IADS Real Time
Data Source Interface

January 2016, v2.0

SYMVIONICS Document SSD-IADS-021

SYMVIONICS, Inc.
Telemetry Systems

Table of Contents

1.0 Introduction

The purpose of this document is to describe how to develop an interface that feeds data to IADS along with how to test and troubleshoot the development effort. This document is part of an overall data source development kit that includes an example data source program with source code.

Section 2 of this document provides a specification describing the overall data source architecture including connection, protocol and format requirements along with rules on providing data. Section 3 describes an example data source program that is available as part of the development kit which can be useful for further understanding and guidance. Section 4 includes instructions on how to use an IADS product named RT Station to check out the interconnections between the data source and IADS Server along with viewing the data on an IADS Client display. Section 5 gives tips on troubleshooting potential problems that may arise during development and checkout.

2.0 Data Source Specification

2.1 Data Source Architecture

The real time data source architecture is one that provides data packets to the IADS Server at as close to fixed rates as possible. Since IADS is a data driven architecture the more consistent the rate the packets are fed to the IADS Server the better to provide a smooth data flow. The recommended data packet frequency is 10-20 milliseconds. To compensate for network and/or other system level delays a capability to buffer up data packets is recommended at the data source to provide some flexibility for potential data delivery delays in order to prevent data overflow/loss between the data source and the IADS Server. This buffering architecture has the advantage of allowing for some “rubber-banding” in the downstream processing without losing data at the data source.

2.1.1 Data Source Socket Interface

To communicate properly with the IADS Server, the data source must be set up as a TCP/IP socket server. The data source will first perform a handshake that specifies the byte order and format of the packets then will begin sending data packets. This protocol is a one way communication going from data source to IADS Server. After the initial handshake the data source will continually send data packets (preferably in a blocked write mode) to the IADS Server. These messages are recommended to be sent at a frequency of 10-20 milliseconds. The message size can vary between data packets so in order to maintain the packet rate you may need to send packets containing only time parameter samples in cases where data parameter rates are low.

Typical usage of the data source is to keep the source running and allow the IADS Server to perform multiple connections over an extended period of time. In order to accomplish this functionality another feature of the data source should be to allow reconnections from the IADS Server without having to restart the data source application.

2.1.2 Handshake Protocol

Upon initial connection to the data source, the IADS Server expects to receive two handshake messages describing aspects of the data source environment. First a one-byte message is expected that defines the byte order of all subsequent messages. The codes to specify the byte order are as follows:

Little Endian = 1

Big Endian = 2

Secondly a four-byte message is expected defining the code of the format of all subsequent data packets. The data packets must not vary from this specified format and must also conform to formatting specifications as defined in the next two sections. There are currently two supported packet formats with the following codes:

Tag/value pair format = 100

Tag/size/value format = 101

Note: Tag/size/value format only supports Little Endian byte order.

Section 2.1.4 will provide more detail on packet format content and field definitions.

2.1.3 Data Packet Header Format

Each data packet from the data source has a header that contains various record and status information followed by a body that contains tag/value pairs. Each header contains 8 32-bit fields (32 bytes) described as follows:

Field	*name*	*Description*
Field 0	Message Size	Total size of header and body (Field 0 non-inclusive)
Field 1	Sequence Number	Message sequence counter
Field 2	Packets Sent	Total number of data packets sent to IADS Server
Field 3	Data loss/Overflow	Total number of data loss/overflow occurrences
Field 4-7	Dummy	Currently unused fields

Except for field 0 (Message Size) and field 1 (Sequence Number) all other fields are essentially unused or optional fields used to describe additional data source status.

Calculating the Message Size field consists of adding the remaining portion of the header (28 bytes) to the entire size of the packet body. For example if the packet body size is 1200 bytes then the Message Size field should contain the value 1228.

See Appendix A for a block diagram of the message format including the header.

2.1.4 Data Packet Body Format

Currently there are two supported data packet body formats.

The first packet body format (code 100) contains sets of tag/value pairs consisting of 16-bit tag fields and 32-bit value fields. A tag is an integer that uniquely identifies a particular parameter. The values are the data associated with each tag instance. The format of a single tag/value pair in 32-bit form is as follows:

Tag1 (16-bit)
Tag2 (16-bit)
Value1 (32-bit)
Value2 (32-bit)

The body consists of **(n)** consecutive sets of these tag/value pairs. Therefore the size of each message will consist of 28 bytes of header (message size field is non-inclusive) plus **(n)** times 12 bytes of body. This also means there are a total of **(n)** times 2 parameters per message.

The second supported packet body format (code 101) contains sets of tag/size/value sets consisting of 32-bit fields containing the integer parameter identifier followed by the data unit size in bytes followed by the data value. The format of a single tag/size/value set is as follows:

Tag (32-bit)

Size (32-bit)

Value (number of bytes specified in Size field)

The body consists of **(n)** consecutive sets of these tag/size/value sets. Therefore the size of each message will consist of 28 bytes of header (message size field is non-inclusive) plus **(n)** times (8 + size) bytes of body where size is the size in bytes of the data associated with each particular tag. Currently this interface does not support variable size data for a particular tag. This also means there are a total of **(n)** parameters per message.

See Appendix A for a block diagram of the supported message formats.

2.1.5 Packet Content Notes

A primary requirement for parameters contained within the stream of data packets is that the order of the data samples coming out of the data source be chronological (time sequential) for a particular parameter. Each periodic parameter (i.e. parameter with sample rate greater than 0) is expected to have a consistent interval (with allowances for some minor rubber-banding) in the data stream correlating to the sample rate specified in the parameter definition file. No pre-alignment of data across parameters is assumed (i.e. no manipulation of the data is required prior to entering IADS).

Time parameters are required to be part of the data stream. Ideally the sample rate of the time parameters should be greater than or equal to the highest sample rate of the data parameters. The time words should be interleaved in the data packets such that each sample of a particular data parameter has a unique time stamp. The following is an example that illustrates a valid time/data sequence inside a packet where P1 and P2 are samples from 2 different data parameters and P2 is half the rate of P1. T1 and T2 are the upper and lower words respectively from the time parameter (see Section 2.3.1 for more detail on time word format):

T1/T2/P1/T1/T2/P1/P2/T1/T2/P1/T1/T2/P1/P2/T1/T2/P1/T1/T2/P1/P2…

Low sample rates on time parameters can have side effects. Since IADS is basically a data driven system the update rates of the IADS Client displays can be affected by the sample rate of the time parameters. Time parameter rates lower than about 50 samples per second may show a ‘stuttering’ behavior on the client displays. See Section 2.3.1 for more detail on time parameters.

2.2 IADS Server Setup

In order for the IADS Server to operate properly a parameter definition file must exist that provides information on how to process the packet contents. The parameter definition file is also known as the PRN file and typically has a .prn extension on the file name but is not required. The details of that file are described as follows:

2.2.1 Parameter Definition File

The IADS Server requires a file to exist prior to system startup that defines information on parameters expected to be received in the data stream from the data source. The set of information includes tag id, parameter name, sample rate and data

format (e.g. integer, float, unsigned integer, etc.) See Appendix B for typical entries in the file.

The first field corresponds to an integer tag identifier to uniquely associate a value with a parameter in the data stream. The second field represents the name of the parameter as specified in the IADS configuration file *ParameterDefaults* table. The third field corresponds to the expected sample rate in samples per second that the parameter will be received from the data source. The sample rates can be integer or floating point. Sample rates of 0 or 0.0 denote aperiodic data. The next field identifies the format representation of the value coming from the data source. The format codes currently supported are as follows:

Data Format	*Description*	*Code Value*
32-bit integer	Integer	0
32-bit unsigned integer	Discrete	1
32-bit single precision floating point	Float	2
64-bit integer	Long	3
64-bit unsigned integer	Ulong	4
64-bit double precision floating point	Double	5
Binary objects	Blob	7

BLOB data can be identified as binary data of any size aligned on byte boundaries.

There is also extended information that can be added to parameter definition entries. The general format of these extended entries is as follows:

Key = Value

Key is a reserved keyword that is recognized by the IADS Server to represent a specific piece of parameter information and Value is the value associated with the keyword. Currently there are two supported extended information keywords: “DataSize=n” and “SystemParamType = type” the latter of which is described further in Section 4.

Currently the IADS Server does not support variable sized samples within a single tag.

See Appendix B for an example Parameter Definition file

2.3 System Parameters

Following are parameters needed for the IADS Server to fully operate properly. These are system-based parameters used for processing other data parameters or for presenting status information. There are two types of system parameters, time parameters and decom status parameters. Time parameters are required to be included as part of the overall parameter set but decom status parameters are optional. The following subsections describe these parameter types:

2.3.1 Time Parameters

Time is represented as a 64-bit word in units of nanoseconds consisting of the time offset from the beginning of the year. As an example if IRIG time is set at 001:01:00:00.000 then the 64-bit time value would be 3600000000000 (i.e. one hour offset from the beginning of the year). The protocol required to transfer time via the data packets consists of splitting the time word into two 32-bit words. The upper 32-bit word must be identified in the parameter definitions file by appending the SystemParamType = MajorTime to the entry of the parameter to be used as the high order time word. The lower 32-bit word must be identified by appending SystemParamType = MinorTime to the parameter definition file entry of the parameter to be used as the low order time word. The sequence of the time words in the packet should be the upper time word followed by the lower time word. See the data flow example in section 2.1.5 where the upper word is represented as T1 and the lower word is represented as T2.

Ideally the sample rate of the time parameters should be greater than or equal to the highest sample rate of the data parameters. The time words should be interleaved in the data packets such that each sample of a particular data parameter has a unique time stamp.

Low sample rates on time parameters can have side effects. Since IADS is basically a data driven system the IADS Client display update rates can be affected by the sample rate of the time parameters. Time parameter rates lower than about 50 samples per second may show a 'stuttering' behavior on the client displays.

2.3.2 Decom Status Parameters

Decom Status is another system parameter used by the IADS Server to obtain data stream information such as sync loss. This parameter is identified in the parameter definition file by using the extended information property "DecomStatus" (see Appendix B for the data format of this parameter). The "DecomStatus" parameter will be defined in the IADS Client display using the default naming convention of _IadsDecomStatus(**n**)_. Where (**n**) is the stream number for multiple PCM stream setups. These parameters are automatically created in the IADS Configuration file upon startup and can be used by the IADS Client along with pre-defined derived functions for display purposes. These parameters are also used for informational purposes by the IADS Operator Console in a real time environment. Even though these parameters are not required in the stream, definition in the parameter definition file (see Section 3.1) is recommended. The currently supported decom status format is shown in Appendix C.

The following is a parameter definition file excerpt showing both the Time and Decom Status system parameter definitions.

```
1 DecomStatus    50.0    2 SystemParamType = DecomStatus
2 Param1         50.0    2
3 Param2         50.0    2
4 Param3         50.0    2
5 Param4         50.0    2
6 TimeUpperWord  1000.0  1 SystemParamType = MajorTime
7 TimeLowerWord  1000.0  1 SystemParamType = MinorTime
```

3.0 Example Data Source Program

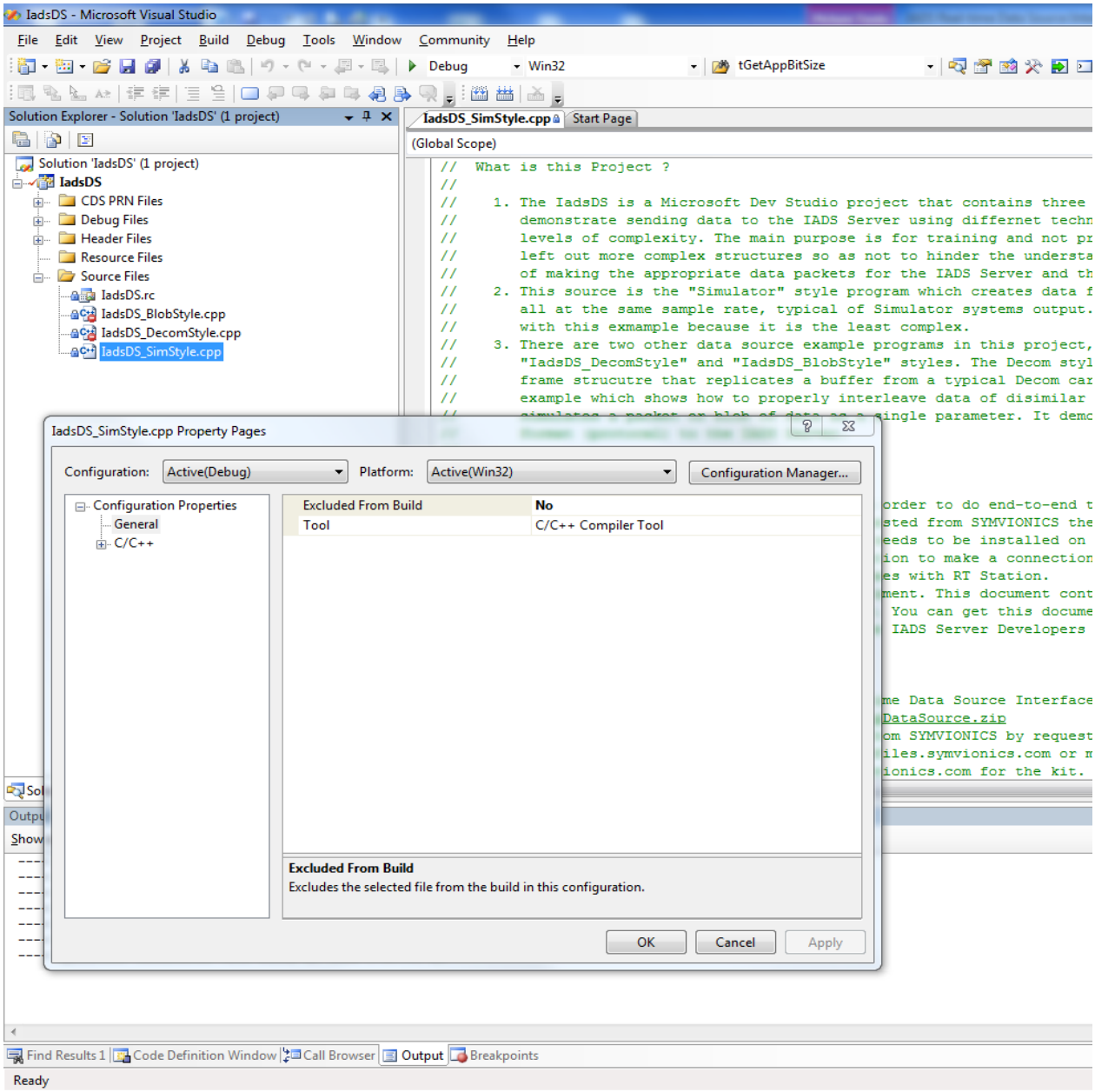
The purpose of the example IADS Data Source program is to provide a better understanding and guidance on the specifics of developing an interface to communicate with the IADS Server. The program initially waits for an IADS Server to connect and then sends data packets containing simulated data. The program also allows for reconnections to the IADS Server without re-launching the application which is a useful feature of the data source.

The IADS Data Source program is available either by requesting the IADS Data Source Developers Kit from SYMVIONICS, Inc. or downloading from the Programming Examples page on the SYMVIONICS web site (http://iads.SYMVIONICS.com/programs.html) by selecting the IADS Data Source option under the section named Data Processing Examples.

3.1 IADS Data Source Project

The IADS Data Source program is a Microsoft Visual Studio 2005 project written in C++ that contains three example programs demonstrating how to output the different packet formats available along with various methods of inserting data inside the packets. The project also contains three parameter definition files that describe the parameter specifications for each example. For more background information on packet setup and communication protocol along with details on parameter definition files see Section 2.0 of this document.

In order to specify which example program to apply, open up the project in Visual Studio and go to the Solution Explorer then right click on one of the source files named IadsDS_SimStyle.cpp, IadsDS_DecomStyle.cpp or IadsDS_BlobStyle.cpp and select Properties. Then in the Property Pages dialog go to the *Excluded From Build* entry located under Configuration Properties->General and specify **No** to include the file or **Yes** to exclude the file. Make sure only one of the cpp files is set to **No** before building the project.



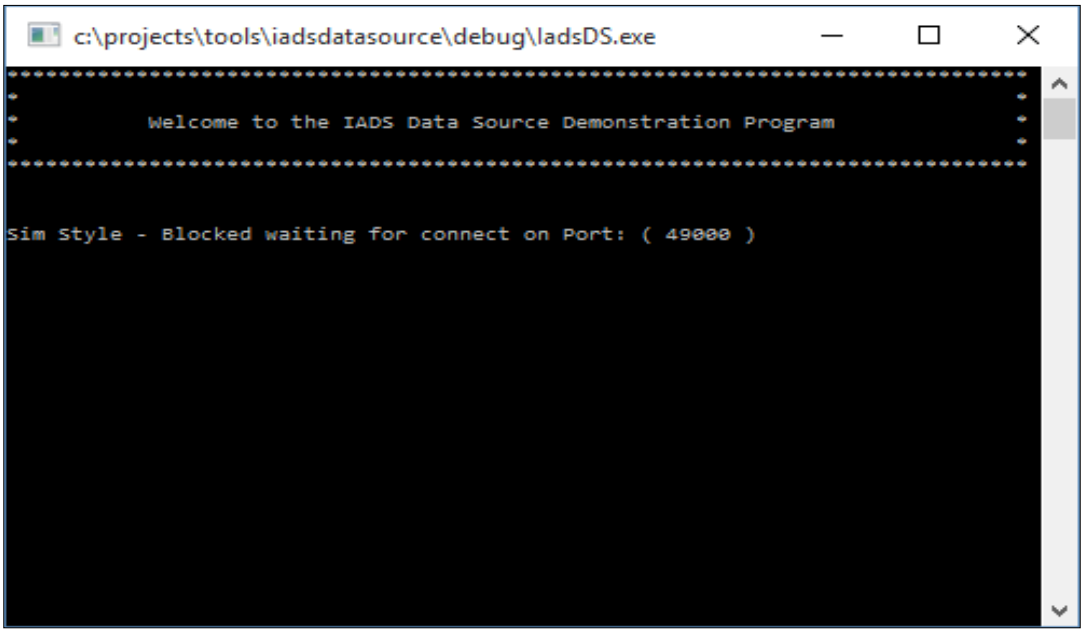
The different example programs are described as follows:

1) Simulator Style (IadsDS_SimStyle.cpp) - This program sends four data parameters along with time words to the IADS Server. The data parameters are all at the same sample rate which is typical of simulator output. Since this is the simplest case we recommend that you start with this example. To setup a project build, the IadsDS_SimStyle.cpp should be the only cpp file where the *Excluded From Buil*d property is set to **No**. The parameter definition file associated with this program is named IadsDS.prn.SimStyle and located in the main project directory. Note that this program applies packet format 100 (tag/tag/value/value).

2) Decom Style (IadsDS_DecomStyle.cpp) - This program sends five data parameters along with time words to the IADS Server. The data parameters are at different sample rates which is typical of decom-based systems. This provides a more complex example showing how to populate packets using differing sample rates. To setup a project for build, the IadsDS_DecomStyle.cpp should be the only cpp file where the *Excluded From Build* property is set to **No**. The parameter definition file associated with this program is named IadsDS.prn.DecomStyle and located in the main project directory. Note that this program applies packet format 100 (tag/tag/value/value).
3) Blob Style (IadsDS_BlobStyle.cpp) - This program sends one Blob parameter along with time words to the IADS Server. The Blob parameter contains four floating point parameters. To setup a project for build, the IadsDS_BlobStyle.cpp should be the only cpp file where the *Excluded From Build* property is set to **No**. The parameter definition file associated with this program is named IadsDS.prn.BlobStyle and located in the main project directory. Note that this program applies packet format 101 (tag/size/value).

3.2 Running the IADS Data Source Program

The IADS Data Source program can be run either within the Visual Studio environment or by creating a shortcut on the Windows desktop that points to the program's executable file (IadsDS.exe). Each version of the example program launches a command window and then goes into a state that waits for the IADS Server to connect.



The next section provides instructions on how to use IADS to test the data source interface.

4.0 IADS Real Time Station

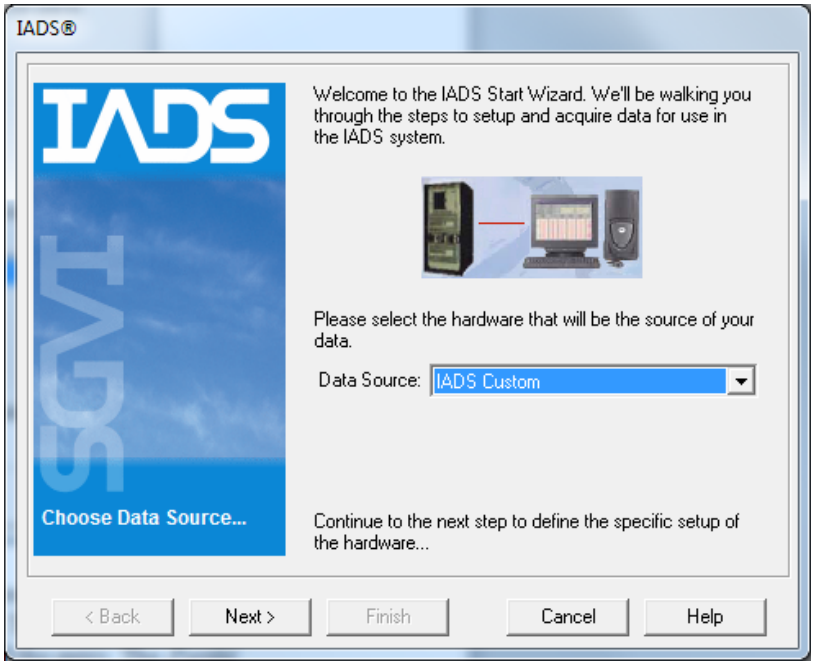
To test the data source interface we recommend using the IADS Real Time Station (RT Station) product in order to perform communication protocol and data flow verification activities. RT Station is an installable application that includes both the IADS Server and IADS Client display subsystems so that data from the data source program can be delivered and viewed in IADS.

The RT Station installation package is available either by purchasing the product from SYMVIONICS, Inc. (Part numbers are IADS-TELEM-RTSTATION-1 or IADS-TELEM-BASE-TPP) or by requesting the IADS Data Source Developers Kit (IADS-TELEM-DEV)

4.1 Running RT Station

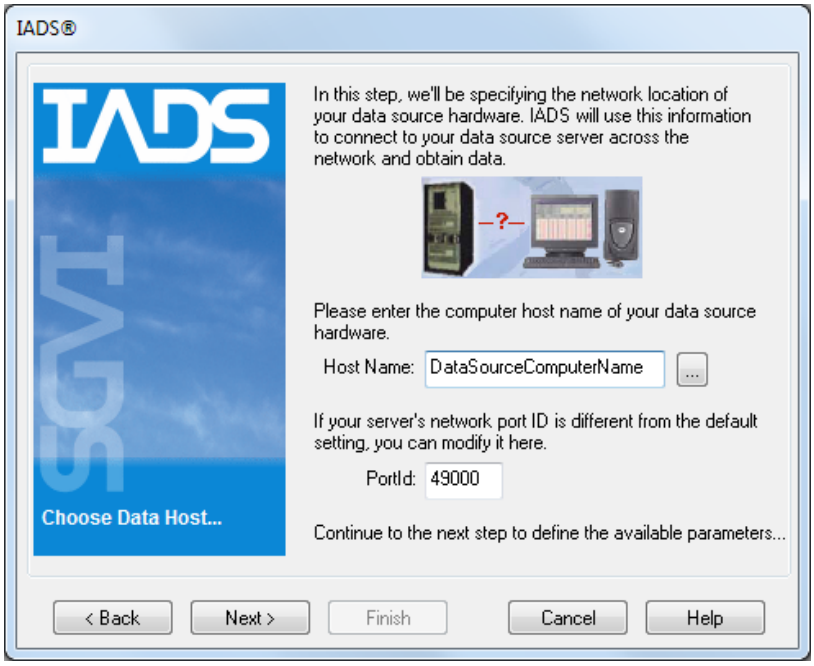
Running RT Station brings up a start wizard that will guide you through the process of selecting setup information that describes how to connect to the data source program and specify the parameter definition file. The startup steps are as follows:

1) Make sure the data source program is running and waiting to connect to IADS.
2) Double click the **IADS Real Time Station** icon on the Windows desktop.
3) On the Choose Data Source page select the **IADS Custom** option from the drop down menu in the *Data Source* field. Click **Next** to continue.

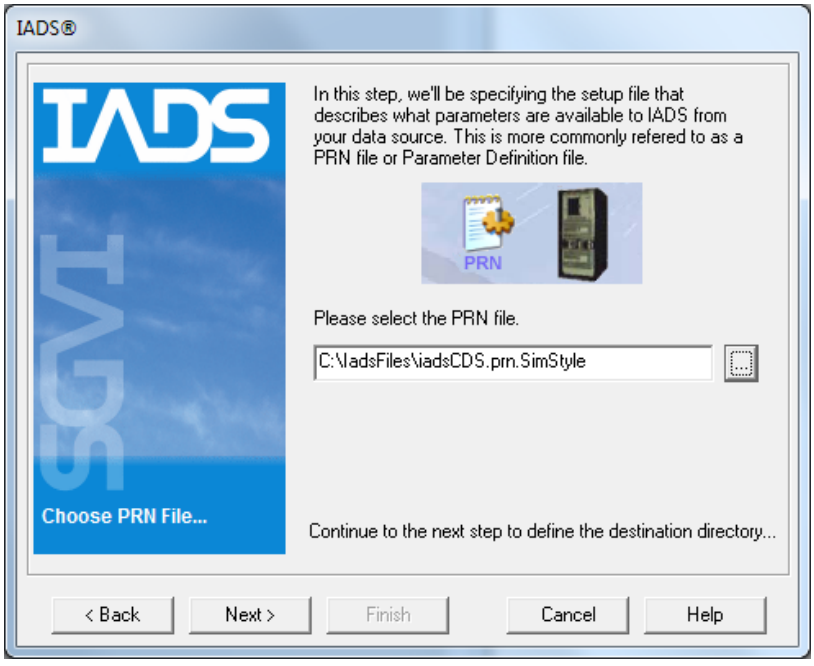


4) On the Choose Data Host page enter the name or IP address of the computer running the data source program in the *Host Name* entry. This can be entered manually or selected via the browse button on the right of the entry. The *PortId* entry defaults to 49000 which is the initial setup in the example data source

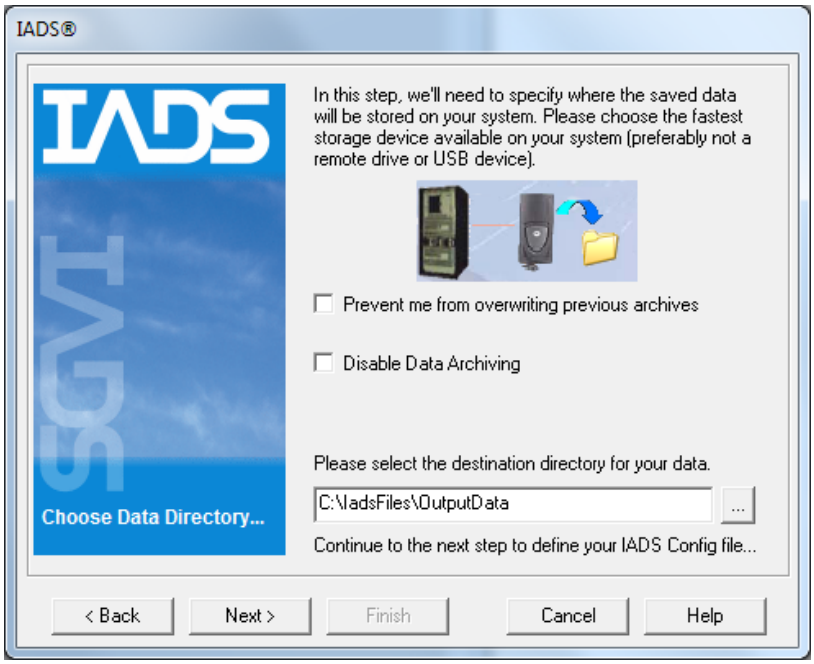
program. This field can be edited to specify the port id that is available on the data source for connection. Click **Next** to continue.



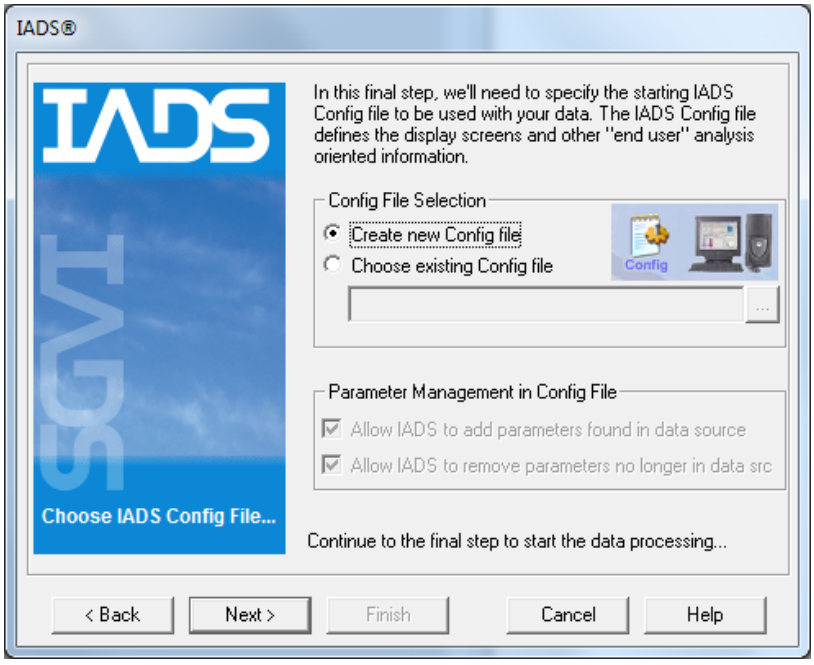
5) On the Choose PRN File page select the parameter definition file that contains the parameter specifications of the data source output. There is a browse button available on the right of the entry to assist in locating the file. If you are running one of the sample programs the matching parameter definition files are in the IADS Data Source project location. Click **Next** to continue.



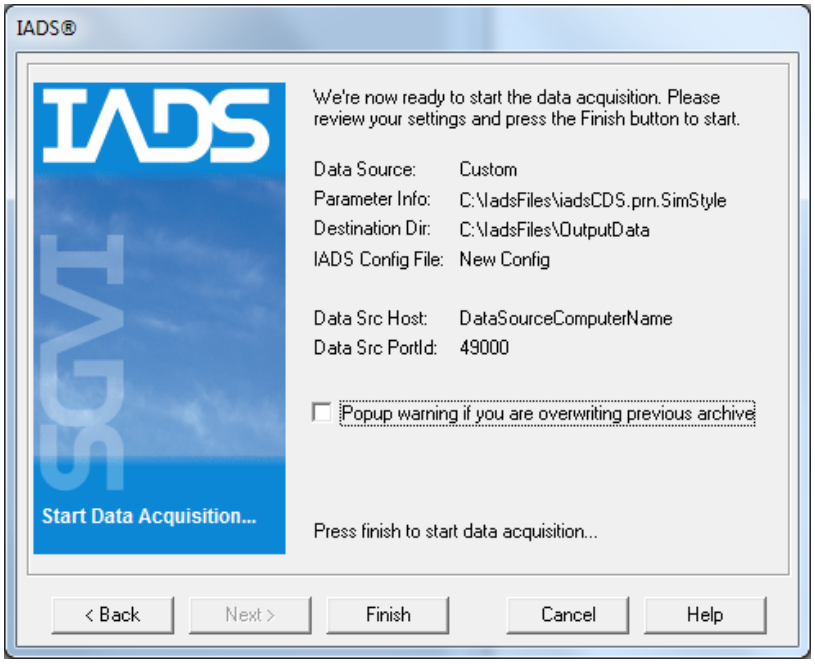
6) On the Choose Data Directory page select the destination folder for your IADS data storage files. A browse button is available on the right of the entry to assist in locating the directory. Click **Next** to continue.



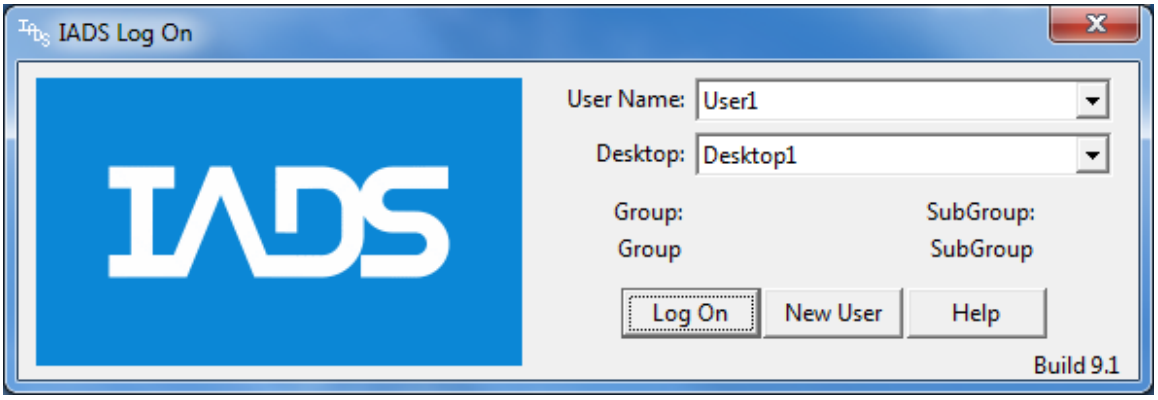
7) On the Choose IADS Config File page select the **Create new Config file** option within the *Config File Selection* section. This will automatically create a new IADS Configuration File from scratch that contains the parameters specified in the parameter definition file you selected earlier in the wizard. Click **Next** to continue.



8) On the Start Data Acquisition page review the settings and click **Finish** to start IADS.



At this point RT Station will connect to the data source and start ingesting and processing data packets. The IADS Client application will then be launched and you will be prompted to startup the client via the IADS Log On dialog. Select the predefined user name User1 and desktop name Desktop1 then click the **Log On** button (This step is automatic if User1 and Desktop1 are the only options.) This user and desktop setup contains a blank Analysis Window (Window1) which acts as a palette for data displays to be added.



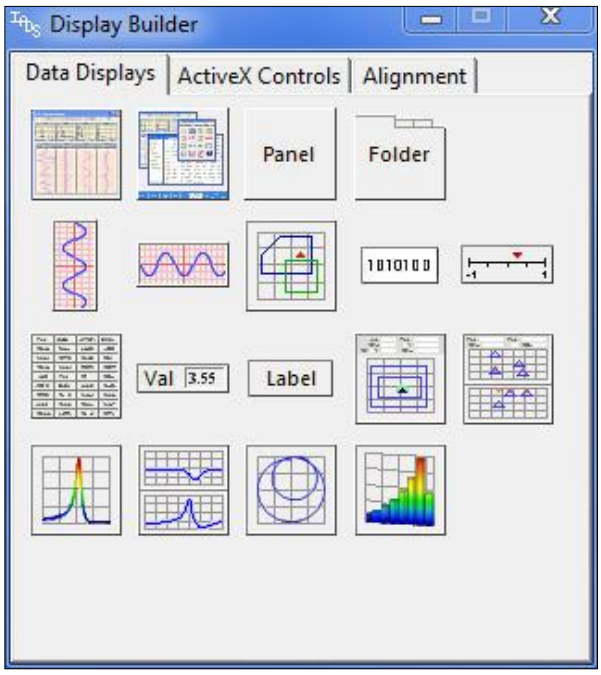
4.2 Creating Displays in IADS

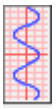
In order to view the data being delivered from the data source you need to create displays within the IADS Client application. IADS displays are created using icons in the Display Builder shown below. The Display Builder button is located in the far right portion of the IADS Dashboard. Use the Window1 analysis window to house the displays. The steps to build up displays are as follows:

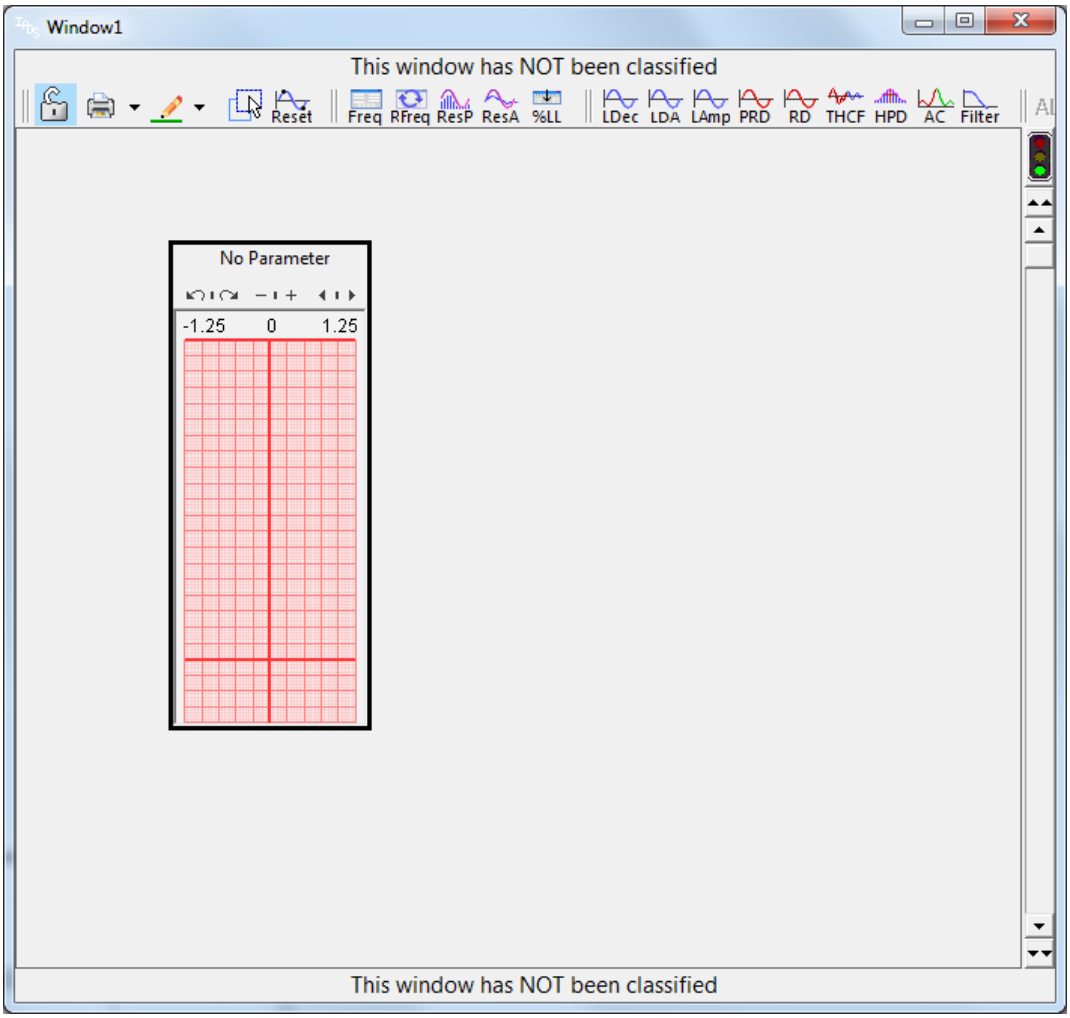
4.2.1 Creating an IADS Display

1) On the Dashboard click the **Display Builder** button.

2) On the Display Builder dialog click the **Data Displays** tab. A selection of display types is presented.

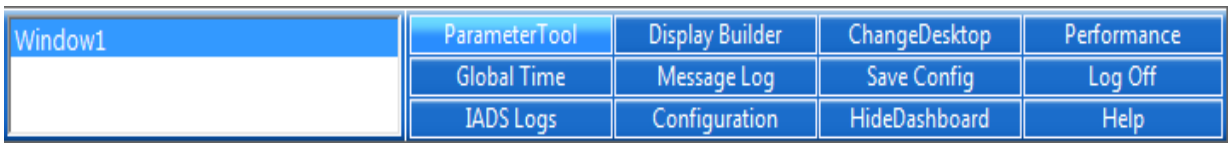


3) On the Data Displays tab click on the **Vertical Stripchart** icon  then hold down the left mouse button and drag onto Window1. This will create a new instance of a strip chart display inside the window.

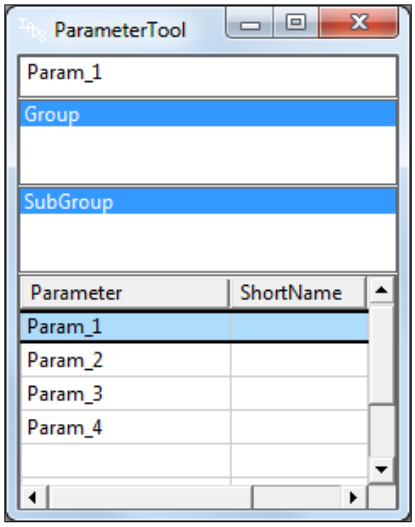


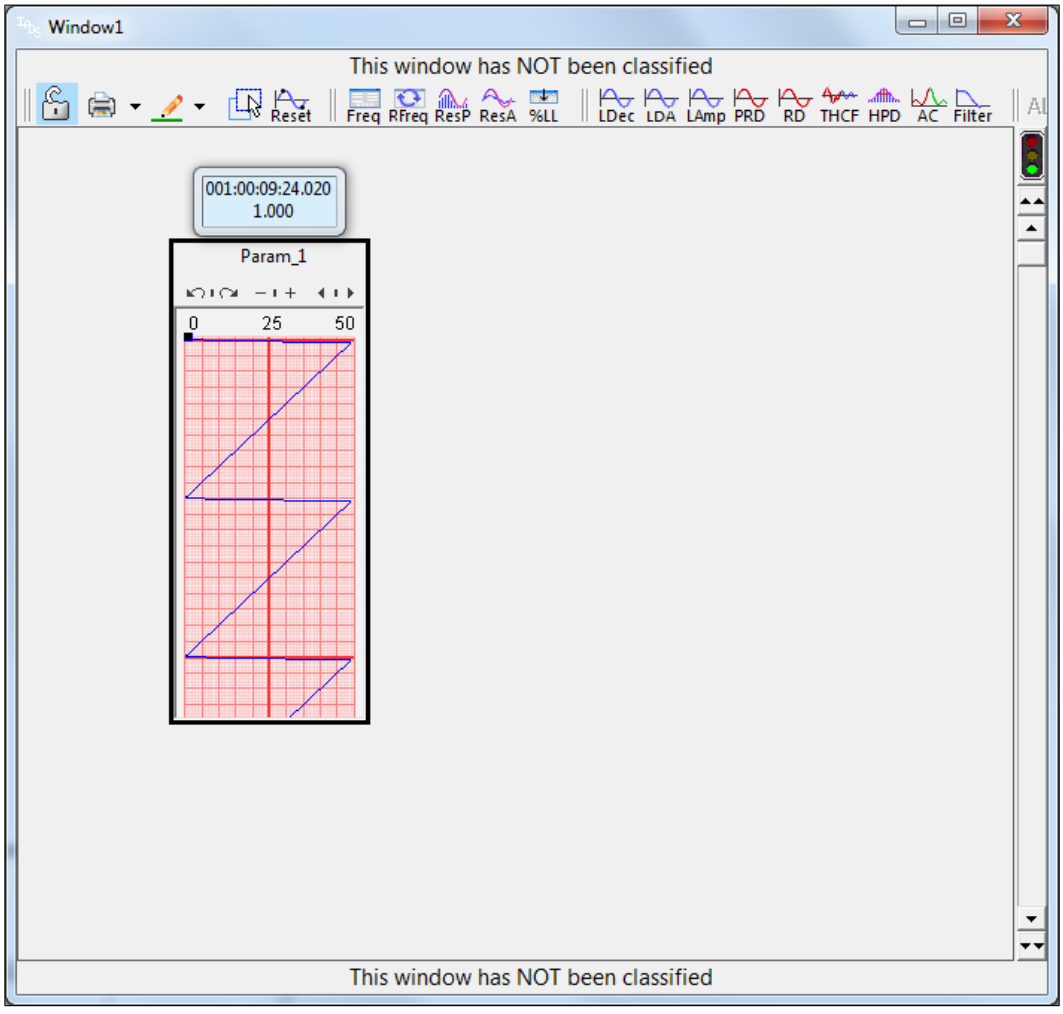
4.2.2 Adding Parameters to an IADS Display

1) On the Dashboard click the **Parameter Tool** button.



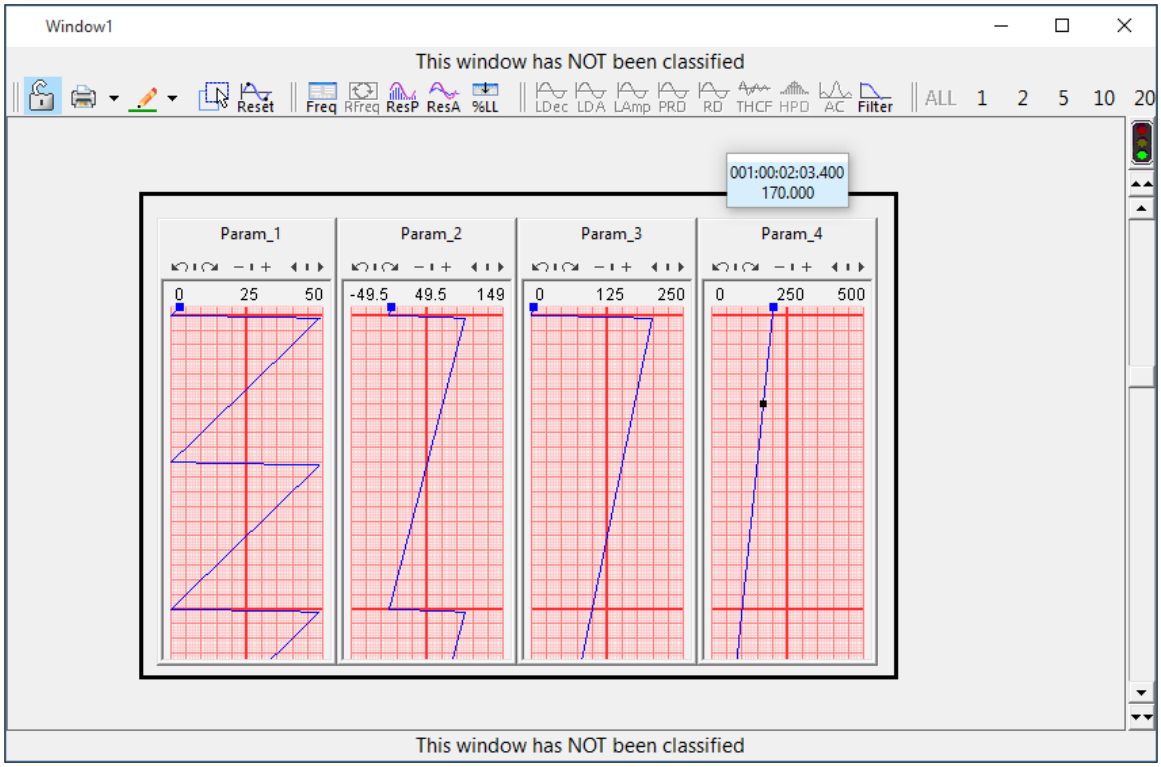
2) On the Parameter Tool select a parameter then hold down the left mouse button and drag onto the strip chart display. Select Value in the popup options after dropping the parameter onto the display (i.e. releasing the left mouse button).



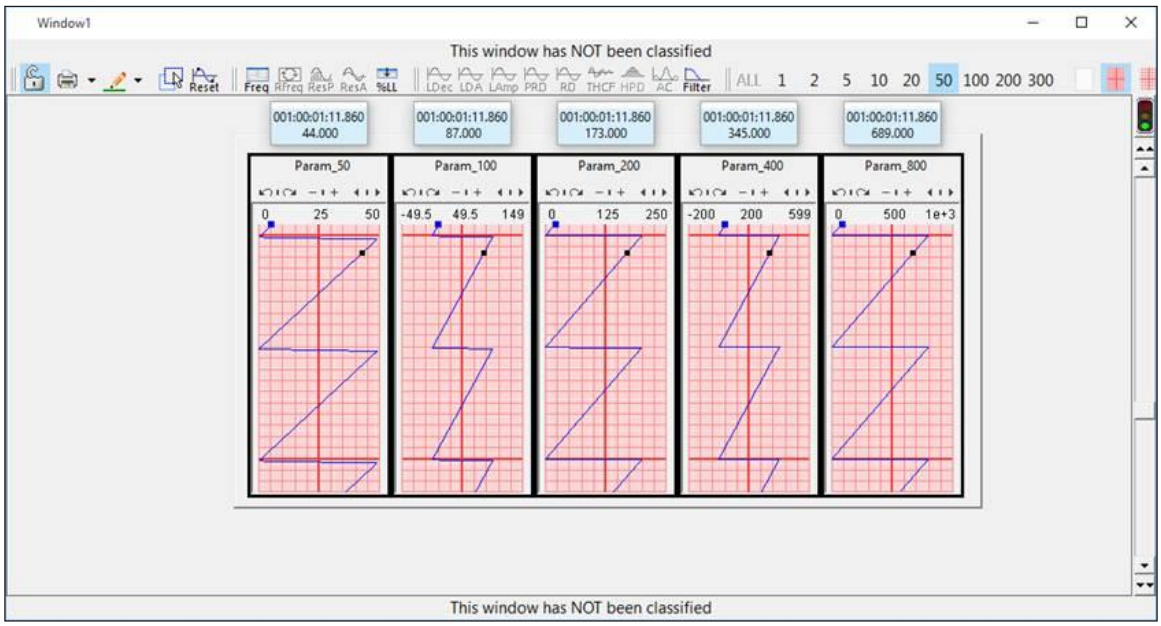


The following are example windows showing data from the three sample programs contained in the IADS Data Source project:

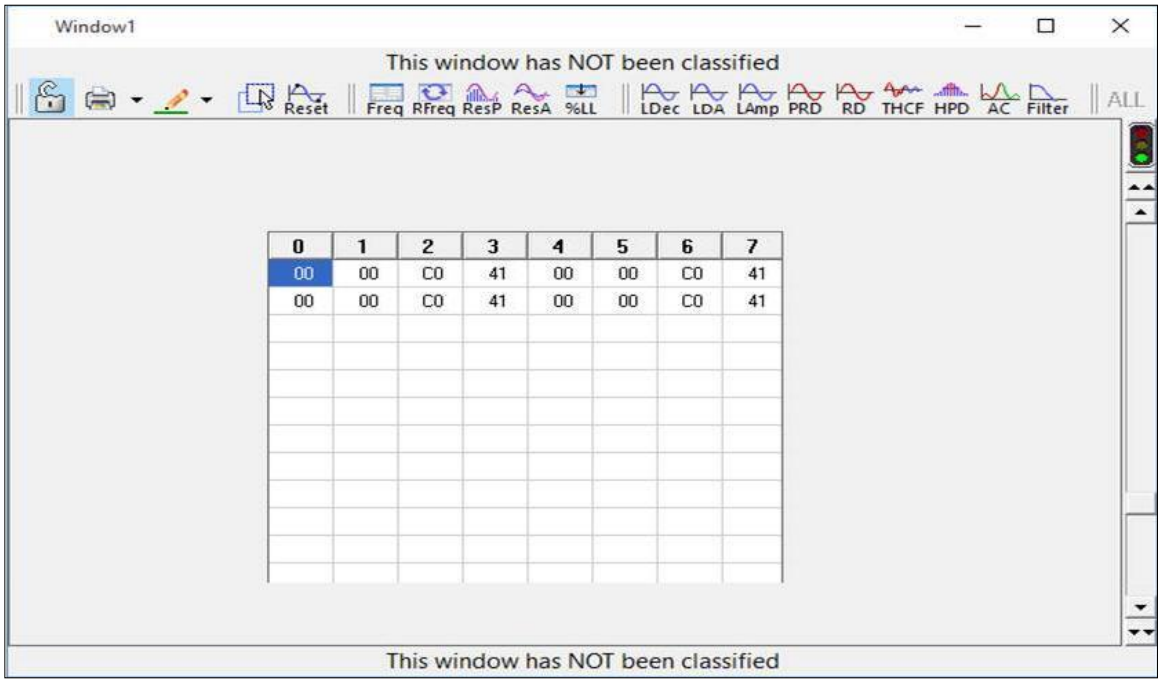
Simulator Style



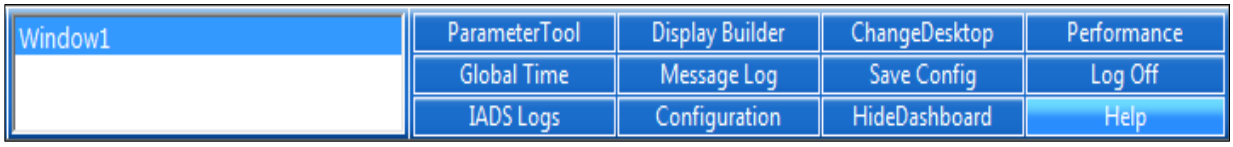
Decom Style



Blob Style



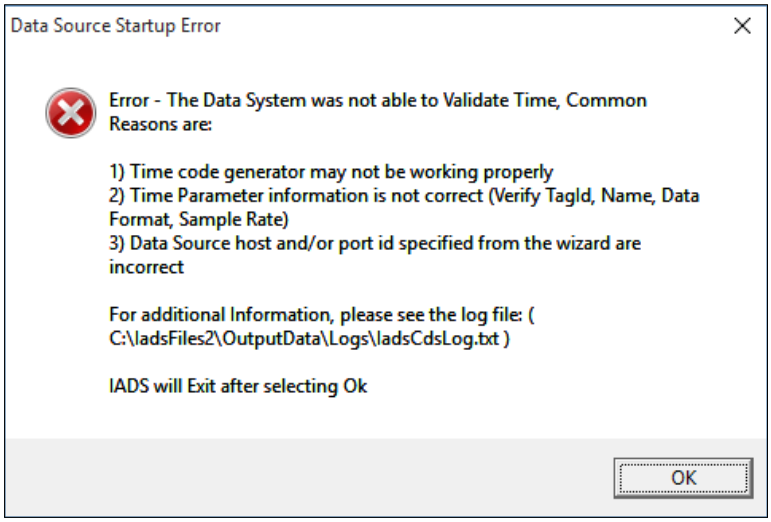
More detail on display types and general client functionality can be found in the IADS Client Help system which can be accessed via the lower right-most button on the Dashboard.



5.0 Troubleshooting

5.1 Validating Time

When the IADS Server connects to the data source and starts receiving packets it goes through a brief period of verifying that time parameter values are increasing at the expected increment. A common problem with a new data source interface is that the IADS Server may not successfully validate time during startup because the time values are not increasing at an increment based on the time word sample rate. For example if the rate of the time parameter is 1000 samples per second the IADS Server will expect time value increments of 1 millisecond per sample. Any time decrements or increments over 2x the expected rate will cause a validation failure. If time does not successfully validate a dialog containing possible reasons for failure will pop up as follows:



Typical reasons for time validation failures are as follows:

1) One or both time parameter tag ids are not found in the data. In this case verify that the parameter definition file that was entered during wizard startup contains the correct tag ids for the upper and lower time parameters. Verify that the data source is actually placing time parameter samples within the data packets. Check that the data source is truly sending data packets to the IADS Server.
2) Invalid time sequence errors. In this case verify the sample rate of the time parameters in the parameter definition file are correct. Verify there are no upstream errors occurring from the time source...e.g. Time Code Generator. Time values that seem corrupted (e.g. IRIG day values outside the range of 1-366) could mean that there is a packet misalignment...verify that the packet size specified in the 1st word of the packet header corresponds to the actual size of the remaining portion of the header along with the packet payload size.

A source of information that assists in determining reasons for validation errors is a file named “timeOut0.txt” located in the Logs folder under the IADS data directory that was specified during wizard startup. This file contains time values of each time word received by the IADS Server during the time validation period. An empty file means that

one or both time parameter tag ids were not sensed during the validation period (see reason 1 above). The file is also valuable for obtaining more detail on invalid time sequence errors (see reason 2 above). An example format of the “timeOut0.txt” file is as follows:

001:00:00:00.020 (86400020000000)

001:00:00:00.040 (86400040000000)

001:00:00:00.060 (86400060000000)

001:00:00:00.080 (86400080000000)

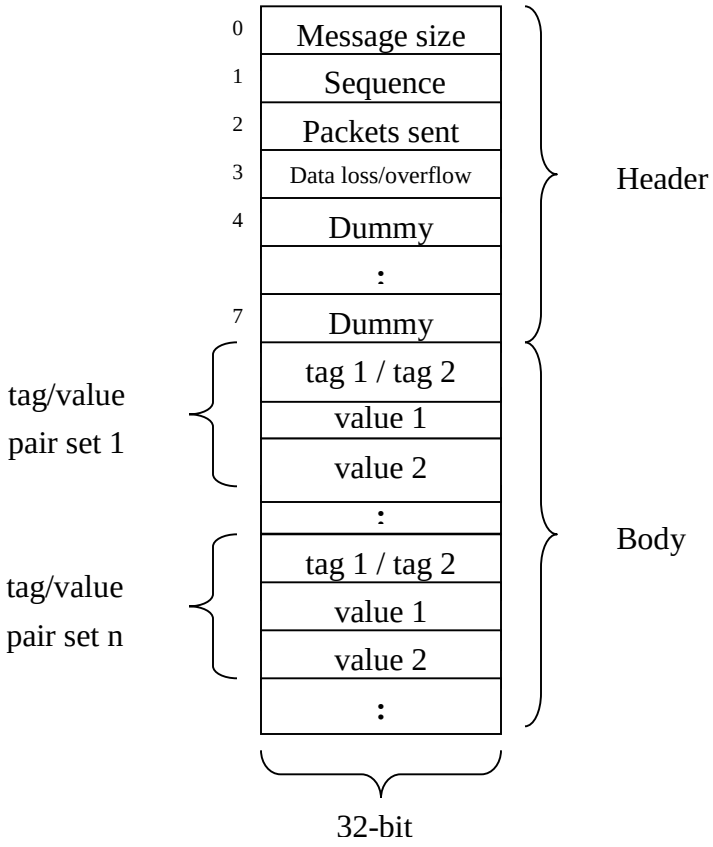
001:00:00:00.100 (86400100000000)

The 1st column is the IRIG representation of the time values and the 2nd column is the 64-bit integer representation of the values (in nanoseconds). In this example the time is incrementing by 20 milliseconds per sample which should correspond to a 50 sample per second rate for the time parameters specified in the parameter definitions file

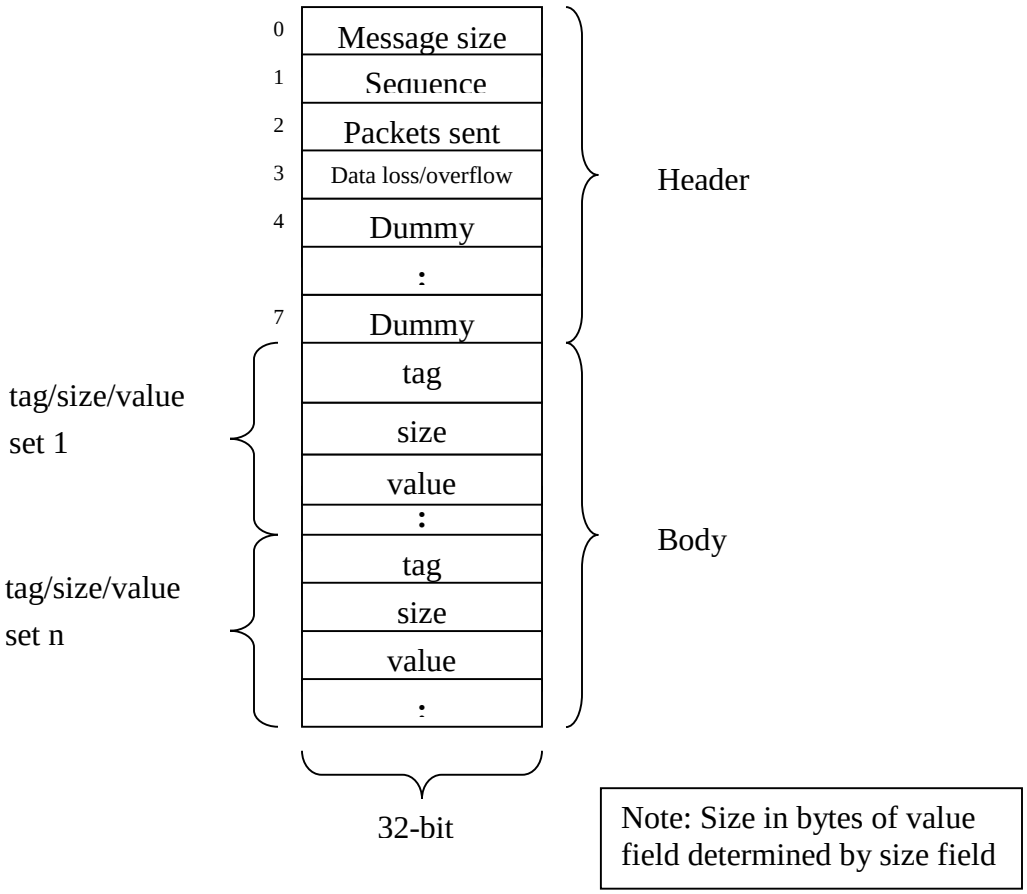
For further troubleshooting assistance please zip up the entire Logs folder located in the IADS data directory and send to iadssupport@SYMVIONICS.com to help in analyzing the issue.

APPENDIX A

IADS Interface Message Format 1



IADS Interface Message Format 2



APPENDIX B

Sample Parameter Definition File

```
1   TimeUpperWord  1000.0          1 SystemParamType = MajorTime
2   TimeLowerWord  1000.0          1 SystemParamType = MinorTime
3   PARAMETER1     12.330334596    2
4   PARAMETER2     24.6606691919   2
5   PARAMETER3     49.3213383838   2
6   PARAMETER4     98.6426767677   2
7   PARAMETER5     197.285353535   2
8   PARAMETER6     394.570707071   2
9   PARAMETER7     789.141414141   2
10  PARAMETERBLOB  10.0            7 DataSize = 92
100 DECOMSTATUS    789.141414141   1 SystemParamType = DecomStatus
```

APPENDIX C

IADS 32-bit Decom Status Parameter Format

31-6	5	4	3	2	1	0
SF(n+1)STAT- SF(n)STAT	SF2STAT	SF2STAT	SF1STAT	SF1STAT	FSTAT	FSTAT

Bits	Signal	Description
0-1	FSTAT	The Frame Status bits are decoded as follows: **1 0** 0 0 Lock 0 1 Check 1 0 Verify 1 1 Search
2-3	SF1STAT	Subframe 1 Status Bits are decoded as follows: **3 2** 0 0 Lock 0 1 Check 1 0 Verify 1 1 Search
4-5	SF2STAT	Subframe 2 Status Bits are decoded as follows: **5 4** 0 0 Lock 0 1 Check 1 0 Verify 1 1 Search