

SPECTRA User Manual

Data Server Node

Reference: SPCTRA_2.5/USER/DSN/2.5

Issue Date: 13th February 1998

Author: Dave Skett

Approved by:

Project Manager:

Distribution:

Internal
Customers
Prospects

This document is the property of **Concept Systems Limited**. It must not be copied, in whole or in part, or otherwise disclosed, without prior written consent. Any copies of this document, or part thereof, must also include a copy of this legend. This document is supplied without liability for errors or omissions.

© Copyright Concept Systems Limited 1998

1 Logie Mill, Beaverbank Business Park, Logie Green Road, Edinburgh EH7 4HG, Scotland
Tel: (+44)131 557 5595 Fax: (+44)131 557 2367 Telex: 727673 CONCEP G

Table of contents

Chapter 0 - About this document	0-1
Revision history	0-1
Document cross-reference	0-1
Conventions used in this document	0-1
Chapter 1 - Introduction	1-1
What is the Data Server?.....	1-1
Getting started.....	1-1
Prerequisites to running the Data Server	1-1
Starting the Node	1-1
Normal operation	1-1
SPECTRA needs this Node!	1-1
Chapter 2 - Data Server items	2-1
Data items	2-1
Data Item Interdependencies	2-1
Data Item Instances	2-3
Obtaining information about data items	2-5
Mode 1	2-5
Mode 2	2-6
Chapter 3 - Theory	3-1
SPECTRA's client/server architecture	3-1
Theory of the producer/consumer model	3-2
Chapter 4 - The dsn_config.dat file.....	4-1
Chapter 5 - Controlling Data Server Debugging (Advanced)	5-1

Chapter 0 - About this document

This document is the user manual for the Data Server Node of Concept Systems Limited's integrated navigation system designed for marine geophysical survey usage (SPECTRA). SPECTRA is a modular system, comprising various Nodes.

This manual describes how to use the Data Server. [Chapter 1](#) introduces the Node and its use. [Chapter 2](#) explains data items. Further chapters give the theoretical background to the way the Data Server works, and describe the `dsn_config.dat` file.

For an overview of how SPECTRA works, see the *SPECTRA User Manual - Beginner's Guide*. For a glossary of terms, see the *SPECTRA User Manual - Glossary*. For brief descriptions of each Node, detailed descriptions of the options provided by other SPECTRA Nodes, and the commands to start Nodes, see the *Spectra Software Installation and Set Up manual*.

Revision history

Document Reference	Date	Notes
SPCTRA_2.0/USER/DSN/1.0	1 st November 1996	First issue
SPCTRA_2.0/USER/DSN/2.0	27 th May 1997	Default configuration values corrected.
SPCTRA_2.5/USER/DSN/2.5	13 th February 1998	Tables of data server items and instances, and advanced section on <code>admin_dsn</code> utility added.

Document cross-reference

Document Reference	Title
SPECTRA/USER/BEGIN	<i>SPECTRA User Manual - Beginner's Guide</i>
SPECTRA/USER/GLOSS	<i>SPECTRA User Manual - Glossary</i>
SPECTRA/USER/ALGO	<i>SPECTRA User Manual - Algorithms</i>
SPECTRA/USER/NCN	<i>SPECTRA User Manual - Network Calculation Node</i>
SPECTRA/USER/RTCN	<i>SPECTRA User Manual - Real Time Configuration Node</i>
SPECTRA/USER/LMN	<i>SPECTRA User Manual - Line Management Node</i>
SHARED/RTNU_OP_2	<i>RTNU & Closure Unit Hardware Installation and Operation</i>
SHARED/MICRO_4	<i>RTNμ Hardware Installation and Operation</i>

Conventions used in this document

The conventions used in this, and all other SPECTRA user manuals, are those defined in the *SPECTRA User Manual - Beginner's Guide*.

Chapter 1 - Introduction

This chapter introduces you to the Data Server and how to get started with it.

What is the Data Server?

The Data Server Node acts as a central data collection and supply system. Data of any type (raw data, processed data, or control information) may be produced by any Node on the network, and forwarded to the Data Server for storage. Other Nodes may then ask the Data Server to supply certain items of data to them.

A system of registering interests in specified items of data allows the Data Server to optimise its operation – by knowing which data items are required by some other Node(s), the Data Server can avoid wasting valuable time and storage space on unwanted items of data.

All Nodes (including the RTNU) connect to the Data Server and transmit or receive data via Remote Procedure Calls (RPCs).

Getting started

Prerequisites to running the Data Server

The Data Server requires the `dsn_config.dat` file, see [Chapter 4 “The `dsn_config.dat` file”](#) for details.

Starting the Node

This Node would normally be started via the Monitor Node. Details of the commands required to start this and other SPECTRA Nodes can be found in the *Spectra Software Installation and Set Up manual*.

Normal operation

The Data Server requires no direct user intervention – it runs in the background and has no graphical user interface. The data items stored by the Data Server are produced and consumed by other SPECTRA Nodes (see [Chapter 2 “Data Server items”](#)).

SPECTRA needs this Node!

The Data Server *must* be running before any other SPECTRA Nodes can communicate and exchange data.

Chapter 2 - Data Server items

This chapter gives the data items produced and consumed by SPECTRA Nodes and stored by the Data Server. It also explains how to find out information about data items.

Data items

The Data Server is simply the repository for many unique items of data. The current mode of SPECTRA operation ensures that at any one time there is only a single instance of each data item held in the Data Server. Each data item is assigned a unique identifying name, consisting of two parts:

group An identifier always started and terminated with the @ character, which identifies the data group type.

id An identifier which uniquely identifies the item within its group.

For example, an obstacle identified by “rig” would be held in the Data Server as:

@OBSTACLE@rig

The underlying operation of SPECTRA can be outlined by describing the key items that are produced and consumed by the following Nodes. The items listed below are the basic subset of items required to operate SPECTRA.

Note: For information on using these Nodes, see the *Spectra Software Installation and Set Up manual* and each individual Node’s user manual and/or online help.

Data Item Interdependencies

The following table illustrates the interdependencies of the various data items between nodes. For example, the second row of the table shows that the SHOTNOW item is produced by the ILC (in the RTNU) and consumed by both the NCN and DLN.

ITEM	PRODUCER														
		ILC	RTCN	LMN	SNN	NCN	DN	DLN	MN	QCLN	BCN	SRI	VIP	SVD	QCHDR
SHOT	ILC														
SHOTNOW	ILC					✓		✓							
GPS	ILC					✓	✓								
RAW	ILC					✓		✓	✓	✓					
ILCCONF	ILC		✓					✓							
ILCSTAT	ILC														
TRIGGERED	ILC									✓					

ITEM	PRODUCER															
		ILC	RTCN	LMN	SNN	NCN	DN	DLN	MN	QCLN	BCN	SRI	VIP	SVD	QCHDR	
INTERFACE	RTCN								✓	✓						
TRIGGERS	RTCN								✓	✓						
CODE	RTCN															
RTCNCNF	RTCN	✓			✓				✓			✓				
CURRENT	LMN					✓	✓	✓		✓		✓		✓		
PROJECT	LMN	✓	✓					✓		✓						
STD_PSD_NAME	LMN					✓	✓	✓		✓	✓	✓		✓		
PSD@STANDARD	LMN					✓	✓	✓								
PSD@ALTERNATE	LMN					✓	✓	✓								
OBSTACLE	LMN						✓									
SURVEY	LMN						✓									
STATION	LMN						✓									
LMNCONF	LMN				✓	✓	✓	✓	✓		✓					
VESSEL	SNN							✓		✓						
ECHOSOUNDER	SNN					✓		✓		✓						
GYRO	SNN					✓		✓		✓						
STREAMER	SNN					✓	✓	✓		✓						
GUNARRAY	SNN					✓	✓	✓								
GUNSEQUENCE	SNN	✓				✓										
NETWORK	SNN					✓	✓	✓		✓		✓		✓		
FILTER	SNN					✓										
OBSKO	SNN					✓			✓							
PRIMARY	SNN					✓	✓	✓		✓					✓	
NCC	SNN			✓		✓								✓		
SATELLITE	SNN					✓	✓	✓								
RAWGPS	SNN							✓								
SOUND	SNN							✓								
PRH	SNN							✓								
SNNCONF	SNN		✓			✓	✓	✓	✓							
TMPNCC	NCNncn	✓														
SHOTPREDICT	NCNncn	✓	✓													
TURNSTATUS	NCNncn						✓				✓					
LINESTATUS	NCNncn			✓		✓	✓	✓	✓	✓		✓		✓	✓	
TMPMAIN	NCNmain	✓														
SHOTPOSITION	NANmain						✓	✓	✓	✓		✓		✓	✓	
NCNCONF	NANmain							✓								
NETSOLUTION	NCN						✓	✓	✓	✓		✓		✓	✓	
NETQUALITY	NCN									✓						
RTBCNF	DN								✓							

ITEM	PRODUCER	ILC	RTCN	LMN	SNN	NCN	DN	DLN	MN	QCLN	BCN	SRI	VIP	SVD	QCHDR
EMERGENCY	DN						✓		✓						
ESTIMATE	DN					✓									
DLNCONF	DLN								✓						
GUNDATA	DLN						✓		✓						
TLNSTAT@P1	DLN														
TLNSTAT@P2	DLN														
BINNER	BCN						✓								
BNCONF	BN						✓								
MNCONF	MN	✓													
SRICONF	SRI						✓		✓						
CURRENT_TURN	TURN					✓	✓					✓			

Data Item Instances

The following table lists every item, shows whether there is only a single instance of the data item or whether there can be multiple instances, gives an example of the data item and a brief description of its content.

ITEM	SINGLE OR MULTIPLE	EXAMPLE	DESCRIPTION
SHOT	SINGLE	@SHOT@	Shot indicator item
SHOTNOW	SINGLE	@SHOTNOW@	Shot indicator item
GPS	SINGLE	@GPS@	GPS time item
RAW	MULTIPLE	@RAW@EA5000	Raw Interface data items
ILCCONF	MULTIPLE	@ILCONF@morar	Configuration items from RTNU or RTN μ
ILCSTAT	MULTIPLE	@ILCSTAT@morar	RTNU or RTN μ status information
TRIGGERED	MULTIPLE	@TRIGGERED@shot	Produced when corresponding trigger fires
INTERFACE	MULTIPLE	@INTERFACE@SIPS	Interface definition items
TRIGGERS	SINGLE	@TRIGGERS@	Triggers configuration item
CODE	MULTIPLE	@CODE@GeneralØØ	Interface application code items
RTCNCONF	SINGLE	@RTCNCONF@	RTCN configuration item
CURRENT	SINGLE	@CURRENT@	Current line item
PROJECT	SINGLE	@PROJECT@	Project information
STD_PSD_NAME	SINGLE	@STD_PSD_NAME@	Name of standard PSD
PSD@STANDARD	MULTIPLE		Standard Projection/Spheriod/Datum data items
PSD@ALTERNATE	MULTIPLE		Alternate Projection/Spheriod/Datum data items
OBSTACLE	MULTIPLE	@OBSTACLE@rig	Obstacle feature data items
SURVEY	MULTIPLE	@SURVEY@lineA	Survey line feature data items
STATION	MULTIPLE	@STATION@Aberdeen	Station feature data items

ITEM	SINGLE OR MULTIPLE	EXAMPLE	DESCRIPTION
LMNCONF	SINGLE	@LMNCONF@	LMN Configuration item
VESSEL	MULTIPLE	@VESSEL@OilSearcher	Vessel item
ECHOSOUNDER	MULTIPLE	@ECHOSOUNDER@echo1	Echosounder item
GYRO	MULTIPLE	@GYRO@OilGyro	Gyro item
STREAMER	MULTIPLE	@STREAMER@s1	Streamer item
GUNARRAY	MULTIPLE	@GUNARRAY@G1	Gunarray item
GUNSEQUENCE	SINGLE	@GUNSEQUENCE@	Controls gun firing sequence
NETWORK	MULTIPLE	@NETWORK@main	Network data definition items
FILTER	MULTIPLE	@FILTER@main	Filter data for an item
OBSKO	SINGLE	@OBSKO@	Observation and compass knockout
PRIMARY	SINGLE	@PRIMARY@	Identifies primary network
NCC	SINGLE	@NCC@	Identifies NCC Network
SATELLITE	MULTIPLE	@SATELLITE@sat1	Satellite item
RAWGPS	SINGLE	@RAWGPS@	Raw GPS configuration
SOUND	SINGLE	@SOUND@	Contains multiple velocity of sound profiles
PRH	MULTIPLE	@PRH@pr1	Pitch roll heave configuration
SNNCONF	SINGLE	@SNNCONF@	SNN configuration item
TMPNCC	SINGLE	@TMPNCC@	Temporary NCC data
SHOTPREDICT	SINGLE	@SHOTPREDICT@	Shot prediction item
TURNSTATUS	SINGLE	@TURNSTATUS@	Current turn status
LINESTATUS	SINGLE	@LINESTATUS@	Current line status
TMPMAIN	SINGLE	@TMPMAIN@	Temporary primary network data
SHOTPOSITION	SINGLE	@SHOTPOSITION@	Shot position item
NCNCONF	SINGLE	@NCNCONF@	Primary network configuration information
NETSOLUTION	MULTIPLE	@NETSOLUTION@main	Network solution data items
NETQUALITY	MULTIPLE	@NETQUALITY@main	Network quality data items
RTBCONF	SINGLE	@RTBCONF@	Binning status item
EMERGENCY	SINGLE	@EMERGENCY@	Position marked during an emergency situation
ESTIMATE	SINGLE	@ESTIMATE@	Position estimate item
DLNCONF	SINGLE	@DLNCONF@	DLN status item
GUNDATA	SINGLE	@GUNDATA@	Details of gun fired
TLNSTAT@P1	SINGLE	@TLNSTAT@P1	Status of P1 tape logging
TLNSTAT@P2	SINGLE	@TLNSTAT@P2	Status of P2 tape logging
BINNER	SINGLE	@BINNER@	Binning configuration item
BNCONF	SINGLE	@BNCONF@	Bullseye node configuration information
MNCONF	SINGLE	@MNCONF@	Monitor node configuration information
SRICONF	MULTIPLE	@SRICONF@vesselA	SPECTRA Robtrack Interface configuration information
CURRENT_TURN	SINGLE	@CURRENT_TURN@	Current turn configuration item
TURN	MULTIPLE	@TURN@current	Turn feature data item

Obtaining information about data items

A utility process (`monitor`) is available, providing useful low level diagnosis of common fault conditions. The `monitor` utility process can be run from any X-window and has two common modes of operation, described below. Further options can be previewed using the `-h` command.

Mode 1

The `monitor` utility can be used to list data items, using the command:
`monitor -l match_string`

This provides a list to the screen of the items in the Data Server corresponding to the string defined by `match_string`. For example, to view the `@RAW@` data items in the Data Server enter the following command:

```
monitor -l @RAW
```

which will produce a listing to the screen similar to the following:

ITEM	INSTANCE	SIZE	IN PROD	PRODUCER	INTERESTS
@RAW@GYRO	1000	48	Yes	ILC	NCN ncc NCN main
@RAW@SB5	2356	560	Yes	ILC	NCN ncc NCN main 2354 2

where:

ITEM is the unique name of the item in the Data Server.

INSTANCE is the number of items that have been placed into the Data Server. If this value is -1 then the item has never been produced.

SIZE is the size of the data in bytes.

IN PROD is "Yes" if the item is in production.

PRODUCER is the mnemonic of the process which has produced the item, e.g. ILC for the Interface Lan Controller (RTNU).

INTERESTS is the mnemonic of the processes wishing to consume the item.

The four-digit number to the right of this entry is the instance number of the last item consumed by this node. In the case illustrated, instance 2356 has been produced by the ILC but the NCNmain has only consumed up to 2354. Comparison of these figures is a quick way to check that a node is consuming properly. The one-digit number on the far right indicates the type of consumer interest and is for internal use only.

If this test is repeated within a few seconds, the frequency of data production can be also be checked.

Chapter 3 - Theory

This chapter outlines the general theory of a producer/consumer model, and relates this to the client/server architecture used by SPECTRA, in which the Data Server is the server and other SPECTRA Nodes are the clients.

SPECTRA's client/server architecture

The SPECTRA Integrated Navigation System is based on a Client/Server architecture. In this case the client is a process (Node) and the server is the Data Server process (Node).

The Data Server acts in a passive role - that is, it does not initiate any actions of its own volition. With the exception of some internal housekeeping tasks, any actions the server does take will be as a direct result of a client requesting a particular service from it. Any time that is not spent doing housekeeping, or servicing a client's request, is spent simply waiting for a new request. The Data Server never initiates any action on any of its clients.

Clients can initiate some action on the Data Server by making a remote procedure call (RPC) on the required Node. Note that although these calls are called "remote", this does not mean that they can only be made between Nodes which are on separate hardware platforms - any number of Nodes on the same platform may still exchange data via RPCs.

Before a client can make an RPC on any other Node, it first has to establish a communications link (or an "RPC link") with that Node. A single RPC link will allow any number of RPCs to be made between the linked parties. As the central reservoir of available information, the Data Server is the Node to which all of the client Nodes connect.

Once the Data Server is running, clients may initiate RPC links with it, then begin negotiations about producing and / or consuming various items of data. There are no predefined data items known to the Data Server - consequently clients have to describe the data item that they intend to produce or consume, and the server has to determine whether to allow such production or consumption to take place.

Any number of clients may choose to consume a single data item, but only one client may produce the item at a time - this is to ensure consistency of data. There are various rules which allow a client to take over production of a data item from another client, without violating this approach of having only a single source active at any time.

There are no restrictions about the order in which the production or consumption of a given data item may be registered – it is quite acceptable to have one or more interests declared in a data item, before having the actual production of that item declared. Remember that we are dealing here with the negotiations which precede the actual production or consumption of data – not the actual production or consumption itself, which may only take place after the negotiations have been completed for the given data item.

A single client may produce and / or consume as many data items as it wishes. Similarly there is no conceptual limit placed on the number of clients that may register with the server.

Theory of the producer/consumer model

In this model, the server provides information calculated or measured by the system's (producer) processes to whichever (consumer) processes require it. The list of data message types is expandable, allowing for data supplied to the server by future or third party processes.

The content of the data messages is configurable by the client processes. For example, if one process provides gyro and radio navigation information and another provides GPS information, then:

- a client process requiring only gyro information need not be burdened with extraneous radio navigation information; and
- a client process requiring radio and GPS navigation information does not have to make two requests.

Data request rates are different for different client processes, and some request rates will be slower than, or out of phase with, the update rate of the data messages. Consequently, some client processes will be interested in the most recent instances of certain data, some in the next sequence from their previous receipt, and some in random access to all instances of the data. To cope with this, a tri-partite *Producer*, *Consumer* and *Broker* model is used.

Producers of information declare to the *Broker* (i.e. the server) that they intend to produce data items of a named type, and supply new values for those items asynchronously. *Consumers* (i.e. client processes) may interrogate the broker to discover what named data types exist and may, as a result of this, declare an interest in the data item. The storage used by the broker for the actual values is determined by the needs of the most demanding consumer, and falls into the following four categories:

- *New*, meaning that the consumer requires the broker to remember the latest value for the data item, but only if the consumer has not already received it;
- *Latest*, meaning that the consumer is only interested in the latest value for the data item;
- *Sequential*, meaning that the consumer requires the broker to remember all values of the data item until the consumer has received them; and

- *Any*, meaning that the consumer requires the broker to remember all values of the data item until notified otherwise.

Data is not time-stamped by the broker - it is up to the producers to provide data structures that include time of data. The broker is only aware of the sequence in which data item values were provided.

Two producers are not allowed to produce the same information.

The amount of storage required by the broker is potentially very large (a binning system, for example, may require *all* of a survey's data in *Any* mode). At the same time, access to the data must be speedy for pseudo-real time processes interested in *Latest* or fast reading *Sequential* accesses. To allow for this, the general scheme implemented is:

- The latest values of all items for which *Latest* interest is registered are held in RAM.
- While available RAM allows, items with *Sequential* interest registered will have all values up to the oldest still unread by a *Sequential* consumer held in RAM. RAM failure will cause the *oldest* unread to be spilled to disk, as, although this may result in some thrashing (as the oldest values will be the first to be read), it is safer in the case where a consumer has died.
- Items not covered by the above (i.e. values older than the oldest unread *Sequential*, but still potentially required by *Any* consumers, plus *Sequential* overflow) will be stored on disk.

Chapter 5 - Controlling Data Server Debugging (Advanced)

This chapter gives information on the `admin_dsn` utility program.

Data server debugging can be controlled without restarting the data server node. This is achieved by means of a utility program called 'admin_dsn'. This program is *not* for normal operation of the system.

To operate:

- admin_dsn With no parameters this will report options and exit.
- admin_dsn 0 Disables debug.
- admin_dsn 1 Enables debug to standard error.
- admin_dsn 1 <filename>
 Enables debug to named file.