Advantage Optimum Ô Toll-free Call Detail Service

ATID-0009

January, 2000

Terminal-to-Network Interface

TABLE OF CONTENTS

DOCUMENT HISTORY

1	January, 2000	Initial issue

DISCLAIMER

Aliant Telecom Inc. reserves the right to modify the interface described in this document for any reason including, but not limited to, ensuring that it conforms with standards promulgated by various agencies from time to time, utilization of advances in the state of the technical arts, or the reflection of changes in the design of any equipment, techniques or procedures described or referred to herein.

ALIANT TELECOM INC. SHALL NOT BE LIABLE FOR ANY DAMAGES OR INJURIES INCURRED BY ANY LEGAL PERSON OR PERSONS, INCLUDING BUT NOT LIMITED TO CORPORATIONS, ARISING DIRECTLY OR INDIRECTLY FROM ANY INCOMPATIBILITY WITH THE NETWORK, OR ANY CAUSE WHATSOEVER.

Readers are specially advised that the technical requirements contained herein may change.

If further information is required, please contact:

Telephony Standards
Suite 640
160 Elgin Street
Ottawa, Ontario
K1G 3J4

In Canada:	1-877-77-TELCO (83526)
Worldwide:	613-781-7393
Fax:	613-781-1658
E-mail:	disclosure@aliant.cdn-telco.com
Web-site:	aliant.cdn-telco.com

1.0 Service Description

Call Detail Reporting is a feature of *Advantage Optimum*™ Toll-free service which provides Subscribers with call related information. Call Detail information is provided electronically and can now be transmitted to the Subscribers CPE over the Internet

2.0 Transmission Protocol

2.1 Characteristics

- CDR Data is transmitted over a simple TCPIP session over the Internet. Basic and standard TCPIP protocols and software tools are used to transmit the data.

- Client authentication and data protection is achieved through use of the Microsoft Crypto API found in MS NT 4.0 Server software.

- The Telco system transmit data packets in TCPIP session. The subscriber CPE should maintain the TCPIP session alive, continuously and indefinitely. A Keep Alive message is send periodically when no data is transmitted.

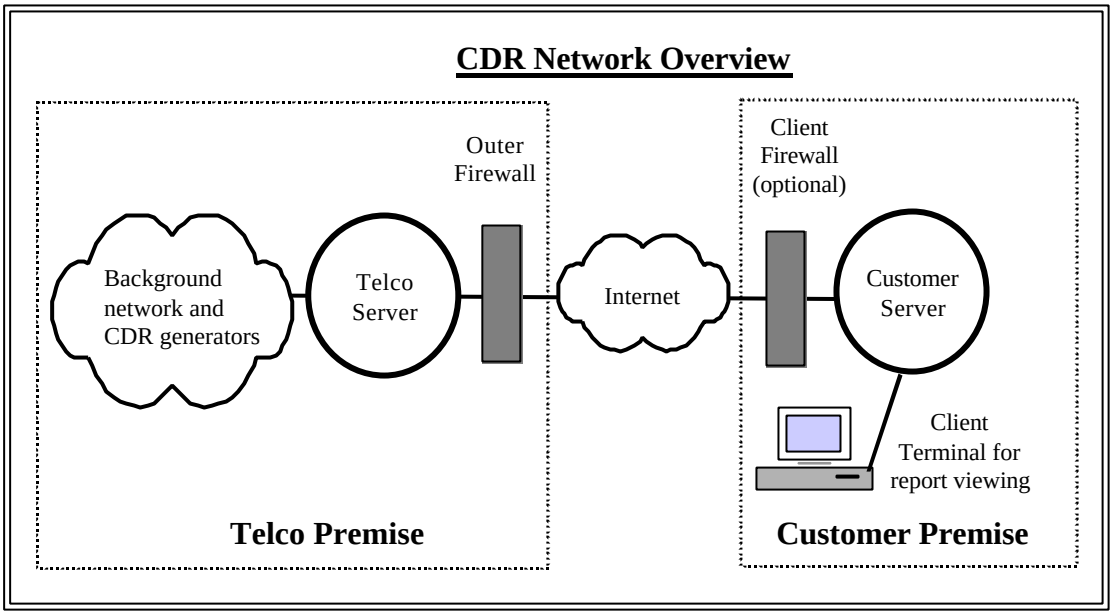
- Data parameters which are not recognized by the terminal should be ignored (i.e. the corresponding data should not be processed).

- Data packets will be transmitted only when the Subscribers CPE is connected (via an Internet connection) to the Telco's systems. All data messages, which correspond, to 800/888 calls, are kept by the Telco's systems and can be recovered or re-transmitted.

- The Client Server should keep track of when it is connected. This will enable data recovery for missing time periods.

3.0 Network Description

Once the Client authenticates itself to the Telco server, the data is sent from the Telco server to the Client server. The Client must establish and maintain a TCPIP session with the Telco. See the network diagram below.



3.1 Preliminary Information Exchange

- Upon subscription to this service, the Telephone Company Business Office sends the TCPIP address and port number of the Telco Server, the clients identification number and specific client Passcode, to the subscribing client. The client needs this information to enter into a session with the Telco.

3.2 Communications Protocol

- The Telco Server will be listening on the Internet for the Client over a specified port.

- A session is established when the Client makes a request to the Telco server to connect. The Telco replies by sending standard 128 bits 3DES based challenge and it's Public Key to the Client for authentication purpose. This reply is encrypted with the Client's Passkey.

- The Client decrypts the reply and calculates the proper response through the MS Crypto API. The Client generates a 3DES session key. Then the Client encrypts both with the Telco Public Key and returns the encrypted message to the Telco Server.

- The Telco decrypts the message, verifies the submitted response and if valid begins sending CDR data for the specific customer. The data is encrypted by the session key.

- During a session, the Telco will require that the Client generate a new session key for security.

- Each day, the Telco will require that the Client re-authenticate itself.

- The Client can issue a request for any missing data on successful connection to the Telco.

- Throughout a session, the Telco can send Keep Alive messages and the Client must respond with a Keep Alive Answer. The specifications of this message are included further.

- At anytime, the Telco can send a Status Probe and the Client must respond with a Probe Answer.

4.0 Communications Session Layout

- The following text describes the CDR interface in a standard C programming language.

- All messages exchanged between the Telco and the Customer server have the following general format: a message leader and a message payload.

- The Message Leader is described below:

Name	**Length**	**Value / Remark**
Leading Word	WORD	0xEEAA
Message Type	WORD	(See details)
Payload Size	WORD	Size of Payload following leader
Reserved	WORD	0
Reserved	WORD	0
Reserved	WORD	0
Tag	DWORD	User use. Returned as received in responses.
Reserved	WORD	0

- Messages Received by the Customer Server

Message	Value	Remark / Payload Structure
Logon Success	0x0111	None
Logon Failure	0x0112	None
Shutdown	0x0123	None
CDR	0x0200	1
CPM	0x0201	2
Probe Request	0x0304	3
Recovery Parameters Response	0x0501	4
Time Synchronization	0x0143	5
Heartbeat	0x0131	None
Security Challenge Request	0x0600	6
Security Challenge Acknowledgement	0x0602	None
Recover Data Response	0x0503	11
Recovered Period	0x0504	12
Recovery Error	0x0503	13

- Messages Sent by the Customer Server

Message	Value	Remark / Payload Structure
Identification	0x0101	7
Logon	0x0110	8
Shutdown	0x0123	None
Probe Response	0x0308	9
Recovery Parameters Request	0x0500	10
Recover Data Request	0x0502	14
Security Challenge Response	0x0601	15
Security Challenge NACK	0x0603	16

- Payload Structure

Payload ID	Structure
1 (CDR)	typedef struct CallStatus_s { unsigned int BusyNetwork : 1; unsigned int BusyCpe : 1; unsigned int BusyOther : 1; unsigned int Answered : 1; unsigned int UnansweredAgentHangup : 1; unsigned int UnansweredCallerHangup : 1; unsigned int Complete : 1; unsigned int Error : 1;

	}
	
	typedef struct CallReleaseCause_s // Network, CPE, Other, Caller, Agent }
	{
	unsigned int BusyNetwork : 1;
	unsigned int BusyCpe : 1;
	unsigned int BusyOther : 1;
	unsigned int CallerHangup : 1;
	unsigned int AgentHangup : 1;
	}
	
	typedef struct CallFeatures_s
	{
	unsigned int Overflow : 1;
	unsigned int CourtesyResponse : 1;
	unsigned int Prompter : 1;
	unsigned int DisplayBlocking : 1;
	unsigned int OverflowIn : 1; // Presently, ignored
	unsigned int OverflowOut : 1; // Presently, ignored
	} CallFeatures_t;
	
	typedef struct Cdr_s {
	__int64 EcdrInternalId;
	int CallId;
	Timestamp_t ReleaseTime;
	Timestamp_t CallDateTime;
	char TollFreeNumber [11];
	char TerminatingNumber [16];
	bool TerminatingIsInternational;
	char OriginatingNumber [16];
	bool OriginatingIsInternational;
	unsigned int CallDuration; // in seconds
	unsigned int RingDuration; // in seconds
	CallStatus_t CallStatus;
	CallFeatures_t FeatureEmployed;

2 (CPM)	typedef struct Cpm_s { __int64 EcdrInternalId; int CallId; Timestamp_t CallDateTime; char TollFreeNumber [11]; char TerminatingNumber [16]; bool TerminatingIsInternational; char OriginatingNumber [16]; bool OriginatingIsInternational; unsigned int Duration; CallStatus_t CallStatus; CallFeatures_t FeatureEmployed;
3 (Probe)	typedef struct OamProbes_s { char OriginatingComputerName[32]; char OriginatingSS7FeederName[41]; DWORD OriginatingProcessId; unsigned long ProbeID; Timestamp_t Timestamp; char TollFreeNumber[11];

	int CustomerHandle; char CustomerName[41]; char CustomerServerName[41]; } OamProbes_t;
4(Recovery Param Response)	typedef struct RecoveryParamResponse_s { unsigned long RequestID; int CdrRecoverablePeriod; int CpmRecoverablePeriod; int CdrFileArchiveSize; int CpmFileArchiveSize; Timestamp_t GMTTime; } RecoveryParamResponse_t;
5 (Time Sync)	typedef struct TimeSynch_s { Timestamp_t GMTTime; } TimeSynch_t;
6 (Security Challenge)	typedef struct ChallengeRequest_s { BYTE ChallengeID[32]; unsigned short TryCount; unsigned short IdentificationOffset; DWORD PublicKeyBlobSize; } ChallengeRequest_t;
7 (Identification)	typedef struct sMsgAppIdentification { unsigned int ApplicationID; } tMsgAppIdentification;
8 (Logon)	typedef struct sMsgLogon { char Username [64]; char Password [64]; } tMsgLogon;

9 (Probe Response)	typedef struct OamProbesResponseFromCustServer_s
	{
	OamProbes_t OriginalProbe;
	int CustServerTag;
	bool CustServerResultFlag;
	int ErrorCode;
	char ErrorText[80];
	bool CustDatabaseResultFlag;
	int DbErrorCode;
	char DbErrorText[80];
	Timestamp_t CustServerProcessingTimestamp;
	}

10 (Recovery Parameter Request)	typedef struct RecoveryParamRequest_s { unsigned long RequestID; } RecoveryParamRequest_t;

11 (Data Recovery Response_	typedef struct RecoveryDataResponse_s { unsigned long RequestID; tMsgSequencer SequenceInfo; } RecoveryDataResponse_t;
12 (Date Recovered Period)	typedef struct RecoveryDataEndOfPeriod_s { unsigned long RequestID; Timestamp_t StartTime; Timestamp_t EndTime; unsigned long NumberOfRecordsRecovered; bool EndOfRequest; } RecoveryDataEndOfPeriod_t;
13 (Recovery Error)	typedef struct RecoveryDataError_ { unsigned long RequestID; Timestamp_t StartTime; Timestamp_t EndTime; int ErrorCode; char ErrorText[80]; } RecoveryDataError_t;
14 (Recovery Data Request)	typedef struct RecoveryDataRequest_s { unsigned long RequestID; int DataType; tMsgSequencer SequenceInfo; } RecoveryDataRequest_t;
15 (Challenge Response)	typedef struct ChallengeResponse_s { BYTE ChallengeID[32]; unsigned short TryCount; unsigned short IdentificationOffset; BYTE IVClient[8]; DWORD SessionKeyBlobSize; } ChallengeResponse_t;
16 (Challenge NACK)	typedef struct ChallengeNack_s { unsigned short TryCount; unsigned short IdentificationOffset; } ChallengeNack_t;

5.0 Acronyms

API	Application Programming Interface
CDR	Call Detail Record
CPE	Customer Provided Equipment
CPM	Call Progress Message
DES	Digital Encryption Standard
TCPIP	Transmission Control Protocol-Internet Protocol