiceDaemon(); }
```

Arguments: None

Return: None

Exception: ServiceUnavailableException, CommandNotImplementedException, PermissionDeniedException

S.2.2.4.4 Shutdown command: This command tells the service daemon to shutdown processing and exit. The shutdown includes:

- No longer accept new connections,
- Gracefully close all connections (including hardware device connections and data streams) except the one the command came in on. A 'service shutdown' notification is sent to each client before the connection is closed.
- Return a success or failure to the 'reset' commanding client,

- Shutdown the command connection the ‘reset’ commanding client. The commanding client does not receive a ‘service shutdown’ notification.
- Shuts down all threads and exits the daemon program.

The client must have administrative ‘write’ permissions on the service for this command to be successful.

Java: class ACEService {
    void shutdownServiceDaemon(); }

    Arguments: None

    Return: None

    Exception: ServiceUnavailableException, CommandNotImplementedException,
        PermissionDeniedException

S-2.2.4.5 Deprecated

S-2.2.4.6 Deprecated

S-2.2.4.7 Start Command Group command: This command is used to start queuing commands that follow this command until a EndCommandGroup command is executed. All succeeding commands are save in a queue and then sent when EndCommandGroup is executed. Each intervening command returns immediately. Only commands that that do not return query values may be queued.

Java: class ACEService {

    void startCommandGroup () }

    Arguments: None

    Return: none

    Exceptions: ServiceUnavailableException, CommandNotImplementedException,
        PermissionDeniedException

    CommandGroupBegun – used when a command group is already being formed.

S-2.2.4.8 EndComandGroup command: This command is used to complete a command group and send the group of commands to the service daemon. See StartCommandGroup command.

Java: class ACEService {

    void endCommandGroup () }

    Arguments: None

    Return: None

Excepetions: ServiceUnavailableException, CommandNotImplementedException,
PermissionDeniedException

2.3 Daemon Specifications

2.3.1 Query Interface Specifications

None.

2.3.2 Behavioral Rule Interface Specification

None.

2.3.3 Mapping Interface Specification

S-2.3.3.1 Reset notification: The server side 'reset' notification creates an ACE Command for network socket transmission to the client. The ACE Command semantics is as follows:

'ResetNotify' – command name

Arguments:

'user' – The name of the owning user of the application that commanded the reset.
Data Type: CMDVAL_STRING
Extent: CMDEXTENT_ANY
Mutually Dependent: None
Mutually Exclusive: None

'application' – The name of the application that commanded the reset.
Data Type: CMDVAL_STRING
Extent: CMDEXTENT_ANY
Mutually Dependent: None
Mutually Exclusive: None

The response from the client is as follows:

No response is sent since the communication socket has been closed by the service.

S-2.3.3.2 Shutdown notification: The server side 'shutdown' notification creates an ACE Command for network socket transmission to the client. The ACE Command semantics is as follows:

'ShutdownNotify' – command name

Arguments:

'user' – The name of the user that commanded the shutdown.
Data Type: CMDVAL_STRING
Extent: CMDEXTENT_ANY

Mutually Dependent: None:
Mutually Exclusive: None:

'application' – The name of the application that commanded the shutdown.
Data Type: CMDVAL_STRING
Extent: CMDEXTENT_ANY
Mutually Dependent: None
Mutually Exclusive: None

The response from the client is as follows:

No response is sent since the communication socket has been closed by the service.

S-2.3.3.3 Deprecated

2.3.4 Operation Specification

S-2.3.4.1 Pre Reset command: The daemon uses this method to specify which method to execute when a 'reset' command is received but before it is processed by the service daemon. The specified method argument is executed while the command threads and command and notification communications are still running.

Java: class ACEServiceDaemonCmds
{ void SetPostResetMethod (Method serviceResetMethod) }

Arguments:
serviceResetMethod – a client method to execute after the 'reset' notification is received and processed.

Return: None

Exceptions: None

void serviceResetMethod(ACECmdLine resetCmdValue)

Arguments:
resetCmdValue – The parsed command line from the client.

Return: None.

S-2.3.4.1 Post Reset command: The daemon uses this method to specify which method to execute when a 'reset' command is processed by the service daemon. The specified method argument is executed after the daemon has closed all command and notification communications and command threads.

Java: class ACEServiceDaemonCmds

    { void SetPostResetMethod (Method serviceResetMethod) }

    Arguments:
    serviceResetMethod – a client method to execute after the ‘reset’ notification is received and processed.

    Return: None

    Exceptions: None

    void serviceResetMethod(ACECmdLine resetCmdValue)

    Arguments:
    resetCmdValue – The parsed command line from the client.

    Return: None.

S-2.3.4.2 Send Notification Command: This method is used to send a notification command that was generated internally, or by hardware (i.e., the command was not sent in from a client, but may be generated by physically powering off a device).

```
Java: class ACEServiceDataThread {
        void sendNotification( ACECmdLine notificationCommand );
}
```

Arguments:
 notificationCommand – the ACECmdLine that is sent as a notification
Return: None
Exceptions: None

2.4 Notification Specifications

This section describes the interface and adapter class that clients use to receive notification commands. In order for clients to receive Service level notifications from a daemon, the clients must implement the ACEServiceListener class. The methods defined in the interface are the methods that are called whenever a Service level notification command is sent to the client.

2.4.1 Interface Specifications

S-2.4.1.1 The Service Interface Class is defined as follows
Java:

```
public interface ACEServiceListener
{
    public void ServiceShutdownCmdNotify( ACEService sender, String user, String application );
    public void ServiceResetCmdNotify( ACESender user, String user, String application );
}
```

2.4.2 Adapter Specifications

S-2.4.2.1 The abstract class is defined as follows
Java:

```
public abstract ACEServiceListenerAdapter implements ACEServiceListener
```

S-2.4.2.2 Each of the methods defined in the ACEServiceListener class are implemented in the ACEServiceListenerAdapter class as trivial implementations, that is each method only returns when called.

3 Design Constraints

None.

4 Deprecated Specifications

S-2.2.2.1 Reset Notification: The client receives a 'reset' notification when some other client connected to the service has issued a 'reset' command. After the service has sent the 'reset' notification is closes the command and notification sockets between the service and the client. The client must use the setNotifyResetMethod to set the client method to execute when a 'reset' notification is received.

Java: class ACEService

{ void setNotifyResetMethod (Object clientObject, Method clientResetMethod) }

Arguments:
clientObject – an instance of an object that has the method 'clientResetMethod'.
clientResetMethod – a client method to execute when the 'reset' notification is received.

Return: None

Exceptions: ServiceUnavailableException, CommandNotImplementedException, PermissionDeniedException

S-2.2.2.2 Shutdown Notification: The client receives a 'shutdown' notification when some other client connected to the service has issued a 'shutdown' command. After the service has sent the 'shutdown' notification is closes the command and notification sockets between the service and the client. The client must use the setNotifyShutdownMethod to set the client method to execute when a 'shutdown' notification is received.

Java: class ACEService

{ void setNotifyShutdownMethod (Object clientObject,
Method clientNotifyShutdownMethod) }

Arguments:

clientObject – an instance of an object that has the method 'clientNotifyShutdwonMethod'.

clientNotifyShutdownMethod – a client method to execute when the 'shutdown' notification is received.

Return: None

Exceptions: ServiceUnavailableException, CommandNotImplementedException, PermissionDeniedException

S-2.2.2.3 Command Watch Notification: The client receives a 'commandWatch' notification only if the client has registered a 'commandWatch' with the service daemon and another user of the service daemon sends the watched command to the service daemon.

Java: class ACEService

```
{ void setNotifyCommandWatchMethod (Object clientObject,
                                    Method clientNotifyCommandWatchMethod) }
```

Arguments:

clientObject – an instance of an object that has the method ‘clientNotifyCommandWatchMethod’.

clientNotifyCommandWatchMethod – a client method to execute when the ‘commandWatch’ notification is received.

Return: None

Exceptions: ServiceUnavailableException, CommandNotImplementedException, PermissionDeniedException

S-2.2.3.3 Add Command Watch command: The client side ACEService->ComandWatch method creates and ACE Command for network socket transmission to the server. The ACE Command semantics is as follows:

‘AddCommandWatch’ – command name

Arguments:

‘CommandName’ – the command name of the command the service daemon is to set a for. The command name has to be of a valid command for the service daemon. An invalid command name will cause an error value in the returned command result.

Data Type: CMDVAL_STRING
Extent: CMDEXTENT_ANY
Mutually Dependent: None:
Mutually Exclusive: None:

The response from the service daemon to the command is as follows:

‘AddCommanWatchResult’ – response name

Arguments:

‘status’ – Specifies the status, success or fail of the command.

Occurrences: CMDVAL_OCCURS_ONCE
Data Type: CMDVAL_STRING
Extent: { Success, Fail }
Mutually Dependent: None
Mutually Exclusive: None

‘msg’ – A message about the error that has occurred. It is more specific that an error number.

> Occurrences: CMDVAL_OCCURS_ONCE
> Data Type: CMDVAL_STRING
> Extent: CMDVALEXTENT_ANY
> Mutually Dependent: errorno, status
> Mutually Exclusive: none

‘errorno’ – An error number for a failure of the command.

> Occurrences: CMDVAL_OCCURS_ONCE
> Data Type: CMDVAL_STRING
> Extent: CMDVALEXTENT_ANY
> Mutually Dependent: msg, status
> Mutually Exclusive: none

S-2.2.3.4 Delete Command Watch command: The client side ACEService->RemoveComandWatch method creates and ACE Command for network socket transmission to the server. The ACE Command semantics is as follows:

‘DeleteCommandWatch’ – command name

Arguments:

‘commandName’ – the command name of the command the service daemon is to remove a watch of. The command name has to be of a valid command for the service daemon that has had a watch set. An invalid command name or a command without a set watch will cause an error value in the returned command result.

> Occurrences: CMDVAL_OCCURS_ONCE
> Data Type: CMDVAL_STRING
> Extent: CMDEXTENT_ANY
> Mutually Dependent: None:
> Mutually Exclusive: None:

The response from the service daemon to the command is as follows:

‘DeleteCommandWatchResult’ – response name

Arguments:

‘status’ – Specifies the status, success or fail of the command.

> Occurrences: CMDVAL_OCCURS_ONCE
> Data Type: CMDVAL_STRING
> Extent: { Success, Fail }

Mutually Dependent: None
Mutually Exclusive: None

‘msg’ – A message about the error that has occurred. It is more specific that an error number.

Occurrences: CMDVAL_OCCURS_ONCE
Data Type: CMDVAL_STRING
Extent: CMDVALEXTENT_ANY
Mutually Dependent: errorno, status
Mutually Exclusive: none

‘errorno’ – An error number for a failure of the command.

Occurrences: CMDVAL_OCCURS_ONCE
Data Type: CMDVAL_STRING
Extent: CMDVALEXTENT_ANY
Mutually Dependent: msg, status
Mutually Exclusive: none

S-2.2.4.5 Add Command Watch command: This command tells the service daemon to notify the client through the notification mechanism when another user of the service daemon executes the specified command. Even though this command changes the state the service daemon the user is not required to have administrative ‘write’ permissions to execute the command. ‘write’ permissions are not required since the service state changes does not impact any other users of the service daemon. There is no need to set a watch for the ‘reset’ or ‘shutdown’ commands since all users are notified of these commands when they occur.

```
Java: class ACEService {
        void addCommandWatch( Method ACEServiceCommandMethod); }
```

Arguments:
ACEServiceCommandMethod – a method of the ACEService object that the daemon is to watch for.

Return: None

Exception: ServiceUnavailableException, CommandNotImplementedException, PermissionDeniedException

BadWatchCommandException – the specified method is not a valid command to watch. This can include the ‘shutdown’ and ‘reset’ methods.

S-2.2.4.6 Delete Command Watch command: This command tells the service daemon to remove a previously commanded command watch. It is an error to specify a method that was not previously set as a command to watch by a previous call to ACEService->CommandWatch. Even though this command changes the state the service daemon the user is not required to have administrative 'write' permissions to execute the command. 'write' permissions are not required since the service state changes does not impact any other users of the service daemon.

Java: class ACEService{

void deleteCommandWatch(Method ACEServiceCommandMethod); }

Arguments:

ACEServiceCommandMethod – a method of the ACEService object that was previously set as a command to watch by a previous call to ACEService->CommandWatch.

Return: None

Exception: ServiceUnavailableException, CommandNotImplementedException, PermissionDeniedException

BadWatchCommandException – the specified method is not a valid command to watch. This can include the 'shutdown' and 'reset' methods.

NoWatchCommandException – the specified method did not have a previous watch set on it.

S-2.3.3.3 Command Watch notification: The server side 'commandWatch' notification creates an ACE Command for the network socket transmission to the client. The ACE Command semantics is as follows:

'CommandWatchNotify' – command name

Arguments:

'CommandLine' – The full command line of the command that was seen by the daemon.
Data Type: CMDVAL_CMD
Extent: CMDEXTENT_ANY
Mutually Dependent: None
Mutually Exclusive: None

The response from the client is as follows:

'CommandWatchNotifyResult' – the name of the command result

Arguments:

5 Glossary

None.

6 Change Log

Version	Date	Changes
0.1	Oct 17 2000	Initial Working Version
0.2	Oct 20 2000	Added remainder of the expected commands.
1.0	Jan 12, 2001	Final Version
		Deprecated the following:
		• S-2.2.2.1
		• S-2.2.2.2
		• S-2.2.2.3
		• S-2.2.3.3
		• S-2.2.3.4
		• S-2.2.4.5
2.0	Feb 15, 2001	• S-2.2.4.6
		• S-2.3.3.3
		Added the following sections:
		• S-2.2.2.5
		• S-2.2.2.6
		• S-2.2.2.7
		• S-2.2.2.8

7 Notes

- none