Road Runner

A TIME WARNER COMPANY

Technical Memorandum June 5, 1998

Abstract

Road Runner Client/Server Session Management Protocol Type 1: Challenge Response Authentication with Server Initiated Status Queries.

Version 1.1

Inquiries directed to:

The Road Runner Group
A Time Warner Company
290 Harbor Drive
Stamford CT 06902-7441
203 351-2000
Fax: 203 351-2299

Terms and Conditions for Use of this Document

_

Road Runner Client/Server Session Management Protocol Type 1: Challenge Response Authentication with Server Initiated Status Queries.

Road Runner Group - A Time Warner Company
Version 1.1
Oct. 1997

_

1.0 Introduction

This document describes a Road Runner session management protocol derived from the Road Runner Client/Server Session Management Services specification.

The protocol defined in this specification supports challenge response authentication and server initiated client status queries.

2.0 Security

To harden the transactions described in this document a challenge-response security method will be employed. Additional methods will be used to prevent replay, denial of service and "man in the middle" attacks.

To prevent "packet cloning", i.e. where a packet with valid credentials is copied and used in a denial of service attack, a random "blinding" value will be incorporated into the hash used to provide authentication credentials.

A "blinding" value is simply a frequently changing value, which when incorporated in a credential's hash, causes the original credentials not be directly visible. A copy of the blinding value is sent in the clear so that the source requesting authentication can use it to recreate the challenge response and determine if authorization should be granted. In addition, in order to make the “blinding” unique (i.e. to prevent the copy and use of “blinded” credentials in other messages), a message type, which is unique to the message transmitted will be included in the hash.

A sequence number will be used for certain transactions to blind the credentials and also prevent replay attack.

On some server platforms the shared secret (i.e. the user’s password) may be stored in the clear, on others it may be encrypted before storage. A “hash method” indicator will be provided in server generated authentication challenges. The hash method informs the client that it should hash the subscriber’s password, using the hash method indicated, before including it in the clients challenge response.

2.1 Client to Server Transaction Security

A secure client to server transaction will start with a message from the client requesting a service from the server. If the request contains a valid user name, the server will issue an authorization challenge to query the client for its credentials. The client should respond to the challenge by providing its credentials (blinded by a blinding value and made unique by the inclusion of the message type). If the client is unable to provide proper credentials, the server will send a response to the client indicating that the client is not authorized to receive the service requested and the server will terminate the transaction. Once properly authenticated, the server will consider the client trusted for the remainder of a particular protocol transaction.

Client responses to server initiated requests must include proper client credentials and a proper sequence number to be accepted by the server.

2.2 Server to Client Transaction Security

Session management server requests to the client are made secure using several mechanisms.

If the client provides valid credentials during the authentication phase of a login transaction, the login server will provide a list of Road Runner servers that are authorized to initiate protocol transactions with the client. The list can contain both server IP addresses or server host names that the client can resolve to IP addresses. When a client application receives a request it will first check the IP address against the list received during the login transaction. If the IP address is not valid a “trusted server” address, i.e. one of the IP addresses in the list, the client will ignore the request.

If a session manager server provides its credentials in a server to client transaction, the server will “blind” its credentials by including a blinding value in the hash of the server’s credentials. The credentials provided will enable the Road Runner client application to validate that the request was received from an authorized Road Runner server.

3.0 Theory of Operation

3.1 Client to Server Transactions

The session protocol will allow a Road Runner client application to perform the following:

3.1.1 Negotiate Protocol. A protocol negotiation service is defined which will allow the Road Runner client and Road Runner session management server to negotiate the transaction protocol to use for a Road Runner session.

The protocol negotiation mechanism allows a single client software implementation to operate in different Road Runner session server environments. A laptop PC subscriber for example, can travel from city to city and the protocol negotiation mechanism will transparently select a Road Runner session management protocol appropriate for use by the client and Road Runner session server.

The client initiates the protocol negotiation transaction by contacting the protocol negotiation daemon listening on a well known TCP server port. “Well known” in this context refers to a server port that is defined and well known within the Road Runner environment. The client provides a prioritized list of protocols it can support in its protocol negotiation request. The server returns a protocol negotiation response which indicates which protocol it has selected for use from the client’s list. The server also returns a login hostname or login IP address and the login port number for the protocol it has selected.

Typically a client will use this transaction when the client is first invoked and then subsequently if the client needs to re-authenticate the subscriber i.e. if the subscriber’s PC supports a power saving suspend mode and the PC exits suspend mode.

3.1.2 Perform a secure user login. The login transaction will allow a Road Runner client to submit a user name and password and obtain a login response which indicates the status of the login request.

The password information will never be transmitted in the clear. During its transaction with the login server, if authentication is required, the client will receive an authenticate challenge. The client will create its challenge response based on the nonce value returned in the authenticate challenge and password the user provides. Before inclusion in the challenge response, the hash method indicated in the server’s authenticate challenge will be applied to the password. For additional security, a "blinding" value (which will include the message type to insure uniqueness), will be hashed into the challenge response to prevent packet cloning attack.. Details on how to calculate the login transaction hash are provided in a section below which provides low level information about protocol messages.

The server's login response is also used to transport protocol configuration information to the Road Runner client, i.e. the login response may contain additional parameters which the client must use to configure the negotiated session management protocol. For the type 1 protocol described in this specification the server will return server port numbers used to support other services provided by the protocol, i.e. the server returns the client status response and logout port numbers. In addition the server will provide a list of authorized session management server names or IP addresses. This list will allow a client application to determine if an incoming request was received from an authorized Road Runner server.

3.1.3 Perform secure user logout. The logout transaction will allow a Road Runner client to request the termination of the current "session". A logout response will be returned which indicates the status of the request.

As in the login transaction, the user's password information will never be transmitted in the clear. During its transaction with the logout server, if authentication is required, the client will receive an authenticate challenge. The client will create its challenge response based on nonce value delivered in the server's challenge and the password the user provides. Before inclusion in the challenge response, the hash method indicated in the server's authenticate challenge will be applied to the password.

Typically a client will use this transaction to manage user access privileges. The client may elect to detect and report client side events that indicate a user session should be terminated. These events may include explicit logout requests from the user or signals from the OS that the current user is logging out of the OS or that the OS is about to shutdown. By trapping these events and using the logout transaction a Road Runner client application can make it difficult for subsequent users to gain the Road Runner access privileges of the "logged in" user. This helps prevent theft of service or violation of privacy.

3.2 Server to Client Transactions

The transaction model will allow a Road Runner server to perform the following:

3.2.1 Obtain the status of the client session. This transaction will allow a Road Runner "session manager", when needed, to query the status of the user's session. If the user's session is active the client will respond with a "session-status" response.

Note: the session server may elect to operate in different modes. It may use this transaction to send status message requests at configurable, periodic intervals or may elect to reduce network traffic by sending these requests only when needed, for example, when network resources are low and need to be reclaimed.
If the client does not respond to a configurable number of client status requests the server implementation will force an implicit logout of the subscriber and reclaim resources used for the subscriber’s session.

To validate status message responses from the client and to prevent replay attack, the status message response will contain the client's credentials, and a sequence number. The sequence number servers a dual purpose, it acts as a blinding value and is also used by the server to help ensure that the clients response is valid, i.e. the server will only accept the status response message if the sequence number received is greater than the last received sequence number. To ensure uniqueness, the message type code of the packet will be included in the clients credentials hash. See the implementation notes section below for details on generating and maintaining the sequence number.

3.2.2 Restart the client. This transaction is used to notify a Road Runner client that it should return to a “startup” state, i.e. the client should initiate a protocol negotiation transaction and attempt to log in the subscriber. The server will provide credentials in the restart request message to enable the client to authenticate the server. For security, a message type code and a time-stamp will be included in the request to “blind” the server's credentials. See the implementation notes below for details on the time-stamp.

4.0 Transaction Details

4.1 Protocol Negotiation Transaction

A protocol negotiation transaction is used to allow the Road Runner client and Road Runner session management server to negotiate the transaction protocol to use for a Road Runner session. The protocol negotiation transaction is a message-based handshake over a standard TCP/IP sockets connection.

The message flow for this transaction is as follows:

1. The client initiates the transaction by sending a protocol negotiation request message to the Road Runner session management server. The request message contains a session ID, a client version indicator, an operating system identifier (i.e. Windows, Mac etc.), OS version information and a list of one or more session management protocol Ids. Each ID represents a protocol that the client is able to support. The elements in the protocol list must be ordered such that the protocols the client most wishes to use appear at the beginning of the list.

2. Upon receipt of a client’s protocol negotiation request, the server validates that a correctly formatted protocol negotiation request was received and that the client software version reported by the client is supported. If the request is found to be invalid for either of these reasons the server will end the transaction by returning a protocol negotiation response. The response will contain a status code, which is set by the server to indicate the type of failure. No additional parameters will be returned in the response. Note that the server's inability to negotiate protocol with the client is not considered a parameter error.
3. If a valid request is received, the server must choose a session management protocol from the client’s list. The protocol selected must be the most preferred protocol supplied in the client’s list that the server can support.
4. If the server is successful in finding a matching, supported protocol, it completes the transaction by responding with a protocol negotiation response. The response contains a status code, the protocol selected and the host name (or IP address) and port number of the login server which the client must use to perform a login transaction for the protocol selected.
5. If the server is unable to support any of the protocols listed in the clients protocol negotiation request message, the server will terminate the transaction by sending a protocol negotiation response. The server must set the returned status code to indicate transaction success or if applicable conditional success and return a zero for the selected protocol. The login host and port parameters will be included in the response but may optionally be set to “null” by the server.

4.1.1 Protocol Negotiation Transaction Flow Diagram

```
Client Message                          Server Message


Protocol Negotiation
Request                    ------------→

                          <------------  Protocol Negotiation
                                         Response
```

4.2 Login Request Transaction

The login request transaction will be used to allow a subscriber to gain access to the Road Runner service features they have been provisioned to use. The user name and password of a subscriber who wishes to log in will be submitted for verification during the login transaction. The login transaction is a message-based handshake over a standard TCP/IP sockets connection.

The message flow for this transaction is as follows:

1. The client initiates the transaction by sending a login request message to the login server. The user name (but not the password), session ID, port number for server requests, and client and OS version information are sent in the message. A login reason code is included in the message. The reason code indicates if this is an initial login request or a re-authenticate request generated when a subscriber's machine exits power saving suspend mode.
2. The server checks the user name submitted. If the user name is invalid, the server will end the transaction by responding with a login response. A status code and optional response text are returned in the login response to indicate that the user name is invalid.
3. If the user name is valid the server challenges the client to provide its credentials by responding with an authenticate response message.
4. The client upon receipt of the authenticate challenge from the server sends an authenticate-login request message which contains its credentials and the blinding value used in creating its credentials. The message type code and blinding value will be included in the hash of the client's credentials to prevent replay attack.
5. The Server responds with a login response message. The login response will contain a status code, which indicates the success or failure of the transaction, and optional response text that provides additional information about the status returned. If the client's credentials are invalid, the server will send the login response and end the transaction. If the client's credentials are valid, the server will log in the subscriber and include additional parameters in its login response. These additional parameters are used by the client to configure the protocol negotiated during the protocol negotiation transaction. Additional protocol parameters for this protocol are the logout port number, status message response port number and a list of authorized Road Runner session management servers.

NOTE: If the login request was valid and the client provided valid credentials, the server will include additional protocol configuration parameters in the login response as described above. To prevent "man in the middle" types of attack the server will compute a hash which includes all of the **parameters** returned in the "successful login" response packet. The calculated hash value will then be "tacked" to the end of the login response packet (i.e. returned in a login hash parameter that is appended to the end of the login response).

The login response hash parameter's value will be a 16 byte value created by performing an MD5 hash on all parameters returned in the login response. The common message header will not be included in this hash and the optional response text parameter will not be included in the hash if it is not present in the login response packet

To insure the validity of the parameters sent in a "successful login" response, the client will apply the shared secret to a hash of all the parameters (except for the hash parameter that appears at the end of the packet) sent in the login response. These parameters were sent in the clear. If the client's computed hash matches that sent by the server the client will assume the parameters are valid.

Note: If the server gets a valid login request from a different user on the current IP address, the server will log the first user out and log the new user in. Refer to the state transition diagram at the end of this document for further details on login request handling.

The following login reason codes are defined for a login request message:

0	Normal
1	Re-Authenticate after suspended

Note: the reason codes are provided for information gathering purposes only. The re-authenticate after suspend reason code is generated when a PC (typically a laptop computer), which supports a form of power management (such as AVP, Advanced Power Management), has been suspended and is exiting the suspended state. In suspended state the PC can not respond to status requests from the server and thus, if suspended for too long the session may be implicitly terminated by the server. On platforms where the OS provides a signal to client applications that the PC is exiting suspend mode, the client, if enabled to do so, can trap the signal and attempt to silently re-establish a session.

Note: See the login response message section below for a list of login response codes.

4.2.1 Login Request Transaction Flow Diagram

```
Client Message                      Server Message


Login Request           ------------→

if (Valid (user name))
 {

                       <------------  Authenticate Response

  Authenticate-Login
  Req                   ------------→
}
                       <------------  Login Response
```

4.3 Logout Request Transaction

The logout transaction begins with a client message to the server requesting to logout i.e. an explicit logout request.

The logout transaction is a message-based handshake over a standard TCP/IP sockets connection.

The server will respond to this request based on the server's state information for this subscriber.

The message flow for this transaction is as follows:

1. The client initiates the transaction by sending a logout request message to the logout server. The user name (but not the password), session ID, and client and OS version information are sent in the message. A logout reason code is included in the message.
2. If the subscriber is already logged out then the server will complete the transaction by immediately sending a logout response message indicating that logout was successful.
3. The server checks the user name submitted. If it is invalid, the server will end the transaction by immediately responding with a logout response. The logout response will indicate that the user name is invalid.
4. If the user name is valid, the server will challenge the client to provide its credentials by returning an authentication response.
5. The client sends an authenticate-logout request message which contains its credentials and the blinding value used in creating its credentials. To prevent replay attack the client's credentials hash will include the blinding value and the message type code for the logout request message.
6. The Server terminates the transaction by responding with a logout response message. The logout response message contains a status code and optional status text. If proper credentials were provided by the client, the server will log out the subscriber.

The following client generated log-out reason codes are defined:

0	User initiated log-out
1	Application Shutdown
2	OS Shutdown.
3	Unknown

Note: See the logout response message section below for a list of logout response codes.

4.3.1 Logout Request Transaction Flow Diagram

```
Client Message                                   Server Message


Logout Request              -----------→
```

```
if (subscriber connected)
{
 if (Valid (user name))
  {
                        <------------  Authenticate Response

     Authenticate-Logout
      Req                ------------→
  }
}
                        <------------  Logout Response
```

4.4 Client Status Request Transaction

A Road Runner session management server initiates a client status request transaction to determine the status of the Road Runner client software. This transaction is carried out using standard UDP packets.

The message flow for this transaction is as follows:

1. The session server initiates the client status request transaction by sending the client a client status request message. A parameter is included in the message that indicates whether status messages will continue to be sent by the server or suspended until further notice. See the message details section below for details on this parameter.
2. The client verifies that the request was received from a valid Road Runner session management server by comparing the IP address of the request packet to the list of valid session management servers it received during the login transaction. If the request was not received from a valid session management server, the request is ignored.
3. If the request was received from a valid session server, the client sends an authenticate-status response message which includes its credentials, a status code and the value used to “blind” the credentials. To prevent replay attack, the hash of the client’s credentials will include the blinding value and the message type code for a client status response.

The following status response codes are defined for a client status response message:

0 Transaction successful. (The client is OK).

Note: A response to the client status request indicates that the client software is active. The status field has been included for future expansion. It may be used for example to indicate that certain functionality of the client is unavailable due to a programming or system error even though the application is still working.

4.4.1 Client Status Request Transaction Flow Diagram

```
Client Message                             Server Message

                         <-----------   Client Status Request

Authenticate-Status  ------------→
```

4.5 Restart Client Request Transaction

A Road Runner session management server initiates a restart request transaction to inform the client application that the server wishes to change configuration parameters of the current session management protocol or change the session management protocol itself. This transaction is carried out using standard UDP packets. The server sends the restart request message to the “request port” specified by the client during the login transaction.

The message flow for this transaction is as follows:

1. The session server initiates the restart request transaction by sending the client a restart request message. The server provides its credentials in the request by performing a hash on the shared secret (i.e. the password), the current nonce value associated with the client session, a server generated “blinding” value, (a 4 octet time-stamp) and the message type code for a restart transaction. The server also includes a clear text version of the time-stamp in the response. The server may report the reason it is requesting the client to restart by including an optional “restart reason” parameter.
2. The client verifies that the request was received from a valid Road Runner session management server by comparing the IP address of the request packet to the list of valid session management servers it received during the login transaction. If the request was not received from a valid server, the request is ignored.
3. The client attempts to authenticate the server. The client validates the servers credentials by performing a hash on the subscriber’s password, the nonce value it previously obtained from the session management server during the login transaction, the clear text blinding value provided in the restart request and the message type code included in the packet’s header. If the client’s hash does not match the hash provided by the server the client will ignore the restart request.
4. If the client is able to validate the server, the client software will enter into a “restart” state. The client will initiate a protocol negotiation transaction followed by a login transaction.

4.5.1 Restart Request Transaction Flow Diagram

```
Client Message                       Server Message

                     <-----------   Restart Request
```

5.0 Message Format Details

This section details the message formats used for client transactions.

NOTES:

- All numeric values in all message packets are in network byte order.

- String data included in messages will be proceeded by string length indicator which defines the length of the string, i.e. string messages with not be null terminated.

5.1 Common Message Header

Transaction message packets will contain a common message header followed by zero or more self defined fields in the format type, length, data.

The common message header is 8 octets in length and has the format defined below.

```
0              15             31             47             63
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| Msg.Type    | Msg. Len    | Session ID                 |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Msg Type - 2 octets, unsigned. Indicates the transaction type for the message.

Msg Len - 2 octets, unsigned. The length of the entire message in octets. This includes the length from the first byte of the common header through and including the last byte of the message packet.

Session ID - 4 octets, unsigned. A logical session identifier. See the implementation note section below for details.

5.2 Message Parameters

Transaction message packets can contain zero or more message parameters. Parameters are self defined fields in the format type, length, data.

Message parameters each contain a fixed length header which is used to specify the type, and total length of the parameter. The format for this header is as follows:

```
0              15             31
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| Param. Type | Param. Len  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Param Type - 2 octets, unsigned. A numeric ID (a name) of the parameter. The ID will be indicated by assigning a number to the parameter. The message parameter type table below lists the numeric values assigned to message parameters.
Param Len - 2 octets, unsigned. Indicates the length of the parameter. The length includes the first octet of the Param. Type through and including the last octet of the parameter's data.

5.3 Message Parameter Types

The following table shows the names and numbers assigned to the parameters used in protocol messages.

Parameter Name (type)	*Assigned Number*
Reserved	0
Protocol List Parameter	1
Protocol Select Parameter	2
Client Version Parameter	3
OS Identity Parameter	4
OS Version Parameter	5
Reason Code Parameter	6
User Name Parameter	7
Request Port Parameter	8
Response Text Parameter	9
Status Code Parameter	10
Authorization Credentials Parameter	11
Nonce Data Parameter	12
Sequence Number Parameter	13
Hash Method Parameter	14
Login Service Port Parameter	15
Logout Service Port Parameter	16
Status Service Port Parameter	17
Suspend Indicator Parameter	18
Status Authentication Parameter	19
Restart Authentication Parameter	20
Time-stamp ("Blinding") Parameter	21
Trusted Session Server List Parameter	22
Login Parameters Hash Parameter	23
Login Host Parameter	24

5.4 Authentication Messages

5.4.1 Authenticate Response Message

The authenticate response message is used to challenge the client for authentication credentials. The format of the message is defined as follows:

```
0              15             31             47            63
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| Common Message Header                                     |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| Hash Method Parameter                       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| Nonce Data Parameter                        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

The Msg. Type field of the common message header will be set to 9 for Authenticate Response messages.

Hash Method Parameter - This parameter's data is 2 octets of unsigned data which indicates the hash method to be used on the shared secret before it is included in the message digest.

The following hash method codes are defined:

0 - do not hash the password before inclusion in the message digest.
1 - perform an MD5 hash on the password before inclusion in the message digest.

Nonce Data Parameter - The data portion of this parameter contains a server generated nonce value. Typically the nonce contains a hash of the subscriber machine IP address, a time-stamp and a server private key, however, a different set of components may be chosen. The primary concern is the uniqueness of the nonce, not the actual components used to generate it. The nonce value will be the 128 bit (16 byte) result of performing an MD5 hash on the components chosen.

5.4.2 Authorization Request Message Format

The Authorization request message format is defined as follows:

```
0               15              31              47             63
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| Common Message Header                                         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| Authorization Credentials Parameter     |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| Time-stamp (“Blinding”) Parameter       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

The Msg. Type field of the common message header will be set to 10 for an Authorization Request messages.

Authorization Credentials Parameter. The data portion of this parameter contains a challenge response that is sent “blinded” by a client generated time-stamp and message type code. The response is a hash of the subscriber’s password (i.e. the shared secret), the nonce value returned in the authorization response, the time-stamp and message type code. Before inclusion in the hash, the hash method indicated by the hash method parameter of the authenticate response must be applied to the shared secret. The formula for the authorization credentials data is H(nonce, secret, time-stamp, message type code).

Time-stamp (“Blinding”) Parameter - the data portion of this parameter contains the time-stamp generated by the client that is used to blind the client’s credentials. The time-stamp data is sent in the clear.

5.5 Protocol Negotiation Transaction Messages

5.5.1 Protocol Negotiation Request Message

The protocol negotiation request message format is defined as follows:

```
0             15             31            47            63
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| Common Message Header                                  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| Client Version Parameter                |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| OS Identity Parameter                   |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| OS Version Parameter                    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| Protocol List Parameter                 |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

The Msg. Type field of the common message header will be set to 1 for a Protocol Negotiation Request message.

Client Version Parameter - The data portion of this parameter is 2 unsigned octets which are the version stamp of the client application.

OS Identity Parameter - the data portion of this parameter is an ASCII string which identifies the Operating System platform the Road Runner client is running on.

OS Version Parameter - the data portion of this parameter is an ASCII string which indicates the operating system version.

Protocol List Parameter - the data portion of the protocol list parameter contains a list of numeric values each 2 unsigned octets in length. Each 2 octet value in the list indicates a Road Runner session management protocol that the client can support. The list is prioritized by the client such that the protocols it desires most to use are located at the beginning of the list. See the “Road Runner Session Management Protocols - Assigned Numbers” document for a list of session management protocols and their assigned numbers.

5.5.2 Protocol Negotiation Response Message

The protocol negotiation response message format is defined as follows:

```
0              15              31              47             63
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| Common Message Header                                         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| Status Code Parameter                         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| Protocol Select Parameter                     |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| Login Host Parameter                          |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| Login Service Port Parameter                  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

The Msg. Type field of the common message header will be set to 2 for a Protocol Negotiation Response message.

Status Code Parameter - the data portion of the status code parameter contains 2 unsigned octets containing one of the following codes:

Group	*Code*	*Description*
Common	0	Transaction successful
Protocol	300	Protocol Neg. Successful but client is out of date.
Neg.	301	Protocol Neg. failed. Client version is invalid.
	302	Protocol negotiation failed. Invalid request.
Server	500	Server unknown error.

Protocol Select Parameter - the data portion of this parameter contains a 2 octet unsigned field that indicates the session protocol the server has selected from the list the client provided during the client's protocol negotiation request. This parameter is not returned in failed protocol negotiation transactions.

Login Host Parameter - Included only if protocol negotiation is successful. The data portion of this parameter is an ASCII text string containing one of the following: a host name OR a dotted quad IP address ex. 192.204.166.76.

Login Service Port Parameter - the data portion of this parameter contains 2 octets of unsigned data that indicate the login service port number for this protocol.

5.6 Login Request Transaction Messages

5.6.1 Login Request Message

The login request message format is defined as follows:

```
0              15              31              47             63
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| Common Message Header                                         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| User Name Parameter                         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| Client Version Parameter                    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| OS Identity Parameter                       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| OS Version Parameter                        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| Reason Code Parameter                       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| Request Port Parameter                      |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

The Msg. Type field of the common message header will be set to 3 for a login request message.

User Name Parameter - the data portion of this parameter is a string containing the user name of the user the client wishes to log in.

Client Version Parameter - The data portion of this parameter is 2 unsigned octets which are the version stamp of the client application.

OS Identity Parameter - the data portion of this parameter is an ASCII string which identifies the Operating System platform the Road Runner client is running on.

OS Version Parameter - the data portion of this parameter is an ASCII string which indicates the operating system version.

Reason Code Parameter - The data portion of this field contains 2 octets of unsigned data which indicate the type of login that is requested.

The following login types are defined for a login request message:

0 Normal
1 Re-Authenticate after suspended

Request Port Parameter - the data portion of this parameter contains 2 octets of unsigned data which indicate the port number the client application will listen on for incoming server requests, i.e. server initiated client status requests and restart requests.

5.6.2 Authenticate-Login Request Message

The authenticate-login request message format is identical to the Authorization Request message format except that the Msg Type in the common header is set to 4.

5.6.3 Login Response Message

The login response message format is defined as follows:

```
0               15              31              47             63
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| Common Message Header                                          |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| Status Code Parameter                      |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| Response Text Parameter (optional)         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| Logout Service Port Parameter              |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| Status Service Port Parameter              |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| Trusted Session Server List Parameter      |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| Login Parameters Hash Parameter            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

The Msg Type field of the common message header is set to 5 for login response messages.

Status Code Parameter - the data portion of the status code parameter contains 2 unsigned octets containing one of the following codes:

Group	*Code*	*Description*
Common	0	Transaction successful
	1	The username supplied was not found.
	2	The password supplied was incorrect.
	3	The account is disabled.
	4	The user has been disabled.
Login	100	Login successful but user was already logged in.
	101	Login authenticate retry limit exceeded.
	102	Login successful but client software is out of date.
	103	Login failed. Client software version is invalid
Server	500	Server unknown error.
	501	Server unable to perform user name validation.
	502	Server unable to perform password validation.

Response Text Parameter - This parameter is optional. If included, the data portion of this parameter contains an ASCII text string message which provides additional information concerning the success or failure of a login request.

Logout Service Port Parameter (optional) - Included if login is successful. The data portion of this parameter contains 2 octets of unsigned data which indicate the server port that will accept logout requests for this protocol.

Status Service Port Parameter (optional) - Included if login is successful. The data portion of this parameter contains 2 octets of unsigned data which indicate the server port that will accept client status message responses for this protocol.

Trusted Session Server List Parameter - Included only if login is successful. The data portion of this parameter is a list containing session server host names, server IP addresses or a mixture of both. The list indicates which Road Runner session management servers are authorized to initiate transactions with the Road Runner client application that is performing the login. The list is an ASCII text string. Each element, i.e. each host name or IP address, is comma separated.

Login Parameters Hash Parameter - the data portion of this parameter contains a 128 bit (16 byte) hash of all the **parameters** being returned in the login response packet with nonce value, the shared secret password, and message type. The common message header is not included in the hash. The formula for the hash calculation is as follows:

H (nonce, secret, status, response text, logout port, status port, trusted server list, msg type).

Note: The type, length and data trio for each parameter are included in the hash.

5.7 Logout Transaction Messages

5.7.1 Logout Request Message

The logout request message format is defined as follows:

The login request message format is defined as follows:

```
0                 15                31                47               63
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| Common Message Header                                                   |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| User Name Parameter                      |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| Client Version Parameter                 |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| OS Identity Parameter                    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| OS Version Parameter                     |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| Reason Code Parameter                    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

The Msg Type field of the common header is set to 6 for a logout request messages.

User Name Parameter - the data portion of this parameter is a string containing the user name of the user the client wishes to log out.

Client Version Parameter - The data portion of this parameter is 2 unsigned octets which are the version stamp of the client application.

OS Identity Parameter - the data portion of this parameter is a string which identifies the Operating System platform the Road Runner client is running on.

OS Version Parameter - the data portion of this parameter is a string which indicates the operating system version.

Reason Code Parameter - 2 octets, unsigned. The reason the client is requesting a logout. NOTE: This parameter is included for informational purposes. It can be used to gather statistics on the type of logouts client machines are performing.

Must be one of the following codes:

0	User initiated log-out
1	Application Shutdown
2	OS Shutdown.
3	Unknown

5.7.2 Authenticate-Logout Request Message

The authenticate-logout request message format is identical to the Authorization Request message format except that the Msg Type field in the common message header is set to 7.

5.7.3 Logout Response Message

The logout response message format is defined as follows:

```
0              15              31              47             63
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| Common Message Header                                    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| Status Code Parameter                      |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| Response Text Parameter (optional)         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

The Msg Type field of the common message header is set to 8 for a logout response message.

Status Code Parameter - the data portion of the status code parameter contains 2 unsigned octets containing one of the following codes:

Group	*Code*	*Description*
Common	0	Transaction successful
	1	The username supplied was not found.
	2	The password supplied was incorrect.
	3	The account is disabled.
	4	The user has been disabled.
Logout	200	Logout successful but user was already disconnected.
	201	Logout authenticate retry limit exceeded.
Server	500	Server unknown error.
	501	Server unable to perform user name validation.
	502	Server unable to perform password validation.

Response Text Parameter - This parameter is optional. If provided, the data portion of this parameter contains an ASCII text string message which provides additional information concerning the success or failure of the logout transaction.

5.8 Client Status Transaction Messages

5.8.1 Client Status Request Message

The server generated client status request message format is defined as follows:

```
0              15              31              47              63
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| Common Message Header                                         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| Suspend Indicator Parameter (optional)  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

The Msg Type field in the common message header is set to 11 for a Client Status request message.

Suspend Indicator Parameter - this parameter is optional. If included, the data portion of this parameter is a single unsigned octet which indicates if server client-status queries are enabled or disabled. A value of zero indicates that server client-status queries are disabled. A value of 1 or greater indicates that server client-status queries are enabled. Not including the parameter is synonymous with sending the default value. The default value is 1, which indicates that server status queries are enabled.

Note: The suspend indicator parameter value may change throughout a session, i.e. the session management server may elect to turn status message requests on and off an unlimited number of times during a session.

5.8.2 Authenticate-Status Response Message

The authenticate-status response message format is defined as follows:

```
0              15              31              47             63
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| Common Message Header                                           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| Status Code Parameter                     |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| Status Authorization Parameter            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| Sequence Number Parameter                 |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

The Msg Type field of the common message header is set to 12 for a authenticate-status response message.

Status Code Parameter - the data portion of the status code parameter contains 2 unsigned octets containing one of the following codes:

0 Transaction successful (The client is “OK”).

Note: A response to the status request indicates that the client software is active. The status code parameter has been included for future expansion. It may be used for example to indicate that certain functionality of the client is unavailable due to a programming or system error.

Status Authorization Parameter - the data portion of this field contains a client hash of the client’s current nonce value, the shared secret a client generated sequence number and the status authorization message code. The formula for the data for this field is H (nonce, secret, sequence #, message type code). Note that the hash method indicated previously during login authentication must be used on the shared secret before inclusion in the client’s hash.

Sequence Number Parameter - the data portion of this field is 4 octets of unsigned data which represent a sequence number generated by the client. The sequence number is used to prevent replay attack. The client increments the sequence number before it includes it in the Sequence Number Parameter and Status Authorization parameter of this message.

5.9 Restart Transaction Messages

5.9.1 Restart Request Message

```
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| Common Message Header                                                 |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| Restart Authentication Parameter       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| Time-stamp (Blinding”)Parameter        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| Reason Code Parameter                  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

The Msg Type field of the common message header is set to 13 for a restart request message.

The following hash method codes are defined:

0 - do not hash the password before inclusion in the message digest.
1 - perform an MD5 hash on the password before inclusion in the message digest.

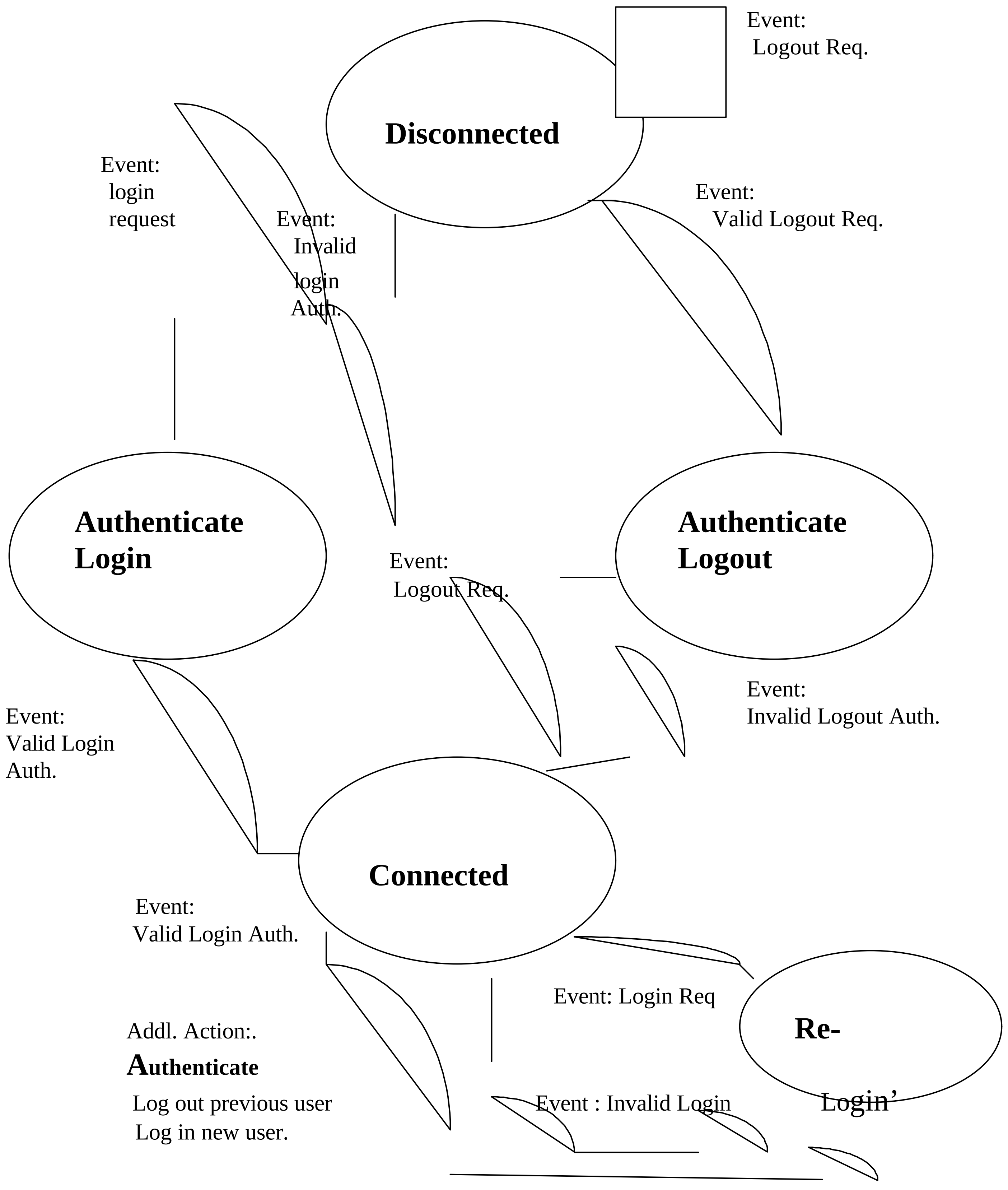
Restart Authentication Parameter - the data portion of this parameter contains a server created hash of the nonce value for the session, the shared secret, a server generated time-stamp and the message type code.

Time-stamp (Blinding”) Parameter - the data portion of this parameter contains a server generated time-stamp value used to “blind” the server’s credentials. The time-stamp is sent in the clear. See the implementation notes below for details on the time-stamp generation and maintenance.

Reason Code Parameter - the data portion of this parameter contains a code which indicates the reason the server is initiating the restart request. The following reason codes are defined:

0 - restart request initiated through administrative interface
1 - 3 reserved
4 - Unknown

6.0 Server State Transition Diagram - Per IP Address



7.0 Server Implementation Notes

7.1 Implicit vs. Explicit Logout

The server will process two types of logout requests.

Explicit logout requests are initiated by client software. These requests always begin with the server's reception of a logout request message. The client software generates the logout request message when it detects a client side logout indication, i.e. the user requested the logout or the OS delivered an event that indicates that the client software should request a logout.

Implicit logout requests are generated by server side software. The server will provide an administrative interface (as detailed below) which allows an implicit logout request (forced disconnect) to occur. In addition the server may perform an implicit logout when a client does not respond to a specified number of server status request messages.

7.2 Status Message Processing

The server will allow the delivery interval for status messages to be configurable. The delivery interval will be specified in seconds.

The server will track the number of concurrent failed status messages sent to a client. The server will increment the failure count when the client does not respond to a status request or when the status response is invalid (i.e. not properly authenticated). It will reset the count when a valid status response is received from the client. If the count exceeds a configurable threshold the server will disconnect the client session i.e. it will perform an implicit logout.

If the status response returned from the client does not have valid authentication credentials, and the failed status count has not been exceeded, the server will immediately retry sending status requests at a different, typically faster, configurable delivery interval. The server will continue sending status messages at the secondary rate as long as invalid status responses are received or until the failed status response threshold is exceeded.

To help control denial of service attacks whereby a client "floods" the server with status responses, the server will maintain a count of packets received from each client. If the packet count exceeds the number of status requests sent to the client, plus a configurable tolerance amount (to allow for valid packet errors), the server will output a log file entry to note the event. See the session manager specifications for details pertaining to the transaction log file.

7.3 Nonce Value, Session ID, Sequence Number and Blinding Value Maintenance

The server will maintain a nonce value and sequence number per session.

Nonce Value - the server will deliver a nonce value to the client application during the authentication phase of a login and logout transaction. It will also provide a new nonce value in the server generated, restart request transaction. The server must update the session table to reflect the last nonce value delivered to the client. The client must include the latest nonce received when it creates its credentials for use in authorization request messages. The nonce value will be transmitted as a stream of 128 bits (16 bytes) that are the result of performing an MD5 hash on the parameters chosen for the creation of the nonce.

Session ID - The session ID is used to support stress testing. It allows a single client machine to run multiple client sessions against the RR server being stress tested.

In normal server operating mode, the server only allows one client session per IP address (i.e. it will not allow multiple client sessions using the same IP address). In order to support stress testing using multiple clients sharing the same IP address, the server and client will both implement a special stress test mode.

When the server runs in stress test mode, it will allow multiple clients to be started and managed, which can use the same IP address. The session ID field allows a particular client session to be identified.

The client will set the session ID as follows:

In "normal" mode the client will set the session ID to any value (some good choices are zero, or the port number the client uses for server requests). The server will ignore the session ID in normal mode and simply "echo" back the same session ID that the client provided.

In "stress test" mode, each client that is invoked will set the session ID to a unique value. As long as the session ID value is unique any value can be chosen. A good choice for the value would be the port number the client has chosen to listen on for server generated requests.

Sequence number - the sequence number is used in client status response messages to prevent replay attack.

When a valid client login completes both the client and session server will reset the sequence number to zero. Before sending a status response, the client will increment the sequence number. The sequence number will then be used when calculating the hash of the client’s credentials that the client provides as part of the status response packet. The client will also provide the sequence number in the clear. The server will track the client’s sequence number. It will check the version of the sequence number send in the clear by the client. If the sequence number is greater than the previous client sequence number, the server will attempt continue to process the response by attempting to validate the credentials received from the client. The server will calculate a hash based on the nonce value, shared secret password and sequence number received in the status response. If the hash matches the hash delivered by the client, the server will accept the status response.

Blinding value - both the client and server will provide a time-stamp used to “blind” the credential information in the authentication responses they exchange. The time-stamp is a 4 octet unsigned value representing the seconds that have elapsed since GMT Jan 1, 1970.

7.4 Administrative Interface

The server should provide both a programmatic and a web based GUI interface
for administration of server functionality such as setting the policy for issuing status messages and overriding the state transition table (for example, providing a way to force a logout).

7.4.1 Administrative Interface

The implementation must allow the following to be performed through the administrative interface:

logging out a specified subscriber or target group of subscribers. The implementation will allow the use of regular expressions to identify the target subscriber or target group of subscribers that will be logged out.

specification of the “failed status response” threshold. This number specifies how many times a server must send a status request to a client without receiving an intervening valid status response. If the threshold is exceeded the server will perform an implicit logout of the client.

specification of the normal status message delivery interval, per subscriber. This positive integer value specifies the interval, in seconds, between the delivery of status requests to a particular client machine or group of client machines.. The implementation will allow the use of regular expressions to change the status interval for a target group of clients.

specification of the status message delivery interval for invalid authentication responses. This positive integer value specifies the interval, in seconds, for the delivery of status requests when status responses contain invalid credentials.

a mechanism to control logout authentication handling. The server’s UI will control whether a valid authentication response is required from the client to complete a logout transaction.

7.5 Detection VS Prevention

The server state machine must support different client environments. These include, but are not limited to, single modem, single PC environments and single modem serving a LAN configurations.

In a single modem, single PC environment the state machine must be able to accept login requests from the IP address assigned to the PC, even if it is already in the connected state for this IP address. This is because the server and client may loose sync if certain client conditions occur. For example, the client machine may crash, and the server may not detect this condition for a period of time (as determined by the status message interval). In this case, the server will continue to think the client is connected. If the server did not allow login requests when the server was in the connected state, the user would have to logout before attempting to log in when they re-powered their PC. This would be very confusing to the subscriber especially since the client software will typically present a login dialog when it is launched. To make it less confusing the server will allow a login request from the client software even if the server is in the connected state. In addition, the server state machine will allow a login request from a different user. The scenario is the same. If a PC crashes we want to allow the same or a different user to restart the PC and log into Road Runner.

Allowing this server state machine behavior however, has an effect on the single modem, LAN environment.

In a LAN environment it is possible, although very difficult, to perform a “denial of service” IP spoof attack. In this type of attack a user connected in a LAN environment can spoof an IP address of another user and attempt to log that user out. The server model will purposely allow these attacks to occur but will log all transaction events to a log file. The log file will allow these attacks to be detected and traced.

8.0 Sample Login Transaction

The following section shows a complete login transaction for a user named Mufasa. The shared secret password is “CircleOfLife”.

8.1 Sample login Request Message:

This is a sample login request assuming the following parameters:

The first 8 octets are a common message header which contains the following fields:

The message type is 0x00003 (login request)
The message is 48d (30h) octets in length.
The session ID is 0x00000000 (arbitrarily chosen by the client).

The Login message contains a User Name Parameter which contains the following fields:

The type for a User Name parameter is 0x0007
The length of the User Name parameter is 0x000A
The data of the User Name parameter is a string containing the user name “Musafa”

The login message contains a Client Version Parameter which contains the following fields:

The type for a Client Version parameter is 0x0003
The length of the Client Version parameter is 0x0006
The client ID and version stamp is 0x0101

The login message contains an OS Identity Parameter which contains the following fields:

The type for an OS identity parameter is 0x0004
The length of the OS identity parameter is 0x0006
The OS identity string is “NT”

The login message contains an OS Version parameter which contains the following fields:

The type for an OS Version parameter is 0x0005
The length of the OS Version parameter is 0x0008
The OS Version string is “4.00”

The login message contains a reason code parameter which contains the following fields:

The type for a reason code parameter is 0x0006
The length of the reason code parameter is 0x0006

The login is a normal login (Type 0).

The login message contains a Request Port parameter which contains the following fields:

The type for a Request Port parameter is 0x0008
The length of the Request Port parameter is 0x0006
The request port is 8001 (arbitrarily chosen).

The login request message format is defined as follows:

```
0              15             31             47             63
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| 0x0003       | 0x0030       | 0x0000       |   0x0000     |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| 0x0007       | 0x000A       | M    u       | f     a      |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| s    a       | 0x0003       | 0x0006       |0x0101        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| 0x0004       | 0x0006       | N      T     | 0x0005       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| 0x0008       | 4    .       | 0      0     | 0x0006       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| 0x0006       | 0x0000       | 0x0008       | 0x8001       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

8.2 Sample Authenticate Response Message:

This is a sample server authenticate response to a client’s login request.

The first 8 octets are a common message header which contains the following fields:

The message type is 0x00009 (authenticate response)
The message is 34d (22h) octets in length.
The session ID is 0x00000000.

The authenticate response message contains a Hash Method parameter.

The type for a Hash Method parameter is 0x000E
The length of the hash method parameter is 0x0006
The hash type is 0, i.e. do not apply any hash method to the password before inclusion in the client’s credentials.

The authenticate response message contains an nonce data parameter generated by the server.

The type for a nonce data parameter is 0x000C

The length of the nonce data parameter is 0x0010
The nonce data is 0x11223344556677889900112233445566 (a fictitious nonce value)

```
0                 15                31                47                63
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| 0x0009         | 0x0034         | 0x0000         |   0x0000      |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| 0x000E         | 0x0006         | 0x0000         |   0x000C      |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| 0x0010         | 0x1122         | 0x3344         |   0x5566      |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| 0x7788         | 0x9900         | 0x1122         |   0x3344      |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| 0x5566         |
+-+-+-+-+-+-+-+-+
```

8.3 Sample Authenticate-Login Request Message:

This is a sample client authenticate-login request message.

The first 8 octets are a common message header which contains the following fields:

The message type is 0x0004 (authenticate-login request)
The message is 52d (34h) octets in length.
The session ID is 0x00000000

The authorization request message contains an Authorization Credentials parameter.

The type for an Authorization Credentials parameter is 0x000B
The length of the Authorization Credentials parameter is 32d (0x0020).
The data for the Authorization Credentials parameter is a hash based on the nonce value, the shared secret, message type and a client generated timestamp.
The hash calculated is 0x11223344556677889900112233445566 (a fictitious value).

The authorization request message contains an Time-stamp parameter.
The type for a time-stamp parameter is 21d (0x0015)
The length of the timestamp parameter is 0x0008
The data for the time-stamp is 0x00004321

```
0                 15                31                47                63
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| 0x0009         | 0x009a         | 0x0010         |   0x7932      |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| 0x000B         | 0x0010         | 0x1122         |   0x3344      |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| 0x5566         | 0x7788         | 0x9900         |   0x1122      |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| 0x3344         | 0x5566         | 0x0015         | 0x0008        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| 0x0000       | 0x4321       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

8.4 Sample Login Response Message

This is a sample login response message indicating that the user name “Mufasa” is not authorized to log in.

The first 8 octets are a common message header which contains the following fields:

The message type is 0x00005 (Login response)
The message is 86d (56h) octets in length.
The session ID is 0x00107932

The logout response message contains a Login Response Hash Parameter which contains the following fields:

The type for a login response hash parameter is 0x0017
The length of the login response hash parameter is 0x0010
The hash of the login response data included in this packet (i.e. the status code and response text is) 0x11223344556677889900112233445566 (a fictional hash value).

The logout response message contains a Status Code parameter which contains the following fields:

The type for a Status Code parameter is 0x000A
The length of the Status Code parameter is 0x0006
The Status code is 0x0001. The user name supplied in the login request was not found.

The login response message contains a Response Text parameter which contains the following fields:

The type for a Response Text Parameter is 0x0009
The length of the Response Text parameter is 48d (0x0030)
The response string is “Only Warner Bros. cartoon characters may log on!!”

```
0              15              31              47             63
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|   0x0005     | 0x0056       | 0x0010       | 0x7932       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|   0x0017     | 0x0010       | 0x1122       | 0x3344       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|   0x5566     | 0x7788       | 0x9900       | 0x1122       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|   0x3344     | 0x5566       |0x000A        | 0x0006       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|   0x0001    | 0x0009      | 0x0030      | O     n     |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| l     y     |       W     | a     r     | n     e     |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| r           | B     r     | o     s     | .           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| c     a     | r     t     | o     o     | n           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| c     h     | a     r     | a     c     | t     e     |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| r     s     |       m     | a     y     |       l     |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| o     g     | o     n     | !     !     |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```