DOI 10.5162/ettc2022/7.1

Evolution of Data Exporting

Fatih HACIÖMEROĞLU[1], Egemen GÜÇLÜ[1], Mehmet KEKEÇ[1], Raşit UZUN[1]
[1] Helicopter Flight Test Instrumentation, Turkish Aerospace, ANKARA

Abstract:
This paper summarizes the data exporting methods and their evolution in the Helicopter Division in Turkish Aerospace. Initially, we could get away with using the utilities, which are provided by FTI System Supplier (Curtiss-Wright). These utilities helped to sustain data export process in first period of the project. However, along with increasing data export tasks, improvements in data exporting demands were eventually becoming inevitable. First, a utility is developed to exploit the batch process capability of the IADS Server license ("*Data Manager*" tool) in a multi-tasking fashion. This custom Data Export tool accelerated the exporting task and saved the day in the tight schedule of the T625 Project, even though it has its own substantial shortcomings namely creating a huge set of regular files without security or redundancy and not being a searchable database. In the meantime, a novel platform (named as "Optimus") is being developed in-house and is now starting to gain popularity among data clients.

Keywords: multiprocessing, multitasking, data export

Introduction

Before the T625 Project, the data export requests were not considerable and we used to utilize either the software suite of KAM-500 Data Acquisition System ("kFlashCard" of "KSM-500" software suite) or the GUI interface of the data conversion and visualization software ("IADS" of Curtiss-Wright) for data exporting. However, the requests in T625 Project, along with both the size of the collected data and the number of manoeuvers and data type combinations, forced us to look for a solution.

We decided to take advantage of a utility in IADS software suite, namely "IADS Data Manager Tool", which can be run from a terminal to export data, without opening an IADS Client session. A custom Data Export tool is developed to exploit this "batch processing" capability and run multiple data export processes in parallel. This solution, despite decreasing considerably the data export times, is by no means an "ideal" data sharing solution. It enabled us to keep up with project deadlines. The main shortcoming is that the data is not kept in a database through which one can query. Being able to make a query, which covers all flight data, is a must and can only be addressed by a dedicated platform, our Data Export Tool is an interim utility to accelerate export process. Other crucial areas which are not in the scope of our tool are data redundancy and data security issues.

Nevertheless, the Data Export Tool relieved time schedule pressure, enabled the continuation of the Project and more importantly from our point of view, increased consciousness for the necessity of a "Data Platform". In the time gained, a novel data platform (named as "Optimus") has been under development by the Artificial Intelligence Group. Here we only present the Data Export Tool in detail, as for Optimus, we provide solely a glimpse.

In the next section, we briefly describe the FTI System employed in T625 Project to provide an understanding about the scale of the FTI System which required novel solutions for data exporting. Then we mention about the problem faced in data exporting emerged by a flooding growth of data volume. Next we provide details about the developed tool which makes the most of the batch process capabilities of "IADS Data Manager" utility. Finally a glimpse into the Optimus platform is provided, which is getting popular and to which new capabilities are being added.

FTI System in T625

The Data Acquisition System in T625 comprises 8 KAM/500 DAQ's, 2 Ethernet Switches, 1 Ethernet Recorder and 2 Rotating DAQ Systems with contactless slip rings (for rotor & blade measurements).

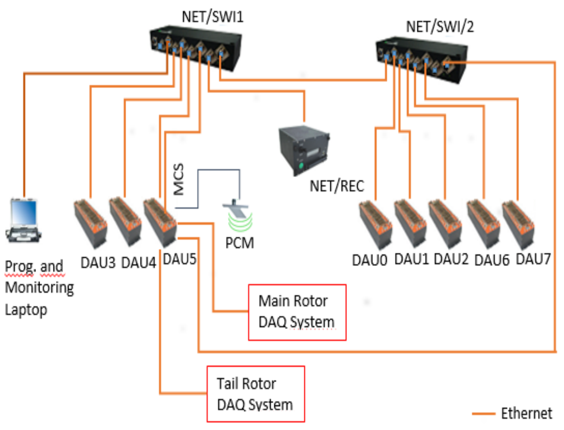


Figure 1 Instrumentation Architecture in T625

There are about 6000 raw parameters defined in the FTI System, about 1000 of them are for analog measurements (among which 150 of them are from rotating systems). We telemeter around 3200 out of the total 6000 raw parameters while the on-board recording rate of Ethernet packets is about 50 Mbps.

Raw parameters are sent from DAU's. The receiving side is responsible to perform the Engineering Unit derivations.

Issues in Data Exporting

Each test, either Ground or Flight Test, contains numerous "TestPoints" and collected data is grouped into numerous "DataGroups".

"*TestPoint*" (or Test Operation) is the name given to the maneouvre of interest which is marked with a start time and an end time. In other words it is a time slot in the whole test duration whose start and stop times are used to slice the data to indicate an interested section of test. Typical examples can be given as "Level Flight at 70 kts at 3000 ft AGL", "Deceleration from 70 kts to 40 kts at 3000 ft AGL", "Taxi 20 kts", "Left Coordinated Turn at 70 kts at 3000 ft AGL", etc. In addition to the executed testpoints, we define an enclosing TestPoint to slice all the time from beginning of Engine Start upto end of Engine power off, the aim of this enclosing testpoint is mark the region for data to be used for fatigue calculations.

"DataGroup" is the name given to the datasets containing parameters originated from same type of source and have same sample rate. Some typical examples can be given as "Pressure Parameters sampled at 256 Hz", "Pressure Parameters sampled at 4096 Hz", "Thermocouple Parameters sampled at 8 Hz", "RTD Parameters sampled at 8 Hz", "Arinc-429 Messages from ADC-1 sampled at 128 Hz", etc. Each one is a distinct DataGroup (note that the distinction comes both from data source type and also from the sample rate). There are over 100 DataGroups defined in T625 Project.

We are required to export data for each TestPoint and DataGroup combination as separate text file (Designers and Analyzers, who are the data customers, prefer the exported data to be in a text file such as "csv" format rather than in binary format). Using typical per flight quantities of 30 TestPoints and 100 DataGroups, it makes 3000 distinct TestPoint-DataGroup pairs and hence 3000 files to export.

IADS software is used for real time data monitoring during ground and flight tests. IADS uses its own data format (with extension "*iadsData*") in order to handle the demanding real time data processing feature. To obtain required csv files, we have been using IADS's export functionality available in an IADS Client or IADS RTStation licenses. They work great for small scale tests but they fall short when there are 3000 csv files to export. Since, to export data from an IADS Client, it is required to playback the IADS's test session, select the DataGroup and select the TestPoint manually. After a data group exported for all test points, another data group needs to be selected manually, which necessiates a user to wait in front on an IADS session during all the process and use the GUI.

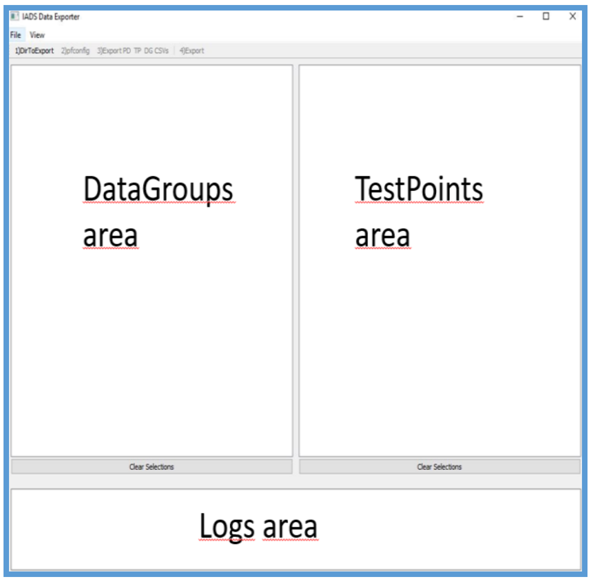
On the other hand, IADS's Server Licence provides a Data Manager Utility, with which one does not need to re-open (playback) a test session in IADS in order to export data. It is only needed to provide the configuration file in test session (the file named as "*pfConfig*" file) and Data Manager can use it to export data in desired format. What makes Data Manager worthy in our application is that it can also be run from command line. This batch capability was the enabler in the creation of the custom export tool. Our Data Export tool only needed to form a queue of the the Data Manager commands (one for each TestPoint and DataGroup) and call them in parallel, consecutively, until the exhaustion.

Custom Data Export Tool

First attempt was to write an application in Python which opened a new thread for each Data Export task (i.e. TestPoint-DataGroup pair). User can select the TestPoints and DataGroups and once exporting is started, the program was keeping track of the threads in order not to exhaust the CPU (obviously starting 2500 threads at once is a sure way to crash the operating system. Our CPU consists of 24 cores). This first program used the low-level thread functions of Python. For instance if alive thread number is set to be 10, the program would initiate first 10 threads and then keeps monitoring their execution, as soon as one thread finishes, the 11th export is initiated. Keeping track of active threads was a considerable part of the code. Each thread used to call an “os.system()” call in Python which creates a separate process. Therefore, our multithread application was actually running as a multiprocess application (Multiple threads can be run in a single core, so that user thinks that they are running simultaneously but in fact there is a fast switching undergoing. On the other hand, multiple processes are run in different cores so that the tasks are running simultaneously in the true meaning of the term). The code was difficult to maintain.

As the second version, we wanted to resort to Python’s built-in modules to handle multitasking (instead of keeping track of alive tasks ourselves). We chose **“*concurrent.futures*”** module in which a “pool” is created and all the export tasks are sent into this pool. We only have to adjust the pool size (a.k.a worker size) to determine how many simultaneous tasks we want the computer to run in separate cores. Our IADS Server computer has a 24 core processor, we generally used 12 as pool size. Using a high-level library such as “concurrent.futures” leads to cleaner code. As for the GUI, **“Qt5”** module is preferred (over “**tkinter**” module which was used in the first version).

The GUI has 4 toolbar buttons that have to be run in order, which is imposed by the program by activating and deactivating them as required.



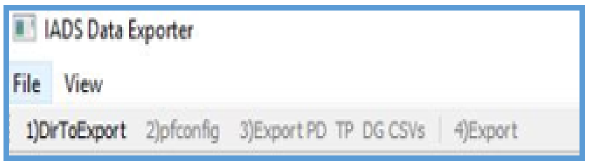
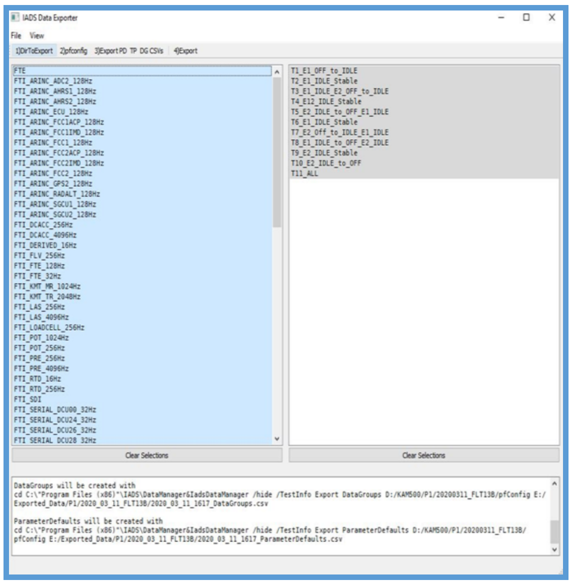


Figure 2 The Data Exporter GUI (top) Toolbar buttons (bottom)

<u>STEP 1) “DirToExport” Button</u>: First user selects the destination directory. A confirmation is presented in the Log area. The destination directory can be a local folder or it can be a folder which is in the intranet of the Company.

<u>STEP 2) “pfconfig” Button:</u> Then user selects the configuration file within the folder where IADS Session related to the test resides.

<u>STEP 3) Export TP DG PDs Button:</u> Pressing this button issues commands to Data Manager to export DataGroups file, TestPoints file and ParameterDefaults file (a fancy name of IADS used to refer to formulas (EU derivations)) to the export folder, as well as population of the DataGroups area and TestPoints area (as in Figure 3). User can select/deselect TestPoints and DataGroups to narrow down the export tasks if desired.



```
FTE
FTI_ARINC_ADC2_128Hz
FTI_ARINC_AHRS1_128Hz
FTI_ARINC_AHRS2_128Hz
FTI_ARINC_ECU_128Hz
FTI_ARINC_FCC1ACP_128Hz
FTI_ARINC_FCC1IMD_128Hz
FTI_ARINC_FCC1_128Hz
FTI_ARINC_FCC2ACP_128Hz
FTI_ARINC_FCC2IMD_128Hz
FTI_ARINC_FCC2_128Hz
FTI_ARINC_GPS2_128Hz
FTI_ARINC_RADALT_128Hz
FTI_ARINC_SGCU1_128Hz
FTI_ARINC_SGCU2_128Hz
FTI_DCACC_256Hz
FTI_DCACC_4096Hz
FTI_DERIVED_16Hz
FTI_FLV_256Hz
FTI_FTE_128Hz
FTI_FTE_32Hz
FTI_KMT_MR_1024Hz
FTI_KMT_TR_2048Hz
FTI_LAS_256Hz
FTI_LAS_4096Hz
FTI_LOADCELL_256Hz
FTI_POT_1024Hz
FTI_POT_256Hz
FTI_PRE_256Hz
FTI_PRE_4096Hz
FTI_RTD_16Hz
FTI_RTD_256Hz
FTI_SDI
FTI_SERIAL_DCU00_32Hz
FTI_SERIAL_DCU24_32Hz
FTI_SERIAL_DCU26_32Hz
FTI_SERIAL_DCU28_32Hz
```

```
T1_E1_OFF_to_IDLE
T2_E1_IDLE_Stable
T3_E1_IDLE_E2_OFF_to_IDLE
T4_E12_IDLE_Stable
T5_E2_IDLE_to_OFF_E1_IDLE
T6_E1_IDLE_Stable
T7_E2_Off_to_IDLE_E1_IDLE
T8_E1_IDLE_to_OFF_E2_IDLE
T9_E2_IDLE_Stable
T10_E2_IDLE_to_OFF
T11_ALL
```

```
DataGroups will be created with
cd C:\"Program Files (x86)"\IADS\DataManager&IadsDataManager /hide /TestInfo Export DataGroups D:/KAM500/P1/20200311_FLT13B/pfConfig E:/
Exported_Data/P1/2020_03_11_FLT13B/2020_03_11_1617_DataGroups.csv

ParameterDefaults will be created with
cd C:\"Program Files (x86)"\IADS\DataManager&IadsDataManager /hide /TestInfo Export ParameterDefaults D:/KAM500/P1/20200311_FLT13B/
pfConfig E:/Exported_Data/P1/2020_03_11_FLT13B/2020_03_11_1617_ParameterDefaults.csv
```

Figure 3 An example view after TestPoints and DataGroups are listed (from a very short test)

<u>STEP 4) "Export" Button:</u> This is the final step, where program starts running processes in parallel.

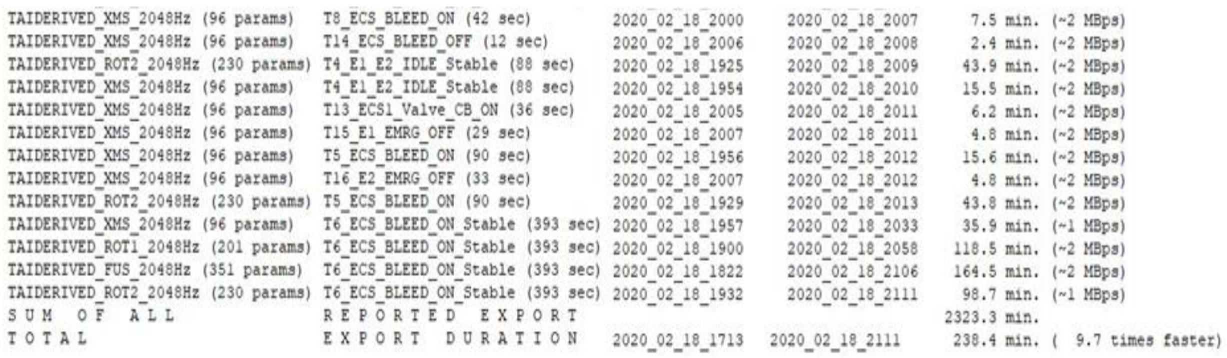
User can determine the number of processes that will run in parallel (i.e. number of workers). Generally, we use 12 workers, because using more would saturate CPU. In fact, this depends on the export location choice, which is another setting user can make. We named two distinct methods for export methods.

Export Method 1) Export each file directly to the destination folder in intranet.

Export Method 2) Export to a local temporary folder first and at the end copy all data to the destination folder in intranet.

For instance, if Method 1 is chosen, the copying of the file to intranet slows down and choosing a worker size of 12 would make the CPU to work at 50%. (Network speed bounds the CPU usage.) However if Method 2 is chosen, since writing to local drive is much faster than sending over Ethernet, the processes run faster and CPU utilization increases to about 80%, during the export process. Depending on the network speed and whether or not we have some other task to do in the Computer, user has the flexibility to choose worker size and export method.

During the execution, a log file is filled with information regarding the individual export tasks (as in Figure 4), such as the number of parameters in the DataGroup, the TestPoint time slot duration. There is also a metric showing how much we gained from parallelism. In this example, the parallel exporting process took about 10 times less than the time it would have taken, had all the exports been taken consecutively (without parallelism).



```
TAIDERIVED_XMS_2048Hz (96 params)   T8_ECS_BLEED_ON (42 sec)          2020_02_18_2000   2020_02_18_2007     7.5 min. (~2 MBps)
TAIDERIVED_XMS_2048Hz (96 params)   T14_ECS_BLEED_OFF (12 sec)        2020_02_18_2006   2020_02_18_2008     2.4 min. (~2 MBps)
TAIDERIVED_ROT2_2048Hz (230 params) T4_E1_E2_IDLE_Stable (88 sec)     2020_02_18_1925   2020_02_18_2009    43.9 min. (~2 MBps)
TAIDERIVED_XMS_2048Hz (96 params)   T4_E1_E2_IDLE_Stable (88 sec)     2020_02_18_1954   2020_02_18_2010    15.5 min. (~2 MBps)
TAIDERIVED_XMS_2048Hz (96 params)   T13_ECS1_Valve_CB_ON (36 sec)     2020_02_18_2005   2020_02_18_2011     6.2 min. (~2 MBps)
TAIDERIVED_XMS_2048Hz (96 params)   T15_E1_EMRG_OFF (29 sec)          2020_02_18_2007   2020_02_18_2011     4.8 min. (~2 MBps)
TAIDERIVED_XMS_2048Hz (96 params)   T5_ECS_BLEED_ON (90 sec)          2020_02_18_1956   2020_02_18_2012    15.6 min. (~2 MBps)
TAIDERIVED_XMS_2048Hz (96 params)   T16_E2_EMRG_OFF (33 sec)          2020_02_18_2007   2020_02_18_2012     4.8 min. (~2 MBps)
TAIDERIVED_ROT2_2048Hz (230 params) T5_ECS_BLEED_ON (90 sec)          2020_02_18_1929   2020_02_18_2013    43.8 min. (~2 MBps)
TAIDERIVED_XMS_2048Hz (96 params)   T6_ECS_BLEED_ON_Stable (393 sec)  2020_02_18_1957   2020_02_18_2033    35.9 min. (~1 MBps)
TAIDERIVED_ROT1_2048Hz (201 params) T6_ECS_BLEED_ON_Stable (393 sec)  2020_02_18_1900   2020_02_18_2058   118.5 min. (~2 MBps)
TAIDERIVED_FUS_2048Hz (351 params)  T6_ECS_BLEED_ON_Stable (393 sec)  2020_02_18_1822   2020_02_18_2106   164.5 min. (~2 MBps)
TAIDERIVED_ROT2_2048Hz (230 params) T6_ECS_BLEED_ON_Stable (393 sec)  2020_02_18_1932   2020_02_18_2111    98.7 min. (~1 MBps)
S U M   O F   A L L                 R E P O R T E D   E X P O R T                                        2323.3 min.
T O T A L                           E X P O R T   D U R A T I O N     2020_02_18_1713  2020_02_18_2111    238.4 min. (  9.7 times faster)
```

Figure 4 An example Log file, providing details of each export task.

Glimpse into the New Platform

The Optimus Platform, being developed by the Artificial Intelligence Group, saves the raw data (pcap files) in a computational cluster (providing built-in redundancy and security) and uses parallel processing techniques to perform EU conversions on the fly. Having saved only the raw data provides substantial storage area savings. This data storage and on-demand data conversion platform is aimed to be equipped with analysis functionalities.

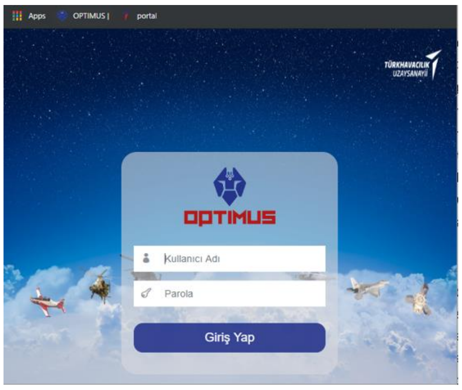


Figure 5 Login Screen of Optimus

Besides, basic statistics (such as minimum, maximum, average for each minute) about all parameters are stored in a separate database in Optimus (it is the first database (in the true sense of the word) used for flight data). This enables users to have a look at the data (as in Figure 6) and also more importantly to query flight data. This is especially worthy since the query can span over multiple flights, i.e. search is not limited to one flight. To illustrate, to investigate if some parameters exceeded some thresholds in all of the prototype flights can be almost instantly obtained by means of Optimus.

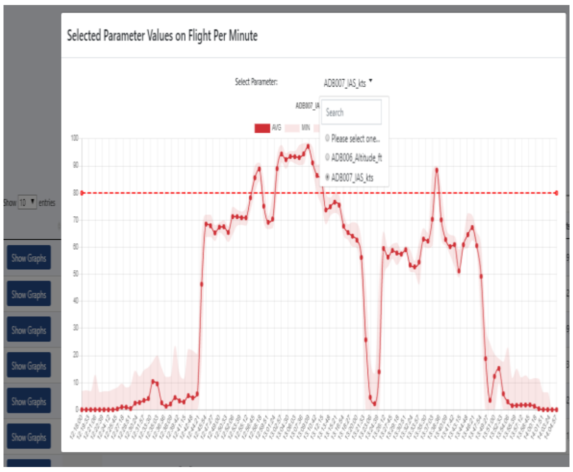


Figure 6 Example view of search capability in Optimus (Many flights are detected, user can plot the parameter (along with exceedance criteria)

Conclusion

A Data Export Tool is developed to overcome the increasing data export requests, where conventional methods based on Supplier utilities was getting insufficient. The created tool makes use of the batch processing capability of the IADS software using parallel processing techniques provided in Python. We have successfully used the tool in T625 Project. Nevertheless increasing frequency of the flights necessitated novel computer science techniques for data storage and processing, which led to development of a new platform. Without the Data Export Tool, we would not be able to meet the data expectations and we would not have time to have Optimus developed in parallel.

References

[1] www.xidml.org

[2] www.qt.io

[3] www.curtisswrightds.com