QIO/SQIO API
Reference Guide

DC 900-1355E

Protogate, Inc.
12225 World Trade Drive, Suite R
San Diego, CA 92128
June 2009

Protogate, Inc.
12225 World Trade Drive, Suite R
San Diego, CA 92128
(858) 451-0865

QIO/SQIO API Reference Guide

Printed in the United States of America

Contents

List of Figures

List of Tables

Preface

Purpose of Document

This document describes the operation and programming interface required to use Protogate's QIO/SQIO product for the Freeway communications server. The information in this document was previously included as an appendix to the *Freeway Data Link Interface Reference Guide*.

Intended Audience

This document should be read by programmers who are interfacing a client application program to Protogate's QIO/SQIO application program interface (API). You should understand the Freeway data link interface (DLI), as explained in the *Freeway Data Link Interface Reference Guide*, the Freeway transport subsystem interface (TSI), as explained in the *Freeway Transport Subsystem Interface Reference Guide,* and your particular protocol programmer's guide (that is, the protocol for which Protogate's QIO/SQIO will convert the commands and responses to run on the Freeway server as described in Chapter 2).

Required Equipment

The QIO/SQIO product requires the following major hardware components to operate:

- a Freeway communications server that runs the communications software
- a client computer that runs the following:
    - TCP/IP

- Freeway DLI
- the user application program

Organization of Document

Chapter 1 is an overview of Protogate's QIO/SQIO product.

Chapter 2 discusses the various protocol converters available for Protogate's QIO/SQIO product.

Protogate References

The following documents provide useful supporting information, depending on the customer's particular hardware and software environments. Most documents are available on-line at Protogate's web site, www.protogate.com.

General Product Overviews

• *Freeway 1100 Technical Overview*	25-000-0419
• *Freeway 2000/4000/8800 Technical Overview*	25-000-0374
• *ICP2432 Technical Overview*	25-000-0420
• *ICP6000X Technical Overview*	25-000-0522

Hardware Support

• *Freeway 1100/1150 Hardware Installation Guide*	DC 900-1370
• *Freeway 2000/4000 Hardware Installation Guide*	DC 900-1331
• *Freeway 8800 Hardware Installation Guide*	DC 900-1553
• *Freeway 2000/4000 Hardware Maintenance Guide*	DC 900-1332
• *Freeway ICP6000R/ICP6000X Hardware Description*	DC 900-1020
• *ICP6000(X)/ICP9000(X) Hardware Description and Theory of Operation*	DC 900-0408
• *ICP2424 Hardware Description and Theory of Operation*	DC 900-1328
• *ICP2432 Hardware Description and Theory of Operation*	DC 900-1501
• *ICP2432 Hardware Installation Guide*	DC 900-1502

Freeway Software Installation Support

• *Freeway Server User's Guide*	DC 900-1333
• *Freeway Software Release Addendum: Client Platforms*	DC 900-1555
• *Getting Started with Freeway 1100/1150*	DC 900-1369
• *Getting Started with Freeway 2000/4000*	DC 900-1330
• *Getting Started with Freeway 8800*	DC 900-1552
• *Loopback Test Procedures*	DC 900-1533

Embedded ICP Installation and Programming Support

• *ICP2432 User's Guide for Digital UNIX*	DC 900-1513
• *ICP2432 User's Guide for OpenVMS Alpha*	DC 900-1511
• *ICP2432 User's Guide for OpenVMS Alpha (DLITE Interface)*	DC 900-1516
• *ICP2432 User's Guide for Windows NT*	DC 900-1510
• *ICP2432 User's Guide for Windows NT (DLITE Interface)*	DC 900-1514

Application Program Interface (API) Programming Support

• *Freeway Data Link Interface Reference Guide*	DC 900-1385
• *Freeway QIO/SQIO API Reference Guide*	DC 900-1355
• *Freeway Transport Subsystem Interface Reference Guide*	DC 900-1386

Socket Interface Programming Support

• *Freeway Client-Server Interface Control Document*	DC 900-1303

Toolkit Programming Support

• *Freeway Server-Resident Application and Server Toolkit Programmer's Guide*	DC 900-1325
• *OS/Impact Programmer's Guide*	DC 900-1030
• *Protocol Software Toolkit Programmer's Guide*	DC 900-1338

Protocol Support

• *ADCCP NRM Programmer's Guide*	DC 900-1317
• *Asynchronous Wire Service (AWS) Programmer's Guide*	DC 900-1324
• *Addendum: Embedded ICP2432 AWS Programmer's Guide*	DC 900-1557
• *BSC Programmer's Guide*	DC 900-1340

- *BSCDEMO User's Guide* DC 900-1349
- *BSCTRAN Programmer's Guide* DC 900-1406
- *DDCMP Programmer's Guide* DC 900-1343
- *FMP Programmer's Guide* DC 900-1339
- *Freeway AUTODIN Programmer's Guide* DC 908-1558
- *Freeway Marketfeed 2000 Programmer's Guide* DC 900-1346
- *Freeway MRS Protocol Programmer's Guide* DC 900-1565
- *Freeway SIO STD-1300 Programmer's Guide* DC 908-1559
- *Freeway SWIFT and CHIPS Programmer's Guide* DC 900-1344
- *Military/Government Protocols Programmer's Guide* DC 900-1602
- *SIO STD-1200A (Rev. 1) Programmer's Guide* DC 908-1359
- *X.25 Call Service API Guide* DC 900-1392
- *X.25/HDLC Configuration Guide* DC 900-1345
- *X.25 Low-Level Interface* DC 900-1307

Document Conventions

The term "Freeway" refers to any of the Freeway server models (for example, Freeway 500/3100/3200/3400 PCI-bus servers, Freeway 1000 ISA-bus servers, or Freeway 2000/4000/8800 VME-bus servers). References to "Freeway" also may apply to an embedded ICP product using DLITE (for example, the embedded ICP2432 using DLITE on a Windows NT system).

Physical "ports" on the ICPs are logically referred to as "links." However, since port and link numbers are usually identical (that is, port 0 is the same as link 0), this document uses the term "link."

Revision History

The revision history of the *QIO/SQIO API Reference Guide*, Protogate document DC 900-1355E, is recorded below:

Document Revision	Release Date	Description
DC 900-1355A	April 1996	Original release (the information in this document was previously an appendix to the *Freeway Data Link Interface Reference Guide*)
DC 900-1355B	July 1997	Add FW_SETAST function (Section 1.6 on page 28)
DC 900-1355C	October 1997	Changes to the following: • Section 1.2.2 on page 20 • Section 1.3.2 on page 24 • Section 2.7 on page 58 • Table 2–19 on page 60 (add HOPEN_SESSION command) • Section 2.7.6 on page 67
DC 900-1355D	January 1999	• Add I/O Status Block note on page 25. • Add Section 2.2.1 on page 34 and Section 2.4.1 on page 46
DC 900-1355E	June 2009	• Update contact information for Protogate, Inc. • Add HDLC-LAPB information.

Customer Support

If you are having trouble with any Protogate product, call us at (858) 451-0865 Monday through Friday between 8 a.m. and 5 p.m. Pacific time.

You can also fax your questions to us at (877) 473-0190 any time. Please include a cover sheet addressed to “Customer Service.”

We are always interested in suggestions for improving our products. You can use the report form in the back of this manual to send us your recommendations.

Chapter

1 QIO/SQIO Overview

Note

SQIO is currently available only for VMS. Future Freeway releases will make SQIO available on all Protogate-supported clients.

This document assumes an understanding of the DLI and TSI configuration process and the configuration parameters as described in the following references:

- The *Freeway Data Link Interface Reference Guide* describes Protogate's DLI configuration process and the configuration files required for the SQIO calls
- The *Freeway Transport Subsystem Interface Reference Guide* describes Protogate's TSI configuration process and the configuration files required for the SQIO calls

The following additional references will be helpful:

- Both the ICP and the Freeway versions of the protocol programmer's guide give the details of the commands and responses that are converted by the QIO/SQIO
- The *ICP3222 VMS Host Driver Programmer's Guide*, DC 900-0419, describes Protogate's QIO ICP driver for an ICP3222
- The VAX VMS Systems Services manual provides a guide to VMS I/O

Users familiar with VAX VMS QIO calls and the ICP3222 version of a Simpact supported communications protocol, might wish to skip to Chapter 2 for the SQIO protocol converter section that discusses that particular protocol.

1.1 Introduction

A number of Simpact VMS customers using VAX-based Simpact ICPs desire to migrate to Freeway in either a VAX or Alpha environment. However, conversion from VMS Queued I/O (QIO) calls within the user application (called a VMS process) requires some reprogramming to use the Freeway data link interface (DLI). This transition can be complex. In addition, new Freeway users sometimes find the concepts of VMS QIO easier to implement than those supported by the Freeway DLI.

To meet the needs of these customers, Protogate provides an application program interface (API) to the Freeway DLI that is, as much as possible, identical to VMS QIO calls. This appendix first shows how normal VMS QIO system calls are made to initiate ICP I/O, and then presents the comparable Protogate Queued I/O (SQIO) call. This not only guides new users on how to interface more easily to Freeway, but also shows how to convert existing VMS QIO applications to Freeway by simply changing the name of the routines that initiate ICP I/O.

Simpact previously implemented a set of QIO drivers to interface to ICPs on various VMS platforms. These QIO drivers allow reads and writes (with or without wait), setting of an Event Flag, and/or the invocation of an Asynchronous System Trap (AST) with a possible AST parameter using standard VMS services. References to QIO within this document refer to Simpact's implementation of these ICP drivers. References to SQIO within this document refer to Protogate's implementation of the Freeway API that is compatible with the VMS QIO driver implementation.

1.1.1 QIO

The normal sequence of events for a VMS application (or process) to interface to a Simpact ICP device (or almost any device) is to have VMS assign a logical variable, called a channel, to the ICP. This assignment is accomplished using the VMS SYS$ASSIGN system service routine. Multiple ICPs can be assigned to multiple channels within a process. Subsequently, all reads and writes are done to the ICP using SYS$QIOW or SYS$QIO (with or without wait). At the completion of the process, the

device associated with a channel is deassigned using SYS$DASSGN. In addition, a process can cancel all reads or writes associated with a channel by calling SYS$CANCEL.

The VMS QIO driver uses the communication channel and process node numbers to exchange messages between processes running in the VMS client system and a task running on the Simpact ICP. The communication channel number identifies the ICP device with which the VMS QIO driver is communicating. A process node number specifies the intended recipient of a message sent over the channel in either direction. Both the host and the ICP have node numbers. ICP node numbers are protocol specific, depending on the protocol that is to be addressed on an ICP. ICP node numbers usually identify a link, or a set of links, on the ICP. Host node numbers are usually unique to a process running under VMS.

When a VMS process wants to write a message to an ICP, it issues a SYS$QIO(W) write request to the appropriate channel. The request includes a parameter that specifies both the node number of the writing process and the node number of the intended reader on the ICP.

When a VMS process wants to read a message from an ICP, it issues a SYS$QIO(W) read request to the appropriate channel. The request includes a parameter that specifies the node number of the intended reader on the host.

1.1.2 SQIO

The normal sequence of events for an SQIO client application to interface to a Protogate Freeway device is to assign a logical variable, called a communication channel, to the Freeway device. A Freeway device is a set of Freeway sessions defined by an SQIO configuration file. Multiple Freeway devices (or sets of sessions) can be assigned to multiple channels within a client process. The client application does this assignment using the SQIO FW_ASSIGN routine. Subsequently, all reads and writes are done to the Freeway device using FW_QIOW or FW_QIO (with or without wait). At the completion of the process, the FW_DASSGN routine deassigns the communication channel. In addition, the client application can cancel all reads or writes associated with a channel by calling FW_CANCEL.

SQIO uses the communication channel and node numbers to exchange messages between an application running in the client system and a task running on a Freeway ICP associated with a session. The communication channel number identifies the Freeway sessions with which the driver is communicating. A node number specifies the session of the intended recipient of a message sent over the channel. Since an ICP node number specifies the session number within a channel, the ICP node number usually identifies a link, or a set of links, on a Freeway ICP.

When a client application wants to write a message to a Freeway ICP session, it issues a FW_QIO(W) write request to the appropriate channel associated with the session. The request includes a parameter that specifies the node number of the intended reader on the Freeway ICP, which identifies a particular Freeway ICP session.

When a client application wants to read a message from a Freeway ICP session, it issues a FW_QIO(W) read request to the appropriate channel associated with the session. The request includes a parameter that specifies the node number of the intended reader on the Freeway ICP, which identifies a particular Freeway ICP session.

1.2 Initiating an Application

1.2.1 QIO (SYS$ASSIGN)

To issue VMS QIO calls, the application must first assign a channel to an ICP with the VMS SYS$ASSIGN call. This call sets up an association between this channel and the ICP on subsequent QIO calls. See the VAX VMS system services manual for a detailed description of SYS$ASSIGN.

Synopsis

SYS$ASSIGN (device_name, &channel [,acmode] [,mbxnam])

Parameters

device_name	ICP device name (for example, ZOA0)
channel	The address of the communication channel that is assigned

Note

Access mode (acmode) and mailbox (mbxnam) are not supported by SQIO.

1.2.2 SQIO (FW_ASSIGN)

To assign a channel to an ICP, the client application issues the SQIO FW_ASSIGN call. This call sets up an association between a logical channel and the Freeway sessions associated with an SQIO configuration file. The logical channel, or communication channel, is used on subsequent calls to SQIO.

SQIO initiates the DLI (dlInit), opens each session (dlOpen) defined by the SQIO configuration file, and associates the communication channel to this set of sessions. Subsequent association of channels to sessions (and associated links on a Freeway ICP) is done by a protocol converter program at the time of SQIO read and write calls (see the parameter descriptions below).

Synopsis

FW_ASSIGN (device_name, &channel)

Parameters

device_name	The name of the SQIO configuration file (for example, ZOA0)
channel	The address of the communication channel that is assigned.

The SQIO configuration file describes the following: the name of the DLI *binary* configuration file, the associated protocol converter name, and the session names to be opened within this configuration. The text format of the file is:

```
DLI binary configuration file name, protocol converter name,
(session name [,session name] [.....])
```

An example SQIO configuration file might contain the following string:

```
dlicfg.bin, BSC3270, (B0L0, B0L1, B1L0, B1L1)
```

Note

Continue a line in the SQIO configuration file with a "–" (dash/minus)

Refer to the *Freeway Data Link Interface Reference Guide* for a description of the DLI configuration process and an example DLI configuration file. The list of session names in the SQIO configuration file are those which are defined in the specified DLI configuration file.

Multiple SQIO configuration files can exist in the SQIO environment. Each SQIO configuration can have a different logical channel assigned using FW_ASSIGN. However, each SQIO configuration file must specify the same DLI binary configuration file name. In addition, each DLI session must be unique (meaning that it is specified only once) among the set of SQIO configuration files.

Caution

To use SQIO calls, several DLI and TSI configuration parameters require specific settings, as shown in Table 1–1. The DLI configuration file is described in the *Freeway Data Link Interface Reference Guide*. The TSI configuration file is described in the *Freeway Transport Subsystem Interface Reference Guide*.

The protocol converter name in the SQIO configuration file is used by SQIO to invoke the proper protocol converter. If no protocol converter is necessary, which is the case when the client application uses SQIO as a protocol-independent API, the protocol converter name in the SQIO configuration file specifies SQIO, and the client application must supply an SQIO header for all SQIO read and write requests (see Chapter 2 for more information on protocol converters and the SQIO header).

Table 1–1: Required DLI and TSI Configuration Parameters for SQIO

Configuration Parameter	DLI or TSI	Required Setting in Configuration File
alwaysQIO	DLI	"yes" (each session description)
asyncIO	DLI	"yes" (main section plus each session description)
mode	DLI	protocol-dependent (required for X.25, BSC 3270, BSC 3780, FMP, SWIFT, and CHIPS); for example, mode = "mgr"
protocol	DLI	"raw" (each session description)
asyncIO	TSI	"yes" (main section plus each connection description)
maxBufSize	TSI	The buffer sizes used for reads and writes should be at least 100 bytes less than this parameter (the default is 1024)

1.3 Read and Write Requests

1.3.1 QIO (SYS$QIO(W))

To issue VMS read or write I/O calls, the client application issues the VMS SYS$QIOW or SYS$QIO calls (for I/O with, or without wait). See the VAX VMS system services manual for a detailed description of SYS$QIOW and SYS$QIO.

Synopsis

```
SYS$QIO(W) ( [efn], channel, function [,&iosb] [,&astadr] [,astprm],
                    &buf_adr, buf_len, NULL, node )
```

Parameters

efn	Event flag to be set on completion of I/O
channel	Communication channel associated with a device
function	Standard VMS read or write function code
iosb	Address of the I/O Status Block fields to receive the I/O completion status
astadr	Address of an Asynchronous System Trap (AST) routine to be executed on I/O completion
astprm	AST parameter passed to the AST routine on I/O completion
buf_adr	Buffer address
buf_len	Buffer length of the write buffer or maximum read buffer size
node	ICP node number. The node number can have different interpretations for individual protocols. For example, some protocols have a unique node number for each link on the ICP, and some protocols use a single node number for all links on the ICP.

1.3.2 SQIO (FW_QIOW or FW_QIO)

To issue SQIO read or write I/O calls, the client application issues the SQIO FW_QIOW or FW_QIO calls (for I/O with, or without wait). These routines have identical parameters and functionality as the VMS SYS$QIOW and SYS$QIO calls, respectively.

Synopsis

FW_QIO(W) ([efn], channel, function [,&iosb] [,&astadr] [,astprm], &buf_adr,
buf_len, NULL, node)

Parameters

efn — Event flag to be set on completion of I/O

channel — Communication channel associated with a device (set of sessions)

function — One of the standard VMS read or write function codes:
IO$_READVBLK, IO$_READLBLK, IO$_READPBLK
IO$_WRITEVBLK, IO$_WRITELBLK, or IO$_WRITEPBLK

iosb — Address of the I/O Status Block fields to receive the I/O completion status

astadr — Address of an Asynchronous System Trap (AST) routine to be executed on I/O completion

astprm — AST parameter passed to the AST routine on I/O completion

buf_adr — Buffer address

buf_len — Buffer length of the write buffer or maximum read buffer size. The buf_len parameter must include the length of the protocol header and the data.

node — ICP node number. The node number can have different interpretations for individual protocols. For example, some protocols have a

unique node number for each link on the ICP, and some protocols use a single node number for all links on the ICP. SQIO node numbers are mapped to Freeway sessions depending on the SQIO configuration file protocol converter name, and functions of the protocol converter (see Chapter 2).

If the client application is operating in protocol-independent mode (the protocol converter name in the SQIO configuration file is specified as SQIO), then the node number is mapped to a Freeway session by using the node number as a direct index into the SQIO configuration file session list. Node number one indicates that the session name is the first session specified in the session list, etc.

Note

The I/O Status Block contains a standard VMS error status in the status field. In addition, the device-dependent field of the I/O Status Block contains the DLI dlerrno field on error returns from DLI calls. Refer to the *Freeway Data Link Interface Reference Guide* for a description of the dlerrno return status on DLI calls.

1.4 Canceling Read and Write Requests

1.4.1 QIO (SYS$CANCEL)

To cancel all active or pending read or write requests associated with an I/O channel, the application issues the VMS SYS$CANCEL call. See the VAX VMS system services manual for a detailed description.

Synopsis

SYS$CANCEL (channel)

Parameters

channel Communication channel

1.4.2 SQIO (FW_CANCEL)

To cancel all active or pending Freeway read and write calls associated with a Freeway communication channel, the client application issues the SQIO FW_CANCEL call. This call affects all the configured sessions for that channel.

Synopsis

FW_CANCEL (channel)

Parameters

channel Communication channel

1.5 Terminating an Application

1.5.1 QIO (SYS$DASSGN)

To terminate its association with an ICP device, the application issues the VMS SYS$DASSGN call for the channel associated with the ICP. See the VAX VMS system services manual for a detailed description of SYS$DASSGN.

Synopsis

SYS$DASSGN (channel)

Parameters

channel Communication channel

1.5.2 SQIO (FW_DASSGN)

To terminate its association with a Freeway ICP, the client application issues the SQIO FW_DASSGN call. SQIO cancels any Freeway read or write requests in progress, clears any queued read or write requests for each session associated with the communication channel, closes each session (dlClose), and terminates Freeway (dlTerm).

Synopsis

FW_DASSGN (channel)

Parameters

channel Communication channel

1.6 Protecting Critical Code

1.6.1 QIO (SYS$SETAST)

To protect critical code sections from AST software interrupts, the application typically uses the SYS$SETAST service as illustrated in the "C" code example shown below. See the VAX VMS system services manual for a detailed description.

```
int status;
status = SYS$SETAST (0);

... critical code goes here ...

if (status == SS$_WASSET)
        SYS$SETAST (1);
```

Synopsis

SYS$SETAST (flag)

Parameters

flag	TRUE to enable ASTs; FALSE to disable ASTs

1.6.2 SQIO (FW_SETAST)

To protect critical code sections from AST software interrupts when using the QIO/SQIO API, the application typically avoids the native SYS$SETAST service, and uses the FW_SETAST service as illustrated in the "C" code example shown below.

```
int status;
status = FW_SETAST (0);

... critical code goes here ...

if (status == SS$_WASSET)
        FW_SETAST (1);
```

Synopsis

FW_SETAST (flag)

Parameters

flag	TRUE to enable ASTs; FALSE to disable ASTs

Chapter

2 SQIO Protocol Converters

A protocol converter is a functional module provided by Protogate within SQIO which is responsible for mapping protocol-specific functions from QIO functionality into SQIO Freeway protocol-dependent functionality. The primary function of the protocol converter is to convert a client QIO write request header into an SQIO header (with optional arguments), and to convert an SQIO read complete header (with optional arguments) into a client QIO read completion header. The protocol converter attempts to simulate, as much as possible, a "seamless" programming environment between the ICP version of a protocol, and the Freeway version of that protocol.

SQIO uses a configuration file to specify operation. The SQIO configuration file defines the DLI binary configuration file name, the protocol converter name to be used, and a list of sessions (defined within the DLI configuration file). Refer back to Table 1–1 on page 22 for SQIO configuration requirements.

The general form of the SQIO configuration file is as follows:

> DLI binary configuration file name, protocol converter name, (session list)
>
> where the protocol converter name is as defined in the following subsections, and (session list) is a list of session names defined within the DLI configuration file, separated by commas; for example, (B0L0, B0L1)

An example SQIO configuration file that accesses the control and data nodes for BSC 3270 links 0 and 1 could be named ZOA1 and contain the following string:

> bscconfig.bin, BSC3270, (B0L0, B0L1, B0C0, B0C1)

An example of an X.25 SQIO configuration file to talk only to link 1 of ICP 0 could be named ZOA1 and contain the following string:

```
x25config.bin, X25, (B0L1)
```

Note

Be sure to include only those session names (from the DLI binary configuration file) of the link(s) your application will actually be accessing. This allows other applications to access the other links specified by the DLI binary configuration file.

The following sections detail the specifics of each SQIO supported protocol, including the protocol converter name and the relationship between session names (and their relative position within the session list) and the SQIO node parameter.

2.1 Generic Protocol Converter

The Generic (protocol-independent) protocol converter allows Freeway users that are familiar with VMS QIO calls, to interface with a standard Freeway protocol without having to learn the standard Freeway DLI interface. The Generic protocol converter name is SQIO.

The node parameter in an SQIO call maps directly to the relative session name within the session list. That is, an SQIO write or read to node 3 is directed to the third relative session name within the session list. When using the Generic protocol converter, the client application supplies an SQIO header as part of the data buffer for all SQIO read and write requests. The SQIO header structure is defined in the dliusr.h include file as shown in Figure 2–1.

The Generic protocol converter performs the mapping between the SQIO header and the DLI optional arguments used in DLI *Raw* read and write operations, as shown in

```
typedef structure
{
        unsigned short      usFunction;     /* Protocol function code          */
        unsigned short      usLink;         /* Link (port) number              */
        unsigned short      usStatus;       /* Protocol status                 */
        short               iModifier;      /* Protocol modifier               */
        unsigned short      usCircuit;      /* Protocol circuit                */
        unsigned short      usSequence;     /* Protocol sequence               */
        unsigned short      usParm1;        /* Protocol parameter 1            */
        unsigned short      usParm2;        /* Protocol parameter 2            */
        short               iDataLength;    /* Protocol data length            */
                                            /* (minus this header size)        */
}       SQIO_HEADER;
```

Figure 2–1: SQIO Header "C" Structure

Table 2–1. In addition, the SQIO header iDataLength field is set to the length of data returned in the Freeway buffer (on read complete).

Table 2–1: SQIO Header Mapping to DLI Optional Arguments

SQIO Header	DLI Optional Arguments
usFunction	pOptArgs.usProtCommand
usLink	pOptArgs.usProtLinkID
usStatus	pOptArgs.usICPStatus
iModifier	pOptArgs.iProtModifier
usCircuit	pOptArgs.usProtCircuitID
usSequence	pOptArgs.usProtSequence
usParm1	pOptArgs.usProtXParms[0]
usParm2	pOptArgs.usProtXParms[1]

The Generic protocol converter can be used for any Freeway protocol, including those described in the following sections, as long as the calling application understands the mapping that should take place into the DLI optional arguments. Customized toolkit protocols will find the Generic protocol converter useful if the application programmer is more familiar or comfortable with VMS QIO calls than with DLI calls.

2.2 BSC 3780 Protocol Converter

2.2.1 Differences Between ICP and Freeway Versions of BSC 3780

The ICP version of BSC 3780 allows data acknowledge nodes, also referred to as Dack nodes. These nodes allow two separate VMS applications to transmit and receive data on the same link. When the data acknowledge node option is *enabled*, all acknowledgments from transmitted data are routed through the Dack node (link number plus 21) rather than through the normal data node (link number plus 1). The transmitting application writes data to the ICP on the data node and reads data acknowledgments from the Dack node. The receiving application reads incoming data through the normal data node.

The Freeway version of BSC 3780 (and thus SQIO) is slightly different in its handling of data acknowledgments and sending incoming data to a client application. Data acknowledgments are returned on *Manager* sessions, and if *Read* sessions are defined, incoming data is returned on the *Read* sessions. Therefore, if there are two separate VMS SQIO applications to transmit and receive data on the same link, the application that writes data defines only *Manager* sessions for the links on which it will be sending data. This application writes transmit data on the normal data node and reads data acknowledgments from the Dack node or the normal data node. The receiving application reads incoming data through the normal data node.

For Freeway, if the data acknowledge node option is *disabled*, incoming data is sent to the VMS SQIO write application until, and if, a read application is started. If the data acknowledge node option is *enabled*, incoming data is discarded until, and if, a read application is started. It is therefore best that the read application be started before the write application enables the affected lines.

2.2.2 BSC 3780 Protocol Converter Description

The BSC 3780 protocol converter attempts to simulate, as much as possible, a "seamless" programming environment between the ICP version of the BSC 3780 protocol,

and the Freeway version of the protocol. The BSC 3780 protocol converter name is BSC3780.

The BSC 3780 protocol converter discards the following responses to commands that exist in the Freeway version of this protocol, but not in the ICP version:

- Link Configuration Confirmation
- Set Buffer Size Confirmation

The client application supplies an ICP header as part of the data buffer for all SQIO read and write requests. The BSC 3780 protocol converter performs the mapping between the ICP header and the DLI optional arguments used in DLI *Raw* read and write operations, as shown in Table 2–2.

Table 2–2: ICP Header Mapping to DLI Optional Arguments (BSC 3780)

ICP Header	DLI Optional Arguments
function[1]	pOptArgs.usProtCommand
link	pOptArgs.usProtLinkID
error_status	pOptArgs.usICPStatus
modifier	pOptArgs.iProtModifier
sequence	pOptArgs.usProtCircuitID

[1] The BSC 3780 protocol converter is responsible for converting the ICP function numbers into the Freeway command numbers, and *vice versa*.

Table 2–3 shows additional DLI optional arguments mapping performed by the BSC 3780 protocol converter.

Table 2–3: Additional Optional Arguments Mapping (BSC 3780)

DLI Optional Arguments	Mapping
pOptArgs.iProtModifier	This field is set to the mode parameter in the DLI configuration file. Valid modes are "mgr" for *Manager* sessions, "control" for *Control* sessions, "read" for *Dack* sessions, and "trace" for *Trace* sessions.
pOptArgs.usProtSessionID	This field is set to the Freeway session ID.
pOptArgs.usProtSequence	This field is set to 0 (zero).
pOptArgs.usProtXParms[0]	This field is set to 5 (3780 protocol value)
pOptArgs.usProtXParms[1]	This field is set to 0 (zero).

A BSC 3780 SQIO application can access links on an ICP through *Manager* (data) nodes, *Control* nodes, *Dack* nodes, or *Trace* nodes. The link is determined by the node parameter in the FW_QIO(W) call (Section 1.3.2 on page 24) which is then mapped to a particular session defined in the DLI configuration file. Table 2–4 summarizes the session mapping provided by the protocol converter. *Manager* and *Dack* nodes are mapped to sessions by first determining the link associated with the FW_QIO(W) node parameter (see Table 2–4), and then finding the matching portNo DLI configuration parameter within the appropriate *Manager* or *Dack* session definitions (see Figure 2–2). *Control* and *Trace* nodes are mapped to sessions using the link field in the ICP header (Table 2–2) to find the matching portNo DLI configuration parameter within the appropriate *Control* or *Trace* session definitions (see Figure 2–2).

Each BSC 3780 session must specify mode = "mgr", mode = "control", mode = "read", or mode = "trace" in the DLI configuration file. The session name associated with each node/link in Figure 2–2 is of the format B0L0…B0L(*n*–1) for *Manager* nodes, B0C0…B0C(*n*–1) for *Control* nodes, B0D0…B0D(*n*–1) for *Dack* (read) nodes, and B0T0…B0T(*n*–1) for *Trace* nodes, where *n* = the maximum number of links.

The following example illustrates the SQIO session mapping:

A BSC 3780 SQIO configuration file, which could be named ZOA1, might contain the following string:

```
bscconfig.bin, BSC3780, (B0L1, B0C1, B0D1, B0T1)
```

Each of the session names (B0L1, B0C1, B0D1, and B0T1) is defined in the bscconfig DLI configuration file (similar to Figure 2–2). To write to each of these sessions, your SQIO application would do the following (also see Table 2–4):

B0L1 Call FW_QIO(W) with the node parameter = 2. The protocol converter then finds the matching session in the bscconfig file with mode = "mgr" and portNo = 1.

B0C1 Call FW_QIO(W) with the node parameter = 41 and the ICP header link field = 1. The protocol converter then finds the matching session in the bscconfig file with mode = "control" and portNo = 1.

B0D1 Call FW_QIO(W) with the node parameter = 22. The protocol converter then finds the matching session in the bscconfig file with mode = "read" and portNo = 1.

B0T1 Call FW_QIO(W) with the node parameter = 42 and the ICP header link field = 1. The protocol converter then finds the matching session in the bscconfig file with mode = "trace" and portNo = 1.

A simpler BSC 3780 SQIO configuration file to use link 0 and link 1 as *Manager* (data) nodes, allowing another application to use the *Control* node, would be:

```
bscconfig.bin, BSC3780, (B0L0, B0L1)
```

Note

The session list in your SQIO configuration file should include only those sessions defined in the DLI configuration file that will actually be accessed by your application.

Table 2–4: Session Mapping (BSC 3780)

Link# Associated with Session Name[1]	Manager Node (mode="mgr")	Dack Node (mode="read")	Control Node[2] (mode="control")	Trace Node[b] (mode="trace")
0	1	21	41	42
1	2	22	41	42
⋮	⋮	⋮	⋮	⋮
n–1	*n*	*n*+20	41	42

[1] The session name associated with each node/link in Figure 2–2 is of the format B0L0…B0L(*n*–1) for *Manager* nodes, B0C0…B0C(*n*–1) for *Control* nodes, B0D0…B0D(*n*–1) for *Dack* nodes, and B0T0…B0T(*n*–1) for *Trace* nodes, where *n* = the maximum number of links.

[2] For writes, the application ICP header link field determines the session. For reads, all *Control* or *Trace* session reads complete to node 41 or 42 respectively.

```
//--------------------------------------------------------------------- //
// DLI configuration file for BSCDEMO/BSCTEST                          //
//                                                                     //
// Define a main section and a section for each ICP link (port). The   //
// defaults of most of the main section parameters are used, so        //
// the parameters do not show up in this file. See the DLI reference   //
// guide for a definition of all DLI configuration parameters.         //
//                                                                     //
// Date:      Initials:      Description:                              //
//                                                                     //
//--------------------------------------------------------------------- //

//--------------------------------------------------------------------- //
//                                                                     //
// "main" section. If not defined defaults are used. If present        //
// the main section must be the very first section of the DLI          //
// configuration file.                                                 //
//                                                                     //
//--------------------------------------------------------------------- //
main                                    // Non-session-specific configs  //
{
    AsyncIO = "Yes";                    // Non-blocking I/O              //
    TSICfgName = "bscqtcfg.bin";        // TSI configuration file name   //
}
```

Figure 2–2: DLI Configuration File Example for BSC 3780 SQIO

```
//-------------------------------------------------------------------- //
// Define a Manager session for four links. Freeway server name       //
// is defined in the TSI configuration file used.                     //
//-------------------------------------------------------------------- //
B0L0                                  // First Manager session name:  //
{
    Protocol = "RAW";                 // RAW session type             //
    Transport = "Conn0";              // Transport connection name    //
                                      //    defined in TSICfgName file //
    BoardNo = 0;                      // ICP board number -- based 0  //
    PortNo = 0;                       // Link number                  //
    Mode = "mgr";
    AlwaysQIO = "Yes";
    AsyncIO = "Yes";                  // Non-blocking I/O             //
}
B0L1                                  // Second Manager session name: //
{
    Protocol = "RAW";                 // RAW session type             //
    Transport = "Conn0";              // Transport connection name    //
                                      //    defined in TSICfgName file //
    BoardNo = 0;                      // ICP board number -- based 0  //
    PortNo = 1;                       // Link number                  //
    Mode = "mgr";
    AlwaysQIO = "Yes";
    AsyncIO = "Yes";                  // Non-blocking I/O             //
}
B0L2                                  // Third Manager session name:  //
{
    Protocol = "RAW";                 // RAW session type             //
    Transport = "Conn0";              // Transport connection name    //
                                      //    defined in TSICfgName file //
    BoardNo = 0;                      // ICP board number -- based 0  //
    PortNo = 2;                       // Link number                  //
    Mode = "mgr";
    AlwaysQIO = "Yes";
    AsyncIO = "Yes";                  // Non-blocking I/O             //
}
```

Figure 2–2: DLI Configuration File Example for BSC 3780 SQIO *(Cont'd)*

```
B0L3                                    // Fourth Manager session name:      //
{
    Protocol = "RAW";                   // RAW session type                  //
    Transport = "Conn0";                // Transport connection name         //
                                        //    defined in TSICfgName file     //
    BoardNo = 0;                        // ICP board number -- based 0       //
    PortNo = 3;                         // Link number                       //
    Mode = "mgr";
    AlwaysQIO = "Yes";
    AsyncIO = "Yes";                    // Non-blocking I/O                  //
}
//-------------------------------------------------------------------------   //
// Define a Control session for 4 links.                                     //
//-------------------------------------------------------------------------   //
B0C0                                    // First Control session name:       //
{
    Protocol = "RAW";                   // RAW session type                  //
    Transport = "Conn0";                // Transport connection name         //
                                        //    defined in TSICfgName file     //
    BoardNo = 0;                        // ICP board number -- based 0       //
    PortNo = 0;                         // Link number                       //
    AlwaysQIO = "Yes";
    AsyncIO = "Yes";                    // Non-blocking I/O                  //
    mode = "control";                   // Control Mode                      //
}

    .
    .                   // Control sessions B0C1 through B0C3                 //
    .
```

Figure 2–2: DLI Configuration File Example for BSC 3780 SQIO *(Cont'd)*

```
//-------------------------------------------------------------------- //
// Define a Data ACK node session for 4 links.                        //
//-------------------------------------------------------------------- //
B0D0                                    // First Dack session name:    //
{
    Protocol = "RAW";                   // RAW session type            //
    Transport = "Conn0";                // Transport connection name   //
                                        //    defined in TSICfgName file //
    BoardNo = 0;                        // ICP board number -- based 0 //
    PortNo = 0;                         // Link number                 //
    AlwaysQIO = "Yes";
    AsyncIO = "Yes";                    // Non-blocking I/O            //
    mode = "read";                      // Read Mode                   //
}

        .
        .                   // Dack sessions B0D1 through B0D3          //
        .


//-------------------------------------------------------------------- //
// Define a Trace node session for 4 links.                           //
//-------------------------------------------------------------------- //
B0T0                                    // First Trace session name:   //
{
    Protocol = "RAW";                   // RAW session type            //
    Transport = "Conn0";                // Transport connection name   //
                                        //    defined in TSICfgName file //
    BoardNo = 0;                        // ICP board number -- based 0 //
    PortNo = 0;                         // Link number                 //
    AlwaysQIO = "Yes";
    AsyncIO = "Yes";                    // Non-blocking I/O            //
    mode = "trace";                     // Trace Mode                  //
}

        .
        .                   // Trace sessions B0T1 through B0T3         //
        .
```

Figure 2–2: DLI Configuration File Example for BSC 3780 SQIO *(Cont'd)*

2.3 BSC 3270 Protocol Converter

The BSC 3270 protocol converter attempts to simulate, as much as possible, a "seamless" programming environment between the ICP version of the BSC 3270 protocol, and the Freeway version of the protocol. The BSC 3270 protocol converter name is BSC3270.

The BSC 3270 protocol converter discards the following responses to commands that exist in the Freeway version of this protocol, but not in the ICP version:

- Link Configuration Confirmation
- Set Buffer Size Confirmation
- Set Polling List Confirmation

The client application supplies an ICP header as part of the data buffer for all SQIO read and write requests. The BSC 3270 protocol converter performs the mapping between the ICP header and the DLI optional arguments used in DLI *Raw* read and write operations, as shown in Table 2–5.

Table 2–5: ICP Header Mapping to DLI Optional Arguments (BSC 3270)

ICP Header	DLI Optional Arguments
function[1]	pOptArgs.usProtCommand
link	pOptArgs.usProtLinkID
error_status	pOptArgs.usICPStatus
modifier	pOptArgs.iProtModifier
(CU << 8) OR'ed with DU	pOptArgs.usProtCircuitID

[1] The BSC 3270 protocol converter is responsible for converting the ICP function numbers into the Freeway command numbers, and *vice versa*.

Table 2–6 shows additional DLI optional arguments mapping.

Table 2–6: Additional Optional Arguments Mapping (BSC 3270)

DLI Optional Arguments	Mapping
pOptArgs.iProtModifier	This field is set to the mode parameter in the DLI configuration file. Valid modes are "mgr" for *Manager* (data) sessions, and "control" for *Control* sessions.
pOptArgs.usProtSessionID	This field is set to the Freeway session ID.
pOptArgs.usProtSequence	This field is set to 0 (zero).
pOptArgs.usProtXParms[0]	This field is set to 4 (3270 protocol value)
pOptArgs.usProtXParms[1]	This field is set to 0 (zero).

A BSC 3270 application can access links on an ICP through *Manager* (data) nodes or *Control* nodes. The link is determined by the node parameter in the FW_QIO(W) call (Section 1.3.2 on page 24) which is then mapped to a particular session defined in the DLI configuration file. Table 2–7 summarizes the session mapping provided by the protocol converter. *Manager* nodes are mapped to sessions by first determining the link associated with the FW_QIO(W) node parameter (see Table 2–7), and then finding the matching portNo DLI configuration parameter within the appropriate *Manager* session definitions (refer back to Figure 2–2 on page 38). *Control* nodes are mapped to sessions using the link field in the ICP header (Table 2–5) to find the matching portNo DLI configuration parameter within the appropriate *Control* session definitions (refer back to Figure 2–2 on page 38).

Each BSC 3270 relative session must specify mode = "mgr" or mode = "control" in the DLI configuration file. The session name associated with each node/link in Figure 2–2 on page 38 is of the format B0L0…B0L(*n*–1) for *Manager* nodes and B0C0…B0C(*n*–1) for *Control* nodes, where *n* = the maximum number of links.

The following example illustrates the SQIO session mapping:

Table 2–7: Session Mapping (BSC 3270)

Link# Associated with Session Name[1]	Manager Node (mode="mgr")	Control Node[2] (mode="control")
0	1	41
1	2	41
⋮	⋮	⋮
n–1	*n*	41

[1] The session name associated with each node/link in Figure 2–2 on page 38 is of the format B0L0…B0L(*n*–1) for *Manager* nodes and B0C0…B0C(*n*–1) for *Control* nodes, where *n* = the maximum number of links.

[2] For writes, the application ICP header link field determines the session. For reads, all *Control* session reads complete to node 41.

A BSC 3270 SQIO configuration file, which could be named ZOA1, might contain the following string:

```
bscconfig.bin, BSC3270, (B0L1, B0C1)
```

Each of the session names (B0L1 and B0C1) is defined in the bscconfig DLI configuration file (similar to Figure 2–2 on page 38). To write to each of these sessions, your SQIO application would do the following (also see Table 2–7):

B0L1 Call FW_QIO(W) with the node parameter = 2. The protocol converter then finds the matching session in the bscconfig file with mode = "mgr" and portNo = 1.

B0C1 Call FW_QIO(W) with the node parameter = 41 and the ICP header link field = 1. The protocol converter then finds the matching session in the bscconfig file with mode = "control" and portNo = 1.

A simpler BSC 3270 SQIO configuration file to use link 0 and link 1 as *Manager* (data) nodes, allowing another application to use the *Control* node, would be:

```
bscconfig.bin, BSC3270, (B0L0, B0L1)
```

Note

The session list in your SQIO configuration file should include only those sessions defined in the DLI configuration file that will actually be accessed by your application.

2.4 FMP Protocol Converter

2.4.1 Differences Between ICP and Freeway Versions of FMP

The ICP version of FMP allows data acknowledge nodes, also referred to as Dack nodes. These nodes allow two separate VMS applications to transmit and receive data on the same link. When the data acknowledge node option is *enabled*, all acknowledgments from transmitted data are routed through the Dack node (link number plus 21) rather than through the normal data node (link number plus 1). The transmitting application writes data to the ICP on the data node and reads data acknowledgments from the Dack node. The receiving application reads incoming data through the normal data node.

The Freeway version of FMP (and thus SQIO) is slightly different in its handling of data acknowledgments and sending incoming data to a client application. Data acknowledgments are returned on *Manager* sessions, and if *Read* sessions are defined, incoming data is returned on the *Read* sessions. Therefore, if there are two separate VMS SQIO applications to transmit and receive data on the same link, the application that writes data defines only *Manager* sessions for the links on which it will be sending data. This application writes transmit data on the normal data node and reads data acknowledgments from the Dack node or the normal data node. The receiving application reads incoming data through the normal data node.

For Freeway, if the data acknowledge node option is *disabled*, incoming data is sent to the VMS SQIO write application until, and if, a read application is started. If the data acknowledge node option is *enabled*, incoming data is discarded until, and if, a read application is started. It is therefore best that the read application be started before the write application enables the affected lines.

2.4.2 FMP Protocol Converter Description

The FMP protocol converter attempts to simulate, as much as possible, a “seamless” programming environment between the ICP version of the FMP protocol, and the Freeway version of the protocol. The FMP protocol converter name is FMP.

The FMP protocol converter discards the following command response that exists in the Freeway version of this protocol, but not in the ICP version:

- Link Configuration Confirmation

The client application supplies an ICP header as part of the data buffer for all SQIO read and write requests. The FMP protocol converter performs the mapping between the ICP header and the DLI optional arguments used in DLI *Raw* read and write operations, as shown in Table 2–8.

Table 2–8: ICP Header Mapping to DLI Optional Arguments (FMP)

ICP Header	DLI Optional Arguments
function[1]	pOptArgs.usProtCommand
link	pOptArgs.usProtLinkID
error_status	pOptArgs.usICPStatus
modifier	pOptArgs.iProtModifier
sequence	pOptArgs.usProtCircuitID

[1] The FMP protocol converter is responsible for converting the ICP function numbers into the Freeway command numbers, and *vice versa*.

Table 2–9 shows additional DLI optional arguments mapping.

Table 2–9: Additional Optional Arguments Mapping (FMP)

DLI Optional Arguments	Mapping
pOptArgs.iProtModifier	This field is set to the mode parameter in the DLI configuration file. Valid modes are "mgr" for *Manager* sessions, "control" for *Control* sessions, and "read" for *Dack* sessions.
pOptArgs.usProtSessionID	This field is set to the Freeway session ID.
pOptArgs.usProtSequence	This field is set to 0 (zero).
pOptArgs.usProtXParms[0]	This field is set to 14 (FMP protocol value)
pOptArgs.usProtXParms[1]	This field is set to 0 (zero).

An FMP application can access links on an ICP through *Manager* (data) nodes, *Control* nodes, or *Dack* nodes. The link is determined by the node parameter in the FW_QIO(W) call (Section 1.3.2 on page 24) which is then mapped to a particular session defined in the DLI configuration file. Table 2–10 summarizes the session mapping provided by the protocol converter. *Manager* and *Dack* nodes are mapped to sessions by first determining the link associated with the FW_QIO(W) node parameter (see Table 2–10), and then finding the matching portNo DLI configuration parameter within the appropriate *Manager* or *Dack* session definitions (refer back to Figure 2–2 on page 38). *Control* nodes are mapped to sessions using the link field in the ICP header (Table 2–8) to find the matching portNo DLI configuration parameter within the appropriate *Control* session definitions (see Figure 2–2 on page 38).

Each FMP session must specify mode = "mgr", mode = "control", or mode = "read" in the DLI configuration file. The session name associated with each node/link in Figure 2–2 on page 38 is of the format B0L0…B0L(*n*–1) for *Manager* nodes, B0C0…B0C(*n*–1) for *Control* nodes, or B0D0…B0D(*n*–1) for *Dack* (read) nodes, where *n* = the maximum number of links.

Table 2–10: Session Mapping (FMP)

Link# Associated with Session Name[1]	Manager Node (mode="mgr")	Dack Node (mode="read")	Control Node[2] (mode="control")
0	1	21	41
1	2	22	41
⋮	⋮	⋮	⋮
n–1	*n*	*n*+20	41

[1] The session name associated with each node/link in Figure 2–2 on page 38 is of the format B0L0…B0L(*n*–1) for *Manager* nodes, B0C0…B0C(*n*–1) for *Control* nodes, and B0D0…B0D(*n*–1) for *Dack* nodes, where *n* = the maximum number of links.

[2] For writes, the application ICP header link field determines the session. For reads, all *Control* session reads complete to node 41.

The following example illustrates the SQIO session mapping:

An FMP SQIO configuration file, which could be named ZOA1, might contain the following string:

```
fmpconfig.bin, FMP, (B0L1, B0C1, B0D1)
```

Each of the session names (B0L1, B0C1, and B0D1) is defined in the fmpconfig DLI configuration file (similar to Figure 2–2 on page 38). To write to each of these sessions, your SQIO application would do the following (also see Table 2–10):

B0L1 Call FW_QIO(W) with the node parameter = 2. The protocol converter then finds the matching session in the fmpconfig file with mode = "mgr" and portNo = 1.

B0C1 Call FW_QIO(W) with the node parameter = 41 and the ICP header link field = 1. The protocol converter then finds the matching session in the fmpconfig file with mode = "control" and portNo = 1.

B0D1 Call FW_QIO(W) with the node parameter = 22. The protocol converter then finds the matching session in the fmpconfig file with mode = "read" and portNo = 1.

A simpler FMP SQIO configuration file to use link 0 and link 1 as *Manager* (data) nodes, allowing another application to use the *Control* node, would be:

```
fmpconfig.bin, FMP, (B0L0, B0L1)
```

Note

The session list in your SQIO configuration file should include only those sessions defined in the DLI configuration file that will actually be accessed by your application.

2.5 SWIFT Protocol Converter

The SWIFT protocol converter attempts to simulate, as much as possible, a "seamless" programming environment between the ICP version of the SWIFT protocol, and the Freeway version of the protocol. The SWIFT protocol converter name is SWIFT.

The SWIFT protocol converter discards the following responses to commands that exist in the Freeway version of this protocol, but not in the ICP version:

- Link Configuration Confirmation
- Set Buffer Size Confirmation

The client application supplies an ICP header as part of the data buffer for all SQIO read and write requests. The SWIFT protocol converter performs the mapping between the ICP header and the DLI optional arguments used in DLI *Raw* read and write operations, as shown in Table 2–11.

Table 2–11: ICP Header Mapping to DLI Optional Arguments (SWIFT)

ICP Header	DLI Optional Arguments
function[1]	pOptArgs.usProtCommand
link	pOptArgs.usProtLinkID
error_status	pOptArgs.usICPStatus
modifier	pOptArgs.iProtModifier
sequence	pOptArgs.usProtCircuitID

[1] The SWIFT protocol converter is responsible for converting the ICP function numbers into the Freeway command numbers, and *vice versa*.

Table 2–12 shows additional DLI optional arguments mapping.

Table 2–12: Additional Optional Arguments Mapping (SWIFT)

DLI Optional Arguments	Mapping
pOptArgs.iProtModifier	This field is set to the mode parameter in the DLI configuration file. Valid modes are "mgr" for *Manager* sessions, "control" for *Control* sessions, "read" for *Dack* sessions, and "trace" for *Trace* sessions.
pOptArgs.usProtSessionID	This field is set to the Freeway session ID.
pOptArgs.usProtSequence	This field is set to 0 (zero).
pOptArgs.usProtXParms[0]	This field is set to 12 (SWIFT protocol value)
pOptArgs.usProtXParms[1]	This field is set to 0 (zero).

A SWIFT application can access links on an ICP through *Manager* (data) nodes, *Control* nodes, *Dack* nodes, or *Trace* nodes. The link is determined by the node parameter in the FW_QIO(W) call (Section 1.3.2 on page 24) which is then mapped to a particular session defined in the DLI configuration file. Table 2–13 summarizes the session mapping provided by the protocol converter. *Manager* and *Dack* nodes are mapped to sessions by first determining the link associated with the FW_QIO(W) node parameter (see Table 2–13), and then finding the matching portNo DLI configuration parameter within the appropriate *Manager* or *Dack* session definitions (refer back to Figure 2–2 on page 38). *Control* and *Trace* nodes are mapped to sessions using the link field in the ICP header (Table 2–11) to find the matching portNo DLI configuration parameter within the appropriate *Control* or *Trace* session definitions (see Figure 2–2 on page 38).

Each SWIFT session must specify mode = "mgr", mode = "control", mode = "read", or mode = "trace" in the DLI configuration file. The session name associated with each node/link in Figure 2–2 on page 38 is of the format B0L0…B0L(*n*–1) for *Manager* nodes, B0C0…B0C(*n*–1) for *Control* nodes, B0D0…B0D(*n*–1) for *Dack* (read) nodes, and B0T0…B0T(*n*–1) for *Trace* nodes, where *n* = the maximum number of links.

Table 2–13: Session Mapping (SWIFT)

Link# Associated with Session Name[1]	Manager Node (mode="mgr")	Dack Node (mode="read")	Control Node[2] (mode="control")	Trace Node[b] (mode="trace")
0	1	21	41	42
1	2	22	41	42
⋮	⋮	⋮	⋮	⋮
n–1	*n*	*n*+20	41	42

[1] The session name associated with each node/link in Figure 2–2 on page 38 is of the format B0L0…B0L(*n*–1) for *Manager* nodes, B0C0…B0C(*n*–1) for *Control* nodes, B0D0…B0D(*n*–1) for *Dack* nodes, and B0T0…B0T(*n*–1) for *Trace* nodes, where *n* = the maximum number of links.

[2] For writes, the application ICP header link field determines the session. For reads, all *Control* or *Trace* session reads complete to node 41 or 42 respectively.

The following example illustrates the SQIO session mapping:

A SWIFT SQIO configuration file, which could be named ZOA1, might contain the following string:

swfconfig.bin, SWIFT, (B0L1, B0C1, B0D1, B0T1)

Each of the session names (B0L1, B0C1, B0D1, and B0T1) is defined in the swfconfig DLI configuration file (similar to Figure 2–2 on page 38). To write to each of these sessions, your SQIO application would do the following (also see Table 2–13):

B0L1 Call FW_QIO(W) with the node parameter = 2. The protocol converter then finds the matching session in the swfconfig file with mode = "mgr" and portNo = 1.

B0C1 Call FW_QIO(W) with the node parameter = 41 and the ICP header link field = 1. The protocol converter then finds the matching session in the swfconfig file with mode = "control" and portNo = 1.

B0D1 Call FW_QIO(W) with the node parameter = 22. The protocol converter then finds the matching session in the swfconfig file with mode = "read" and portNo = 1.

B0T1 Call FW_QIO(W) with the node parameter = 42 and the ICP header link field = 1. The protocol converter then finds the matching session in the swfconfig file with mode = "trace" and portNo = 1.

A simpler SWIFT SQIO configuration file to use link 0 and link 1 as *Manager* (data) nodes, allowing another application to use the *Control* node, would be:

```
swfconfig.bin, SWIFT, (B0L0, B0L1)
```

Note

The session list in your SQIO configuration file should include only those sessions defined in the DLI configuration file that will actually be accessed by your application.

2.6 CHIPS Protocol Converter

The CHIPS protocol converter attempts to simulate, as much as possible, a "seamless" programming environment between the ICP version of the CHIPS protocol, and the Freeway version of the protocol. The CHIPS protocol converter name is CHIPS.

The CHIPS protocol converter discards the following responses to commands that exist in the Freeway version of this protocol, but not in the ICP version:

- Link Configuration Confirmation
- Set Buffer Size Confirmation

The client application supplies an ICP header as part of the data buffer for all SQIO read and write requests. The CHIPS protocol converter performs the mapping between the ICP header and the DLI optional arguments used in DLI *Raw* read and write operations, as shown in Table 2–14.

Table 2–14: ICP Header Mapping to DLI Optional Arguments (CHIPS)

ICP Header	DLI Optional Arguments
function[1]	pOptArgs.usProtCommand
link	pOptArgs.usProtLinkID
error_status	pOptArgs.usICPStatus
modifier	pOptArgs.iProtModifier
sequence	pOptArgs.usProtCircuitID

[1] The CHIPS protocol converter is responsible for converting the ICP function numbers into the Freeway command numbers, and *vice versa*.

Table 2–15 shows additional DLI optional arguments mapping.

Table 2–15: Additional Optional Arguments Mapping (CHIPS)

DLI Optional Arguments	Mapping
pOptArgs.iProtModifier	This field is set to the mode parameter in the DLI configuration file. Valid modes are "mgr" for *Manager* sessions, "control" for *Control* sessions, "read" for *Dack* sessions, and "trace" for *Trace* sessions.
pOptArgs.usProtSessionID	This field is set to the Freeway session ID.
pOptArgs.usProtSequence	This field is set to 0 (zero).
pOptArgs.usProtXParms[0]	This field is set to 13 (CHIPS protocol value)
pOptArgs.usProtXParms[1]	This field is set to 0 (zero).

A CHIPS application can access links on an ICP through *Manager* (data) nodes, *Control* nodes, *Dack* nodes, or *Trace* nodes. The link is determined by the node parameter in the FW_QIO(W) call (Section 1.3.2 on page 24) which is then mapped to a particular session defined in the DLI configuration file. Table 2–16 summarizes the session mapping provided by the protocol converter. *Manager* and *Dack* nodes are mapped to sessions by first determining the link associated with the FW_QIO(W) node parameter (see Table 2–16), and then finding the matching portNo DLI configuration parameter within the appropriate *Manager* or *Dack* session definitions (refer back to Figure 2–2 on page 38). *Control* and *Trace* nodes are mapped to sessions using the link field in the ICP header (Table 2–14) to find the matching portNo DLI configuration parameter within the appropriate *Control* or *Trace* session definitions (see Figure 2–2 on page 38).

Each CHIPS session must specify mode = "mgr", mode = "control", mode = "read", or mode = "trace" in the DLI configuration file. The session name associated with each node/link in Figure 2–2 on page 38 is of the format B0L0…B0L(*n*–1) for *Manager* nodes, B0C0…B0C(*n*–1) for *Control* nodes, B0D0…B0D(*n*–1) for *Dack* (read) nodes, and B0T0…B0T(*n*–1) for *Trace* nodes, where *n* = the maximum number of links.

Table 2–16: Session Mapping (CHIPS)

Link# Associated with Session Name[1]	Manager Node (mode="mgr")	Dack Node (mode="read")	Control Node[2] (mode="control")	Trace Node[b] (mode="trace")
0	1	21	41	42
1	2	22	41	42
⋮	⋮	⋮	⋮	⋮
n–1	*n*	*n*+20	41	42

[1] The session name associated with each node/link in Figure 2–2 on page 38 is of the format B0L0…B0L(*n*–1) for *Manager* nodes, B0C0…B0C(*n*–1) for *Control* nodes, B0D0…B0D(*n*–1) for *Dack* nodes, and B0T0…B0T(*n*–1) for *Trace* nodes, where *n* = the maximum number of links.

[2] For writes, the application ICP header link field determines the session. For reads, all *Control* or *Trace* session reads complete to node 41 or 42 respectively.

The following example illustrates the SQIO session mapping:

A CHIPS SQIO configuration file, which could be named ZOA1, might contain the following string:

chpconfig.bin, CHIPS, (B0L1, B0C1, B0D1, B0T1)

Each of the session names (B0L1, B0C1, B0D1, and B0T1) is defined in the chpconfig DLI configuration file (similar to Figure 2–2 on page 38). To write to each of these sessions, your SQIO application would do the following (also see Table 2–16):

B0L1 Call FW_QIO(W) with the node parameter = 2. The protocol converter then finds the matching session in the chpconfig file with mode = "mgr" and portNo = 1.

B0C1 Call FW_QIO(W) with the node parameter = 41 and the ICP header link field = 1. The protocol converter then finds the matching session in the chpconfig file with mode = "control" and portNo = 1.

B0D1 Call FW_QIO(W) with the node parameter = 22. The protocol converter then finds the matching session in the chpconfig file with mode = "read" and portNo = 1.

B0T1 Call FW_QIO(W) with the node parameter = 42 and the ICP header link field = 1. The protocol converter then finds the matching session in the chpconfig file with mode = "trace" and portNo = 1.

A simpler CHIPS SQIO configuration file to use link 0 and link 1 as *Manager* (data) nodes, allowing another application to use the *Control* node, would be:

```
chpconfig.bin, CHIPS, (B0L0, B0L1)
```

Note

The session list in your SQIO configuration file should include only those sessions defined in the DLI configuration file that will actually be accessed by your application.

2.7 X.25 and HDLC Protocol Converters

The X.25 and HDLC Protocol Converters attempt to simulate, as much as possible, a "seamless" programming environment between the ICP32*xx*/ICP6000/ICP9000 (subsequently denoted as ICP3222) version of the X.25 protocol and the Freeway version of the X.25 protocol. The X.25 protocol converter name is X25. The protocol converter name to use the HDLC LAPB mode in X.25 is HDLC.

The client application supplies an ICP header as part of the data buffer for all SQIO read and write requests. The X.25 protocol converter performs the mapping between the ICP header and the DLI optional arguments used in DLI *Raw* write and read operations, as shown in Table 2–17 and Table 2–18, respectively.

Table 2–17: Client Write Header Mapping to DLI Optional Arguments (X.25)

Application Supplied ICP Header	DLI Optional Arguments
command	pOptArgs.usProtCommand
arg (if command supplies a link, or the link associated with the station of a user[1] command, otherwise 0)	pOptArgs.usProtLinkID
arg (if HROTATE, otherwise 0)	pOptArgs.iProtModifier
station (if user[a] commands only, otherwise 0)	pOptArgs.usProtCircuitID

[1] The *X.25 Low-Level Interface* manual describes user and manager commands.

For a detailed understanding of the Freeway version of X.25 see the *X.25 Low-Level Interface* manual. In particular, the table entitled "*Host Packet PROTHDR Fields*" is very instructive. SQIO currently supports only Service Access Points (SAPs) for X.25 and Diagnostics (DLI_X25_SAP_X25 and DLI_X25_SAP_DIAG).

The SQIO configuration file for SAP X.25 must include at most, one *Manager* session (opened as DLI_X25_MGR_API); and possibly multiple *User* sessions (opened as DLI_X25_USER_API). Each *User* session defines a single unique link. The normal

Table 2–18: ICP Read Header Mapping from DLI Optional Arguments (X.25)

SQIO Supplied ICP Header	DLI Optional Arguments
command	pOptArgs.usProtCommand
station	pOptArgs.usProtCircuitID
size	Returned DLI read size
arg	pOptArgs.usProtLinkID (unless command type indicates that iProtModifier supplies value or D-bit value is supplied in Quality of Service data)

assumption is that an SQIO configuration file defines a single Freeway ICP board. However, multiple SQIO configuration files can be assigned (using FW_ASSIGN, Section 1.2.2 on page 20), including via multiple VMS processes, as long as each specified session has a unique session type (*Manager* or *User*), ICP board number, and link number combination (for example, a *Manager* session on link 5 of board 0 cannot be defined in two different SQIO configuration files).

The Freeway implementation of X.25 allows many Switched Virtual Circuits (SVCs, or stations) per link. Since there can be only one DLI *User* session per SQIO link, it follows that there can be multiple stations existing for a single DLI *User* session. SQIO maintains the association of stations to DLI sessions based on how the stations are configured on the Link/Station configuration command. A station cannot be configured on multiple links (or DLI sessions). All SQIO reads and writes are done through the single DLI *Manager* session and/or the DLI *User* sessions.

An X.25 SQIO application can access the Freeway version of the X.25 protocol with either manager commands or user commands and their associated sessions. *Manager* commands are always mapped to the single *Manager* session defined within the SQIO configuration file, and user commands are always mapped to the appropriate *User* session defined within the SQIO configuration file. The SQIO protocol converter for X.25 automatically does this mapping for the application. See Table 2–19 for a definition of commands to their type (manager or user).

Table 2–19: Mapping of ICP32*xx* X.25 Commands to Freeway X.25 Commands

X.25 Command Number	ICP32*xx* X.25 Host Command	Freeway X.25 Client Command	Command Type[1] Manager (M) or User (U)
1	HCALL	DLI_X25_HOST_CALL_REQ	U
3	HCONNECT	DLI_X25_HOST_CALL_ACCEPTED	U
5	HHANGUP	DLI_X25_HOST_CLR_REQ	U
7	HTONE	DLI_X25_HOST_CLR_CONFIRMED	U
9	HRSET	DLI_X25_HOST_RESET_REQ	U
11	HRSETC	DLI_X25_HOST_RESET_CONFIRMED	U
13	HINIT_SLP	DLI_X25_HOST_INIT_SLP	U
	HINIT_MLP	DLI_X25_HOST_INIT_MLP	U
15	reserved	reserved	
17	HDATA	DLI_X25_HOST_DATA	U
19	HINT	DLI_X25_HOST_INTERRUPT	U
21	HINTC	DLI_X25_HOST_INT_CONFIRMED	U
23	HENABLE	DLI_X25_HOST_ENABLE_LINK	M
25	HDISABLE	DLI_X25_HOST_DISBALE_LINK	M
27	HCONFIG	DLI_X25_HOST_CFG_LINK	M
29	HREG_ICF	DLI_X25_HOST_ADD_INCALL_FILTER	U
31	HDEL_ICF	DLI_X25_HOST_DEL_INCALL_FILTER	U
33	HSTATS	DLI_X25_HOST_GET_STATISTICS	M
35	HREJECT	DLI_X25_HOST_CFG_IREJECT_FORMAT	M
37	HBUFI	DLI_X25_HOST_CFG_BUF	M
39	reserved	reserved	
41	HABORT	DLI_X25_HOST_ABORT	U
43	HCSCON	DLI_X25_HOST_CFG_CALL_SERVICE	M
45	HCONMLP	DLI_X25_HOST_CFG_MLP	M
47	HREDIRECT	DLI_X25_HOST_REDIRECT	U

Table 2–19: Mapping of ICP32*xx* X.25 Commands to Freeway X.25 Commands *(Cont'd)*

X.25 Command Number	ICP32*xx* X.25 Host Command	Freeway X.25 Client Command	Command Type[1] Manager (M) or User (U)
49,51	reserved	reserved	
53	HMONITOR[2]	DLI_X25_HOST_MONITOR_REQ	M
55	HVERSION	DLI_X25_HOST_GET_VERSION	M
57	HSTATS_CLEAR	DLI_X25_HOST_CLR_STATISTICS	M
59	HSTATS_SAMPLE	DLI_X25_HOST_SAMPLE_STATISTICS	M
61	HADJUST_FLOW	DLI_X25_HOST_ADJUST_FLOW_CTRL	U
63	HOPEN_PVC	DLI_X25_HOST_OPEN_PVC	U
65	HCLOSE_PVC	DLI_X25_HOST_CLOSE_PVC	U
67	reserved	reserved	
69	HCLSTATE	DLI_X25_HOST_CTL_LINE_STATE_REQ	M
71	HREGRQ	DLI_X25_HOST_REGISTER	M
73,75,77	reserved	reserved	
79	HROTATE	DLI_X25_HOST_ROTATE	M
81	HBUFCLEAR	DLI_X25_HOST_BUF_CLR	M
99	HOPEN_SESSION	DLI_X25_HOST_OPEN_SESSION_NEG	M or U

[1] The *X.25 Low-Level Interface* manual describes user and manager commands.

[2] Only used for a diagnostic session with DLI mode = "read"

When a station is configured with the HCONFIG command, a link and station (or circuit) are specified in the application header. The X.25 protocol converter uses this link/station association to determine which *User* session is to be used on subsequent writes of user commands. Obviously, this implies that all user commands must include a station (or circuit) in the station field of the application header.

Each X.25 session defined in the SQIO configuration file must be defined in the DLI configuration file. Each session must specify mode = "mgr" for the *Manager* session or mode = "user" (or mode = "read") for the *User* sessions in the DLI configuration file. Figure 2–3 shows an example X.25 DLI configuration file specifying links 0 and 1.

From this example, all manager commands would be directed to the DLI session named "B0M0". All user commands that specify a station that was HCONFIG'd with link 0 would be directed to the DLI session named "B0U0", and user commands that specify a station that was HCONFIG'd with link 1 would be directed to the DLI session named "B0U1". Table 2–19 shows the mapping of ICP32*xx* X.25 Commands to Freeway X.25 Commands. Note that, while not necessary, the link number specified with the *Manager* session should coincide with a link number specified with one of the *User* sessions.

SQIO supports monitoring of one or more data links. A DLI monitor (or diagnostic) session is defined by specifying mode = "read" for a specified link. Monitoring is enabled and disabled using the HMONITOR command.

There are differences in the implementation between the ICP3222 (1988 CCITT/ISO) X.25 and the Freeway version of X.25. SQIO eliminates most of these differences with the X.25 protocol converter. The following discussion highlights these differences.

2.7.1 Node/Session Numbers

2.7.1.1 ICP3222

Normal Host commands and ICP responses are sent on Node 1 (P4 parameter of a QIO call). Special Host commands can be sent on an expedited Node number 2 to bypass a possible blockage on Node 1.

2.7.1.2 SQIO

Commands sent on node 1 are processed with a normal DLI write by SQIO. Commands sent on node 2 are processed with an expedited DLI write by SQIO.

```
//-------------------------------------------------------------------//
//                                                                   //
// "main" section.                                                   //
//                                                                   //
//-------------------------------------------------------------------//
main
{
   AsyncIO = "Yes";              // Asynchronous mode          //
   TSICfgName = "x25qaltcfg.bin";   // TSI configuration file name //
}

//-------------------------------------------------------------------//
// Define a section for the Manager session. Freeway server name    //
// is define in the tsi configuration file used.                     //
//-------------------------------------------------------------------//
B0M0
{
   Protocol = "RAW";             // RAW session type           //
   Transport = "Conn0";          // Transport connection name   //
                                 // defined in TSICfgName file  //
   BoardNo = 0;                  // ICP board number -- based 0 //
   PortNo = 0;                   // Port (link) number.        //
   Mode = "mgr";
   AlwaysQIO = "Yes";
   AsyncIO = "Yes";
}
//-------------------------------------------------------------------//
// Define a User section for 2 ports.                                //
//-------------------------------------------------------------------//
B0U0
{
   Protocol = "RAW";             // RAW session type           //
   Transport = "Conn0";          // Transport connection name   //
                                 // defined in TSICfgName file  //
   BoardNo = 0;                  // ICP board number -- based 0 //
   PortNo = 0;                   // Port (link) number.        //
   AlwaysQIO = "Yes";
   AsyncIO = "Yes";
   mode = "user";                // User Mode                  //
}
B0U1
{
   Protocol = "RAW";             // RAW session type           //
   Transport = "Conn0";          // Transport connection name   //
                                 // defined in TSICfgName file  //
   BoardNo = 0;                  // ICP board number -- based 0 //
   PortNo = 1;                   // Port (link) number.        //
   AlwaysQIO = "Yes";
   AsyncIO = "Yes";
   mode = "user";                // User Mode                  //
}
```

Figure 2–3: DLI Configuration File Example for X.25 SQIO

2.7.2 ICP3222 Host Commands Missing from Freeway X.25

2.7.2.1 Host Clear Confirmation (7)

The ICP3222 default is for the ICP to control clear confirmation for call service. (The SQIO version for Freeway allows this command at user discretion.)

2.7.2.2 Host Configure ICP Command Reject Format (35)

Freeway always returns a "verbose" formatted reject command packet (unchanged from the verbose form on the ICP3222). (The SQIO version for Freeway allows this command at user discretion.)

2.7.2.3 Host Safe Store Acknowledgment (79)

The SQIO version for Freeway allows this command at user discretion.

2.7.3 ICP3222 ICP Responses Missing from Freeway X.25

2.7.3.1 ICP No Station Assigned (60)

This packet is returned to the Host if an attempt is made to enable a link where no stations have been assigned to the link.

2.7.4 Differences Between ICP3222 and Freeway Commands

2.7.4.1 Call Service Configuration

X.25 provides a set of data elements that define quality of service (QOS). Each data element of QOS is defined by 1 to 3 subfields. The first subfield defines the type of element (identification code). The second subfield contains either a single data element, or if there is a variable length third subfield, then the second subfield contains the length of the third subfield. Most of the data elements that comprise QOS are identical between the ICP3222 versions of X.25 and the Freeway version of X.25. Exceptions of QOS ele-

ments that are documented in the ICP3222 X.25 manual but are not documented in the Freeway manual are:

- Configure Clear Confirmation Control — HF_CLR (20)
- Local DTE Address Length — HF_ADDR_LEN (45)

Note

The SQIO version for Freeway allows these QOS elements at user discretion.

2.7.4.2 Host Call Request (1) / Host Call Accepted (3)

D-bit support is specified with a QOS element instead of using the application header arg field:

Byte 0 = 46 (HF_D_BIT_SUPPORT)
Byte 1 = (0 for no, 1 for yes)

2.7.4.3 Host Configure Communication Link (27)

Link Configuration

Word 0 / bit 5 Not supported for Freeway. The ICP3222 can specify DCE startup as: 0 = DM, 1 = SABM.

Word 0 / bit 7 The ICP3222 can specify high speed as: 0 = no, 1 = yes. Freeway can specify high speed either with a custom data rate, or identical to the ICP3222 version of X.25.

Station Configuration

The X.25 Freeway documentation does not support the following ICP3222 options:

- ICP rotate window function (2)

- Host flow control function (3)
- Host safe store function (4)

Note

The SQIO version for Freeway allows these station configuration options at user discretion.

2.7.4.4 Quality of Service (QOS)

Freeway has additional quality of service (QOS) elements that do not exist for the ICP3222. See the table entitled "*qos Support in Packets*" in the *X.25 Low-Level Interface* manual for a QOS matrix.

2.7.4.5 Allow Stations to Receive a Call Request on a Link

- A Host Add Incall Filter command must be sent to the link for each station on the link that is allowed to receive a connection request.
- On receipt of an ICP Auto Connect, send a Host Delete Incall Filter to allow other stations on the same link to also receive connection requests.

2.7.4.6 Host Adjust Flow Control

This command must be used for the Freeway version of X.25 for virtual circuits (stations). For a detailed description, see the *X.25 Low-Level Interface* manual.

2.7.4.7 Host Enable Link (23) / Host Disable Link (25)

The Freeway version of X.25 does not allow the application link parameter to be –1, which was used by the ICP3222 version of X.25 to imply all links. For SQIO, each link must be separately enabled and disabled.

2.7.5 ICP Response Differences

2.7.5.1 The ICP Rotate Xmit Window (28)

For Freeway X.25 transmission acknowledgment, this response cannot be suppressed.

2.7.5.2 ICP Command Reject (36)

Freeway always returns a "verbose" formatted reject command packet (unchanged from the verbose form on the ICP3222).

2.7.5.3 D-bit Support

ICP Incoming Call (2), ICP Call Accepted (4), and ICP Auto Connect (46) can have the QOS D-bit support field returned in addition to the application header arg field.

2.7.6 Miscellaneous

SQIO automatically opens DLI *Manager* and *User* sessions on the first write or read that contains a manager or user command. User commands also cause SQIO to automatically open *Station* (circuit) sessions. For switched virtual circuits (SVCs), there can be multiple *Station* sessions per single *User* session. However, receipt of data from a station (circuit) cannot be properly handled by SQIO until the *Station* session is opened. Therefore, it is recommended that a new session-oriented command, HOPEN_SESSION (99), be sent to force an open for the *Station* session before an incoming packet, such as a Call Request packet, is received on an incoming line. This new command should be sent for each configured station (circuit). The ICP header should have the command field set to HOPEN_SESSION (99), the station number for the "station" session, with the argument field set to the link number for this station.

Index

Customer Report Form

We are constantly improving our products. If you have suggestions or problems you would like to report regarding the hardware, software or documentation, please complete this form and mail it to Protogate at 12225 World Trade Drive, Suite R, San Diego, CA 92128, or fax it to (877) 473-0190.

If you are reporting errors in the documentation, please enter the section and page number.

Your Name: _____

Company: _____

Address: _____

_____

_____

Phone Number: _____

Product: _____

Problem or
Suggestion: _____

_____

_____

_____

_____

_____

Protogate, Inc.
Customer Service
12225 World Trade Drive, Suite R
San Diego, CA 92128