AUSTRALIAN SECURITIES COMMISSION

DATASTREAM APPLICATION COMMUNICATION PROTOCOL SPECIFICATIONS

Version 1.01

4th March 1997

CONTENTS

1. INTRODUCTION

1.1 Background

The Australian Securities Commission (ASC) supports an Application Program Interface (API) for external access to ASC data services. This API is specified at three levels:

- the hardware level, defined in the DATA COMMUNICATIONS specification, which describes connection requirements for hardware and low-level data communications, up to and including the TCP/IP transport protocol. This level covers layers 1 to 4 of the Open Systems Interconnection (OSI) reference model.

- the presentation level, defined in this document, which describes the Application Communication Protocol (ACP) required to support conversations at the higher application level. Intending API users will need this specification to be able to write the supporting software for any applications using the API. This level corresponds to the OSI Presentation Layer, layer 6.

- the application level, defined in "DATASTREAM MESSAGES SPECIFICATION", which defines the layout and content of data service request and reply messages. Intending API users will need this specification to be able to write programs supporting specific application queries and services, such as e.g. a Company Extract. This level corresponds to the OSI Application Layer, layer 7.

1.1 Overview

This document specifies communications protocols at the application communications level for applications using the Australian Securities Commission (ASC) Datastream Application Program Interface (API) for ASC external data services. Specifically, it includes:

- TCP/IP socket API connection by the client application to the Datastream server.

- message structure and fields required to support conversations.

- conversation structure and control.

It does not include hardware and low-level protocols required to support TCP/IP connectivity between the client and the ASC Datastream server. In other words, it assumes a TCP/IP link exists.

Datastream clients are able to obtain services from any ASC applications supported by Datastream by establishing a TCP/IP socket connection with the Datastream server and sending service requests to the Datastream server. Data is sent in both directions as discrete messages of varying size. The first message from the client must include user authentication data (i.e. userid and password). Replies from the Datastream server will include an "id token", which is a randomly-generated tag which the client must insert in the header of the next message to identify the client to the Datastream server without requiring re-authentication of the client userid, and to provide a measure of conversation flow control.

2. TCP/IP SOCKET CONNECTION

2.1 Socket Type and Address

Client applications will connect to the Datastream server by attempting to connect to a specified TCP/IP port on the Datastream server. The specific IP address and port number will be advised when the client is issued with a userid and password allowing Datastream access.

The socket must be a TCP socket (not UDP), hence protocol family PF_INET and socket type SOCK_STREAM.

The first message sent over the connection must contain user authentication data. It is then up to the client whether they use the socket connection for one conversation or many.

2.2 Character Encoding

All messages, both client and reply, must consist entirely of printable ASCII characters in the range 32 - 126 inclusive, plus tab (9) and linefeed (10).

All message headers will conform to the same rule, so that the data can be easily human-readable.

2.3 Message Header Data types

Message header data fields can be of two types:

- alpha, consisting of ASCII characters 32 - 126;
- numeric, consisting of ASCII characters 48 - 57.

3. MESSAGE LAYOUT

3.1 Overview

Client messages consist of:

- a fixed length message header containing control data.
- a variable length application transaction message in a format specified by the application specification documents.

Reply messages consist of:

- a fixed length message header containing control data.
- variable length application transaction reply data.

3.2 Message Headers

3.2.1 Client Message

Client message headers are defined as follows:

Offset	Length	Data type	Field name	Description
0	8	alpha	user_id	In the first message of a client session this field contains the Datastream client userid assigned by the ASC, blank filled. In the second and subsequent messages this field must contain the token supplied in the previous reply message from the Datastream server.
8	8	alpha	password	In the first message of a client session this field contains the Datastream client password as stored in Datastream server user table. In the second and subsequent message this message must be blank filled.
16	8	alpha	trans_ type	Identifies ASC service (and hence application), blank filled. Services are 1. "dsascot" (ASCOT Information Products) 2. "dsusrsrv" (Datastream user services).
24	8	numeric	msg_len	Length of the application message which follows, i.e. excluding this header. Format is ASCII numerics, zero filled.

Total header length = 32 bytes.

The maximum message length is theoretically unlimited, but in practice applications will impose limits on the amount of data that can be transferred in or out, either by the nature of the application message or by arbitrary application rules.

At the ACP level, an arbitrary limit of 1Mb is imposed on message length, to trap looping processes and to avoid pointless overloading of servers. If an overlength inbound message is detected, then the Datastream server will return an error message. If an outbound message reaches the limit then the outbound message will be truncated. Since ASC applications will be expected to apply their own controls, this is a last-ditch defence measure, and should rarely, if ever, occur in practice.

3.2.2 Reply Message

Offset	Length	Field name	Description
0	8	id_token	A random value generated as a temporary client id, which the client must include in the user_id field of the next message. The value is formatted as a hexadecimal string padded to the full 8 bytes.
8	12	timestamp	Time that message was received by the Datastream server, in format YYMMDDHHMMSS. This is supplied to assist in identifying related events in the ASC network that occurred at the same time as the message was being processed.
20	8	status_code	This field indicates the status of the previous message at this protocol level. If an error occurred then this field is set to a 4-digit error number (see "Error Handling") and the message body contains error text and supporting data. If no error occurred then this field is set to zeroes. Note that this field describes errors at the ACP level, not the application level. Applications return application error data in application reply data.
28	8	msg_len	Length of the application message which follows, i.e. excluding this header. Format is ASCII numerics, zero filled.

Total header length = 36 bytes.

4. MESSAGE CONTROL FLOW

4.1 Overview

Conversations always proceed in a strictly defined sequence. Because the socket connection is defined to be TCP, both client and server are easily able to preserve conversation state between messages. This means that differentiating between first and subsequent messages is not difficult, allowing the conversation flow to be tightly controlled.

Message sequence within a conversation and a session will always be as follows:

- client sends first message, with userid and password. Datastream validates userid/password.
- if authentication processing fails to validate the userid/password then an error message is returned.
- Datastream uses the trans_type field to decide routing of the message to the appropriate ASC application.
- if Datastream does not receive a reply from the ASC application within a reasonable period (initially 1 minute) then an error message is returned to the client and the session ended by closing the socket.
- the selected ASC transmission returns application data to the Datastream server.
- the Datastream server composes a reply message for the client consisting of a reply header plus the application data exactly as supplied by the ASC application. The reply header includes a new randomly-generated id_token field value. This reply message is then sent to the client.
- the Datastream server waits for a new message from the client. If one is received then it is processed as above, except that for the second and subsequent messages, the user_id is not authenticated. Instead, the id_token is compared with the id_token returned in the previous reply. If they do not match, or if a non-blank password is received, and error message is returned and the connection is closed. If a new message is not received within the timeout period of 1 minute, then the connection is closed. Client applications which discover that a connection has been closed must reopen the connection and resupply the user-id and password.

When considering the issue of timeouts, developers of client applications must keep in mind that connections can be re-established very quickly. User authentication processing on the Datastream server is extremely fast. Taken together, these things mean that the overhead of establishing or re-establishing a connection is very low, so there is no need to retain an open connection if there is no client traffic.

This approach is based on Web-server methods. With Web servers, timeouts are typically 15 seconds, so Datastream is in fact allowing more time for a client response within the same conversation than the typical Web server.

4.2 First Client Message of Session

This message must always include authentication data consisting of userid and password, which must match the values stored in the Datastream access control database.

Userids are supplied when the ASC advises client administration of permission to access Datastream. This advice will also include an initial password. Note that passwords are valid for 30 days only and must then be changed. Request messages with expired passwords will only be allowed past the validation check if they are requesting a password change.

Passwords can be changed by using a Datastream password change application message - see "Changing Passwords" below.

5. ERROR HANDLING

5.1 Overview

Where possible, errors are handled by the application, and error text is returned to the user formatted according to the message language used by the application. Where this is not possible, then the Datastream server handles the error by returning:

- an error number in the status_code field of the reply message header.
- supporting information as structured text in the transaction data part of the reply message, i.e. following the header. As always, the header will indicate the length of this error message data. The layout of this structured text is described in the section.

As much information as possible will be returned to the client as well as being logged by the server, so that support staff can identify the cause of the error as quickly as possible.

5.2 Client Action on Error

In general, clients should be advised that if they receive a Datastream server error, they should retry the operation. If the problem persists, they should report it to their help desk.

If client server processes retry queries, then they should close the connection (if it has not already been closed by the Datastream server) before retrying for any errors between 100 and 201.

A more generalised method of handling the situation would be to retry on the same connection, and automatically attempt to reconnect if the connection is no longer open.

5.3 Errors in Downstream Applications

If fatal application errors occur downstream from the Datastream server, i.e. after the message has been passed on to the application, then the message is lost, and the client application must resubmit it.

The reasons for this lie in the server design. The Datastream server has been intentionally designed as a generic server able to process high volumes of messages of almost any size. To accomodate this, messages are not stored, but are streamed through to the application serving the requested transaction type.

Where application transactions are billable, a backout mechanism has been included in the Datastream design so that clients will only be billed for transactions where the reply has successfully been sent.

5.4 Error Messages

Datastream errors are listed here, together with references to notes in the following section which provide further explanation of error conditions and handling.

Error no	Description	Note
101	Header message length exceeds maximum	N1
102	Unknown userid	N2
103	Invalid password	N3
104	Invalid ID token	N4
105	Error sending to client	N5
106	Error receiving from client	N5
107	Protocol Violation	N6
108	Password expired	N7
201	Unknown transaction type	N8
202	Error sending to application	N9
203	Error getting reply from application	N10
204	Application error while processing request	N11

5.5 Error Message Supporting Information Layout

As described above, when an error message is returned to the client, as much supporting information as possible is included in the form of structured text, to help support staff diagnose problems quickly.

This supporting information is structured into a set of elements which have a common form, and are separated by commas. Elements consist of an upper case label, an "=" sign, and a text string. This gives the message the form of:

LABEL1=string1,LABEL2=string2,...,LABELn=stringn

If data transmitted by the client is included, then any embedded commas are translated to space characters before assembling the reply message, so that the structure of the reply message can be relied on.

Labels and the elements they describe are:

Label	Element contents	Note
ERROR	The error number, as listed in section 5.4	
MSG	The error message text for the error number	N12
SYSERR	Operating system error id, if any	
DIAGMSG	Additional diagnostic information	N13
PROCTYPE	The type of process that trapped the error	N14
PROCNUM	The id of the process that trapped the error	N15

CLMSGID	The message id assigned to the client message	N16
USER_ID	The userid received in the first client message	
RECVHDR	The message header received from the client	N17
RECVMSG	The message transaction data from the client	N18

Rules governing message composition are:

- all of the above elements are included in the reply message.
- if there is no data to include as the text string portion of an element, then the element is included, but with a null text string, i.e. just the LABEL= part.
- received data is only included if processing reached the point of reading it from the socket. For example, if an invalid password is detected, then the rest of the client message is ignored, and so will not be read and included in the reply message.
- some characters are translated, as follows:
    - tabs (ASCII 09) are translated to '>' (ASCII 62).
    - line-feeds (ASCII 10) are translated to '\' (ASCII 92).
    - commas are translated to space (ASCII 32).
    - all other characters outside the range ASCII 32 - 126 are translated to space (ASCII 32).

These translations are done to make the resulting message more consistent and readable.Note that the supporting information described here is also logged by the Datastream server. This has two implications:

- the client message id can always be used to match a client error message with a datastream log record - but note that logs are not kept forever. The log record can also be matched with ASCOT billing records, so that transactions which are billed but result in an error can be automatically identified and backed out.
- because the error messages are reliably structured, text processing tools can be used to extract and analyse errors for patterns.

EXAMPLE

Here is an example of the reply message transaction data sent to a client following an application error:

```
ERROR=204,MSG=Application error while processing request,
SYSERR=0,DIAGMSG=ASC APPLICATION ERROR 3148 LINE 1230
PROGRAM GDS001P,PROCTYPE=dsascot   PROCNUM=0,
CLMSGID=3292914A,RECVHDR=ASPIT51 ASPIT51 dsascot
00000066,RECVMSG=YHDASCDRON>0100>9>CI8130>A2185>2190
8\YRQF1>2>SYDNEY\YTRENDDRON>3\
```

This shows that an application error occurred at line 1230 of an application program GDS001P, while processing the client request message, which happened to be an organisation name search request.

5.6 Note Refererences

Ref	Description
N1	The msg_len field in the message header indicates that the message length is greater than the maximum allowed (currently 1Mb). The message cannot be processed.
N2	The userid supplied in the header was not found in the Datastream userid file.
N3	The password supplied in the header does not match the password in the Datastream userid file for the userid supplied.
N4	The token supplied on a second or subsequent message in a session did not match the ID token supplied in the last message sent to the user. The session is terminated after sending the error message.
N5	Clients are unlikely to receive a message containing this error, because the connection will be closed by the server when it encounters the error condition. However the error will be logged by the server. This type of error can also be significant because the client may not receive the results of a billable transaction. In this case, the transaction billing will be automatically reversed in ASCOT.
N5	Clients are unlikely to receive messages containing these errors, because the connection will be closed by the server when it encounters the error condition. However the errors will be logged by the server.
N6	This covers any error condition caused by invalid header data not covered by any of the above errors.
N7	Password supplied in header of first message for a session is valid for the userid, but has not been changed for more than 30 days. The only message which will be accepted for that user is a password change message.

N8	The value in the trans_type field of the message header did not match any of the supported Datastream server transaction types.
N9	The message was received ok by the Datastream server, but an error occurred when attempting to send the message to the internal ASC application server responsible for processing application messages of the type received. The message is discarded, and the client should retry the operation.
N10	The message was sent ok to the internal ASC application server responsible for processing application messages of the type received, but an error occurred receiving a reply from the application server. This type of error can also be significant because the client may not receive the results of a billable transaction. In this case, the transaction billing will be automatically reversed in ASCOT.
N11	The requested application encountered an error while processing the request which it was not able to handle at the application level. The error message will contain additional diagnostic information from the application, which may only be intelligible to programmers.
N12	This is the error message text from the table in section 5.4 above.
N13	Where this is supplied, it is consists of extra information generated by the software at the time the error occurred which helps to identify the problem from the software's viewpoint.
N14	This identifies where in the chain of processes handling the client message that the error occurred. This will be the trans_type if the error occurred while the message was being processed by the application, otherwise it will have the value "datastream" to indicate that the Datastream front'end processor was in control of the message at the time of the error.
N15	There can be many instances of a given type of process running in parallel, so this identifies which of those instances was the one handling the client message which got the error.
N16	Each client message is assigned a unique message id by the Datastream server when the message is first received. This is used to match log and database data from different parts of the system, so that, for example, billable transactions which generated output which failed to reach the user due to an error can be identified and subsequently automatically reversed.
N17	Header data received from the client is displayed as is, except that any embedded unprintable characters or commas are translated to space characters before including in the reply error message. Unprintable characters are defined to be any characters outside the range ASCII 32 - 126.

N18	Up to 512 characters from the start of the client message transaction data is included here. This should normally be sufficient to include all the data received. As with the message header, any unprintable characters or commas are translated to space characters before inclusion in the reply message. Unprintable characters are defined to be any characters outside the range ASCII 32 - 126.

6. USER SERVICES

6.1 Overview

To access datastream user services, the message header trans_type must be set to "dsusrsrv".

User services are requested using application messages constructed using the message language defined in the INFORMATION PRODUCTS SYSTEM DATASTREAM MESSAGES SPECIFICATION. Segment tags for user services messages start with X instead of Y, to avoid confusion with segments used in Information Products System messages.

6.2 Changing Password

6.2.1. Overview

Passwords can be changed by using the Datastream password change application message URPW. Passwords remain valid for 30 days from the time they were last changed or set. After the 30 days, only URPW messages will be accepted for a user.

6.2.2. Password Change URPW Request Message (version 1.00)

Seq	Group	Segment	Element	Usage	Repeat	Context Meaning	Notes	Codes
1	**1**			M		Header Group		
2		**XHD**		M		Header Segment Identifies the message type		
3			0018	M		Message type. Must be = 'ASCURPW'		
4			0012	M		Message version		
5	**2**			M		Request Group		
6		**XPW**		M		Request Password Change		
7			0018	M		Userid		
8			0018	M		Old password		
9			0018	M		New password		
10	**3**			M		Trailer Group		
11		**XTR**		M		Trailer Segment. Flags message end		
12			0018	M		Must be = 'ENDURPW'		
13			0007	M		Segment count		

New password requirements are:

- must be between 6 and 8 characters long.
- must contain only upper or lower case alphabetic or numerics (note that passwords are case sensitive).

6.2.3. Password Change USPW Reply Message (version 1.00)

The server will provide the result of the change in the following message:

Seq	Group	Segment	Element	Usage	Repeat	Context Meaning	Notes	Codes

1	**1**			M		Header Group		
2		**XHD**		M		Header Segment Identifies the message type		
3			0018	M		Message type. Must be = 'ASCUSPW'		
4			0012	M		Message version		
5	**2**			M		Results Group		
6		**XPR**		M		Password Change Request Results		
7			0016	M		Password change result code		0001
8			0027	M		Password change result message text		
9	**3**			M		Trailer Group		
10		**XTR**		M		Trailer Segment. Flags message end		
11			0018	M		Must be = 'ENDUSPW'		
12			0007	M		Segment count		

6.2.4. Code table 0001 - Password Change Result codes

PW00	Password changed ok
PW01	Userid not same as in message header
PW02	Invalid old password
PW03	New password less than 6 characters
PW04	New password is same as old
PW05	URPW message had invalid structure
PW06	Invalid characters in new password
PW07	Internal error attempting to change password

Appendix A - Amendment History

Version 00.20
Released to brokers as a draft for discussion on 2/10/96.

Version 00.30
Released to brokers on 11/11/96.
5.2 Error numbers have changed completely, and explanations expanded.
6.3 Change offsets in password change message and reply. Change password change errors.

Version 1.00
Released to brokers on 16/12/96
3.2.1. Added "dsusrsrv" as a service
4.2 Passwords now expire after 30 days.
5. Error handling rewritten completely, but note that error numbers and meanings do not change, except to add error 204.
6. Add user services "dsusrsrv", alter password requirements, alter password maintenance.
B. Add appendix B

Version 1.01
Released to brokers on 04/03/97
3.2.1. change length from 24 to 32 (correction is to incorrect doco, not to message)
5.1. remove detail
5.2. add detail
5.5. add USER_ID details to table
5.5. add detail about translation

Appendix B - User Services Sample Messages

Following are sample application messages. Delimiter characters are shown as > for tab (element delimiter), and \ for newline (segment delimiter).

URPW (version 1.00)

```
XHDASCURPW>0100\
XPWASPIT01>GO2TOWN>FLY4ME\
XTRENDURPW>3\
```

USPW (version 1.00)

```
XHDASCUSPW>0100\
XPWPW01>Password changed ok\
XTRENDUSPW>3\
```