Aviation Software

DFT Database API

Prepared by: Toby Wicks, Software Engineer

Version 1.1

19 November 2010

Aviation Software

Table of Contents

Aviation Software

Overview

Document Overview

This document contains detailed specifications of the application programming interface (API) used to access DFT databases. The API provides support for accessing data stored from the following versions of DFT: DFT6.5, DFT7 and DFT8. This specification only details areas of the API directly related to the purpose of importing duty and flight data. Please forward any queries regarding the API to toby@aviationsoftware.com.au.

Contact Details

Robin Wicks	rob@aviationsoftware.com.au
Toby Wicks	toby@aviationsoftware.com.au
Phone	0419 909 340
Address	P.O. Box 36 Brooklyn Park SA 5032 Australia
Website	www.aviationsoftware.com.au

Aviation Software

Database Overview

DFT Packages

Aviation Software provides the following packages and modules - all of which are commonly referred to as DFT:

1. **DFT Standard** - provides tracking of duty, flight times, night flights, instrument approaches, currency checks, pilot details and configuration.
2. **DFT+Logbook** - provides all of the features provided in DFT Standard plus additional tracking of aircraft and logbook data. This package is sold as an additional module to the DFT Standard package. This package uses the same code base and version numbering as the DFT Standard package.
3. **Personal Logbook** - only provides tracking of aircraft and logbook data. It is considered a personal or electronic logbook and is typically used by private pilots. This package uses the same code base and version numbering as the DFT Standard package.

DFT Standard and **DFT+Logbook** are the packages most commonly purchased. All three packages utilise the same database structure. Blank data is written to the database for those features not supported by a particular package. For example, the Personal Logbook version does not allow the user to enter duty hours, however these fields still exist in the database and are set to blank (0000) values. However, there is a minor difference regarding storage of Aircraft Types in the DFT Standard and logbook versions. This difference is explained in detail in the API specification.

References to **DFT** in this document refer to all three of the above packages, unless otherwise stated.

File Structures

The DFT database structure consists of a number of flat-files located within one directory. Each directory is considered a unique, self-contained and separate database. Typically, the database is located in the C:\Program Files\DFT8 directory along with the program executable and dlls. However, DFT can be configured to utilise data from a central repository on a file server or any other directory location. In an office situation, DFT is typically setup to utilise a UNC or mapped drive file share so that each computer with DFT installed can access the central DFT database. Specifically, a command line argument can be passed to DFT8.EXE which specifies the full path to the database directory.

Pilot data is stored in 5 files each ending with the following extensions: 60D, 60L, 60C, 60A and 60R. Each file is prefixed with the name of the pilot. Each DFT database will also contain an Aircraft.dat, Admin.60C and GlobalOptions.dat file.

Listed below are the different files found in a DFT database:

File	Description
Aircraft.DAT	Contains all of the aircraft (call sign, engine type and configuration). All pilots can reference an aircraft in this table. This file is only relevant to the logbook versions of DFT. However, a blank Aircraft.dat file will be created for the DFT Standard package.
Admin.60C	An administrative file - similar to the GlobalOptions.dat file. For the purposes of importing DFT data this file is irrelevant.
GlobalOptions.DAT	Contains global settings pertaining to the running of DFT. For the purposes of importing DFT data this file is irrelevant.
<PilotName>.60A	Contains the past experience / brought forward hours for all aircraft types that the pilot has flown. This file is updated via the **Past Experience** setup screen. This file is only relevant to the logbook versions of DFT.
<PilotName>.60C	Contains all of the pilot's settings, password, currency checks, comments and CAO flight regulations. These settings are modified from the **Setup** and the **Currency Checks** screens. For the non-logbook versions this file also stores the pilot's 3 aircraft types for which flight hours are tracked.
<PilotName>.60D	Contains all of the pilot's duty, flight times, night flight and instrument approaches. Essentially all of the data that is displayed on the **Day Hours** tab.
<PilotName>.60L	Contains all of the pilot's logbook data as displayed on the **Logbook** tab. This file may exist even though the pilot has not entered any logbook data or does not have a registered version of DFT +Logbook or Personal Logbook. This file is only relevant to the logbook versions of DFT.
<PilotName>.60R	Contains all of the pilot's remarks for a particular day. This is the remarks field as displayed on the **Day Hours** tab. This is not to be confused with the Details field entered on the **Logbook** tab. The remarks typically contain the flight information when used with the DFT Standard package. In the logbook versions the the flight details are usually entered into the Details field in the logbook - however, the remarks field still may contain relevant information.

Below is an example of a typical directory listing, containing data for 2 pilots, John Smith and Marion West:

- Admin.60C
- Aircraft.dat
- GlobalOptions.Dat
- John Smith.60A
- John Smith.60C

- John Smith.60D
- John Smith.60L
- John Smith.60R
- Marion West.60A
- Marion West.60C
- Marion West.60D
- Marion West.60L
- Marion West.60R

Aviation Software

Database API

Overview

The database API is exposed via the DFTDB.DLL ActiveX DLL which contains classes and methods for reading and writing to a database. The DFTDB.DLL is utilised by the DFT8.EXE program to read and write to a database. The DFTDB.DLL is installed with the DFT program in the same directory as the DFT8.EXE. The DFTDB.DLL also contains classes related to business logic and algorithms that are not relevant to the purpose of importing data. Only the classes that are relevant to the importing of data have been detailed. A complete code listing detailing the relevant classes and properties is provided at the end of this document.

Requirements & Installation

For the purposes of importing data, a DFTDB.DLL from a logbook version of the software is required in order to be able to access the standard data and logbook data. A DFTDB.DLL from a standard (non-logbook) DFT package will not have the ability to access any logbook data.

To install the DFTDB.DLL without requiring the installation of DFT8:

1. Download and install the VB6 runtime library. It is available from the Microsoft website here: http://support.microsoft.com/kb/192461
2. Register the DLL using regsvr32. i.e. run the command: *regsvr32 C:\LocationOfDFTDBLibrary\DFTDB.DLL*
3. The DLL is now installed and can be utilised from Visual Studio, Visual Basic 6.0 or a compiled program.

Classes

DFTDB.DftDatabase Class

The DFTDatabase class is the root class in the object hierarchy. The class provides access to the pilots collection, aircraft collection and various other global data and settings. To open a database, the full path (UNC paths are permitted) to the directory is specified, together with an access key provided by Aviation Software and the customer's serial number. The access key defines which data can be read from the database and must be obtained from Aviation Software. If a particular feature is not available because of the access key a COMException with error code -2147221491 and a description 'Access is not available to this function ...' will be displayed when a disallowed method or property is accessed. Please contact Aviation Software if this occurs. For the purposes of importing data, the customer's serial number is not required - it is only useful for licensing purposes. An empty string can be passed for this argument.

In VB.NET the declaration is as follows:

```
Dim objDatabase As New DftDB.DftDatabase
objDatabase.OpenDatabase("C:\Program Files\Dft8", "AccessKey", strRegisteredCompanySerialNumber:="")
```

DFTDB.DftDatabase Properties

DFTDB.DftDatabase.LogbookMode Property	
Description	This property indicates whether the database contains logbook data. It is useful for determining what data should be imported. This property interrogates the database and determines whether the aircraft table/collection contains any aircraft data. If the aircraft table is empty, then the pilot data most likely does not contain any logbook data. The code example demonstrates the use of this property.

DFTDB.AircraftSet Class

The DFTDB.AircraftSet class provides access to a collection of DFTDB.Aircraft objects and is accessed from the DFTDB.DftDatabase.Aircraft property. The collection represents all of the aircraft setup in the database and contained within the Aircraft.DAT file. Each pilot logbook entry must reference an aircraft from this collection. The collection will always contain at least one aircraft. The API automatically creates a default aircraft with call sign C000 if no aircraft exist in the database.

The AircraftSet collection is only relevant for the logbook versions of DFT.

To enumerate through all of the aircraft in the database:

```
For Each objAircraft As DftDB.Aircraft In objDatabase.Aircraft
    Debug.Print("Call Sign: " & objAircraft.Registration)
Next
```

An aircraft can also be accessed from the AircraftSet collection using the aircraft's call sign:

```
Dim objAircraft As DftDB.Aircraft = objDatabase.Aircraft("CallSign")
```

DFTDB.Aircraft Class

The DFTDB.Aircraft class provides access to the aircraft's configuration and is accessed from the DFTDB.AircraftSet collection. Detailed below are the Aircraft properties:

DFTDB.Aircraft.Registration Property	
Description	Call sign / registration number that is used to uniquely identify the aircraft. The call sign must be unique.
Size / Limitations	20 Characters. Also defined by property DFTDB.DFTDatabase.Constants.AircraftRegistrationLength.

DFTDB.Aircraft.TheType Property	
Description	The aircraft's type. This field is used to determine a pilot's unique list of aircraft types.
Size / Limitations	20 Characters. Also defined by property DFTDB.DFTDatabase.Constants.AircraftTypeLength.

DFTDB.Aircraft.Engines Property	
Description	Returns an enum that indicates whether the aircraft is single or multi-engined.

DFTDB.Aircraft.EngineType Property	
Description	Returns an enum that indicates the aircraft's engine type - whether it is a piston or turbine engined aircraft. The TurboProp and TurboJet engine types are customer specific and generally not used.

DFTDB.Aircraft.IsDeletable Property	
Description	Read-only property that indicates whether the aircraft can be deleted. An aircraft can only be deleted if it is not being referenced by a pilot's logbook and is not the only aircraft remaining in the collection. There must always be one aircraft setup in the database.

DFTDB.Pilots Class

The DFTDB.Pilots class provides access to a collection of DFTDB.Pilot objects and is accessed from the DftDatabase.Pilots property. The collection represents all of the pilots contained within the database directory.

To enumerate through all of the pilots:

```
For Each objPilot As DftDB.Pilot In objDatabase.Pilots
    Debug.Print("Name: " & objPilot.Name)
Next
```

A pilot can also be accessed from the Pilots collection using the pilot's name. The pilot name is the name as displayed on the Pilot Summary screen. For example, to access the Pilot object for "John Smith":

```
Dim objPilot As DftDB.Pilot = objDatabase.Pilots("John Smith")
```

DFTDB.Pilot Class

The DFTDB.Pilot class provides access to all data associated with a pilot. Specifically, it provides access to a pilot's duty times, flight hours, instrument checks, logbook entries, currency checks and configuration options.

Detailed below are the pilot's settings and configuration options - details regarding duty times, flight hours, instrument checks, logbook entries and currency checks are covered in further sections.

DFTDB.Pilot.Name Property	
Description	Returns the name of the pilot as specified by the pilot's .60D file name. The .60D extension is **not** returned. This is the name as displayed on the Pilot Summary screen.
Size / Limitations	20 Characters. Also defined by property DFTDB.DFTDatabase.Constants.PilotNameLength.

DFTDB.Pilot.Password Property	
Description	Returns the pilot's password in an unencrypted format. A password is optional. If no password is specified an empty string is returned.

Size / Limitations	20 Characters. Also defined by property DFTDB.DFTDatabase.Constants.PilotPasswordMaxLength.

DFTDB.PilotDetails.FullName Property	
Description	The pilot's full name. This may be different from the **DFTDB.Pilot.Name** property described above. This field is optional and is for reference purposes only. This field is visible on the pilot's Setup screen.
Size / Limitations	80 Characters. Also defined by property DFTDB.DFTDatabase.Constants.PilotCurrencyDetailsLength.

DFTDB.PilotDetails.Seniority Property	
Description	The pilot's seniority. This field is optional and is for reference purposes only. This field is visible on the pilot's Setup screen.
Size / Limitations	80 Characters. Also defined by property DFTDB.DFTDatabase.Constants.PilotCurrencyDetailsLength.

DFTDB.PilotDetails.Address Property	
Description	The pilot's full address (street, suburb, postcode, state). This field is optional and is for reference purposes only. This field is visible on the pilot's Setup screen.
Size / Limitations	80 Characters. Also defined by property DFTDB.DFTDatabase.Constants.PilotCurrencyDetailsLength.

DFTDB.PilotDetails.CommencedEmployment Property	
Description	The date that the pilot commenced employment. This field is optional and is for reference purposes only. This field is visible on the pilot's Setup screen.
Size / Limitations	80 Characters. Also defined by property DFTDB.DFTDatabase.Constants.PilotCurrencyDetailsLength.

This field is visible on the pilot's Setup screen

DFTDB.PilotDetails.ParentOrGuardian Property	
Description	The name of the pilot's parent or guardian. This field is optional and is for reference purposes only. This field is visible on the pilot's Setup screen.
Size / Limitations	80 Characters. Also defined by property DFTDB.DFTDatabase.Constants.PilotCurrencyDetailsLength.

DFTDB.PilotDetails.Phone Property	
Description	The pilot's phone / mobile phone number. This field is optional and is for reference purposes only. This field is visible on the pilot's Setup screen.
Size / Limitations	80 Characters. Also defined by property DFTDB.DFTDatabase.Constants.PilotCurrencyDetailsLength.

DFTDB.PilotDetails.Fax Property	
Description	The pilot's fax number. This field is optional and is for reference purposes only. This field is visible on the pilot's Setup screen.
Size / Limitations	80 Characters. Also defined by property DFTDB.DFTDatabase.Constants.PilotCurrencyDetailsLength.

DFTDB.PilotDetails.Comments Property	
Description	Any comments relating to the pilot. This field is optional and is for reference purposes only. This field is visible on the Currency Checks screen.
Size / Limitations	400 Characters. Also defined by property DFTDB.DFTDatabase.Constants.PilotCommentsLength.

DFTDB.PilotDetails.BaseOrLocation Property	
Description	The base or location of the pilot. This field is optional and for most customers used for reference purposes only. This field is visible on the pilot's Setup screen.
Size / Limitations	80 Characters. Also defined by property DFTDB.DFTDatabase.Constants.BaseOrLocationLength.

DFTDB.PilotSettings.CAOSetup Property	
Description	Specifies the regulations that the pilot adheres to. This is primarily of relevance regarding different flight time limitations and warnings. This field is visible on the pilot's Setup screen.

DFTDB.PilotSettings.FlightRules Property	
Description	Indicates whether the pilot adheres to Visual Flight Rules (VFR) or Instrument Flight Rules (IFR). This field is visible on the pilot's Setup screen.

DFTDB.PilotSettings.DutyCalculationMethod Property	
Description	Indicates the method by which the total duty is calculated. It can be one of the following: • dbDutyCalculationVFR - Indicates that the duty is calculated as the sum of all duty times • dbDutyCalculationIFR - Indicates that the duty is calculated as the sum of all duty times plus 50% of the rest period. • dbDutyCalculationDutiesPlusRestPeriod - Indicates that the duty is calculated as the sum of all duty times plus the rest period. This field is visible on the pilot's Setup screen.

DFTDB.PilotSettings.GPSUnits Property	
Description	Returns either 1 or 2 - indicating the number of GPS units that have been configured in the GPSName() array. Returns 1 if the first GPS name has been configured but the second GPS name is blank. Returns 2 if the second GPS name has been configured. If GPSUnits is 1 then the GPS approach (generally named RNAV or GPS npa) must be completed every 90 days. If GPSUnits is 2 then an approach on each GPS unit must be completed every 180 days. This field is visible on the pilot's Setup screen.

DFTDB.PilotSettings.GPSName(1 to 2) Property	
Description	Returns the name of the GPS unit to be tracked along with the other instrument approaches in the Day Hours tab. GPSName(1) will always return a valid GPS name and GPSName(2) may return an empty string. If GPSName(2) is not empty GPSUnits will return 2. This field is visible on the pilot's Setup screen.

DFTDB.PilotSettings.StartingFlightHours	
Description	Returns the pilot's brought forward hours as at a particular date (DFTDB.PilotSettings.StartingFlightHoursReferenceDate). This value is summed with the flight hours entered in DFT to calculate the pilot's total flight hours. This property is only relevant for the standard (non-logbook) version of DFT.

DFTDB.PilotSettings.StartingFlightHoursReferenceDate	
Description	Returns the reference date for which the DFTDB.PilotSettings.StartingFlightHours property refers. This property is only relevant for the standard (non-logbook) version of DFT.

DFTDB.PilotSettings.PastExperienceHoursTotalling Property	
Description	Indicates how the pilot's past experience is entered. There are two options: 1. dbPastExperienceTotalsBroughtForward - Indicates that the pilot's brought forward hours for Instructor, ICUS, Command, Co-Pilot, Dual, and Instrument Ground and Flight are known as totals and are not known for each aircraft type flown. The brought forward hours are accessed from the objPilot.AircraftTypes.BroughtForward property. 2. dbPastExperienceAircraftTypeTotalsBroughtForward - Indicates that the pilot's brought forward hours for Instructor, ICUS, Command, Co-Pilot, Dual, and Instrument Ground and Flight are known for each aircraft type flown. The brought forward hours are accessed from the objPilot.AircraftTypes("TypeName") property. This property is only relevant for the logbook version of DFT.

DFTDB.PilotSettings.PastExperienceReferenceDate Property	
Description	Indicates the date upon which the aircraft type past experience / brought forward hours refers (see DFTDB.Pilot.AircraftTypes for further details). This property is only relevant for the logbook version of DFT.

DFTDB.PilotSettings.FirstEntryDate	
Description	Returns the date of the first entry made in either the Day Hours or Logbook tabs. This field can be used to determine the starting date from which to import data from the DFT database.

DFTDB.PilotSettings.LastEntryDate Property	
Description	Returns the date of the last entry made in either the Day Hours or Logbook tabs. This field can be used to determine the last date from which to import data from the DFT database.

DFTDB.PilotAircraftTypeSetDFT Class

The PilotAircraftTypeSetDFT class contains a collection of 4 PilotAircraftTypeDFT objects. Each object represents a unique aircraft type for which flight time tracking is calculated. The PilotAircraftTypesDFT collection is unique to each pilot as each pilot may track different aircraft types. The aircraft types are setup on the Setup screen on the Aircraft tab.

Flight hours can be split between the 3 aircraft types and the **Other** aircraft type by associating the flight hours with each aircraft type. For example, if 5 flight hours are completed for the day, 2 hours can be associated with aircraft type 1 and 3 hours associated with aircraft type 2. The flight hours are associated by using the drop-down list of aircraft types displayed underneath the flight time field on the Day Hours tab.

This class is only relevant for the standard (non-logbook) versions of DFT. For the logbook version the DFTDB.PilotAircraftTypes class should be used.

To enumerate through the 4 pilot's aircraft types:

```
For Each objAircraftType As DftDB.PilotAircraftTypeDFT In objPilot.AircraftTypesDFT
    Debug.Print("Type: " & objAircraftType.TheType)
Next
```

If the aircraft type name is known an AircraftTypeDFT object can be returned directly from the AircraftTypesDFT collection:

```
Dim objAircraftType As DftDB.PilotAircraftTypeDFT = objPilot.AircraftTypesDFT("C101")
```

DFTDB.PilotAircraftTypeDFT Class

The DFTDB.PilotAircraftTypeDFT class represents an aircraft type and details the aircraft's engine type, engine configuration and the pilot's brought forward hours.

This class is only relevant for the standard (non-logbook) versions of DFT. For the logbook version the DFTDB.PilotAircraftType class should be used.

Detailed below are all of the properties for the PilotAircraftTypeDFT class:

DFTDatabase.PilotAircraftTypeDFT.TheType Property	
Description	The aircraft type name that is being tracked. For example, C101, C121. If the name is not specified in the pilot's Setup screen then Aircraft1, Aircraft2 or Aircraft3 is returned.
Size / Limitations	15 Characters. Also defined by property DFTDB.DFTDatabase.Constants.AircraftTypeDFTMaxLength.

DFTDatabase.PilotAircraftTypeDFT.EngineType Property	
Description	Returns an enum that indicates the aircraft type's engine type - specifically whether it is utilises a piston or turbine engine. The TurboProp and TurboJet engine types are customer specific and generally not used.

DFTDatabase.PilotAircraftTypeDFT.EngineConfiguration Property	
Description	Returns an enum that indicates whether the aircraft is a single or multi-engined.

DFTDatabase.PilotAircraftTypeDFT.StartingHours Property	
Description	The flight time performed on this particular aircraft type as at a particular date (StartingHoursReferenceDate).
Size / Limitations	Double floating point value.

DFTDatabase.PilotAircraftTypeDFT.StartingHoursReferenceDate Property	
Description	The date for which the starting flight hours (StartingHours) refers.

DFTDB.PilotAircraftTypes Class

The PilotAircraftTypes class contains a collection of aircraft types tracked by a pilot and the past experience associated with each aircraft type or the past experience for all aircraft types. See the DFTDB.PilotSettings.PastExperienceHoursTotalling property for further information.

This collection is only relevant to the logbook version of DFT.

To enumerate through the pilot's aircraft type starting hours:

```
Select Case objPilot.Settings.PastExperienceHoursTotalling
  Case dbPastExperienceAircraftTypeTotalsBroughtForward
    For Each objAircraftType As DftDB.PilotAircraftType In objPilot.AircraftTypes
      Debug.Print("Starting Hours for Aircraft Type " & objAircraftType.TheType & ":")
      Debug.Print("Single Engine ICUS Day: " & objAircraftType.PastExperience.SingleEngineICUS.Day)
      Debug.Print("Single Engine ICUS Night: " & objAircraftType.PastExperience.SingleEngineICUS.Night)
      Debug.Print("Single Engine Dual Day: " & objAircraftType.PastExperience.SingleEngineDual.Day)
      etc.
    Next
```

```
    Case dbPastExperienceTotalsBroughtForward
        Debug.Print("Single Engine ICUS Day: " & objPilot.AircraftTypes.BroughtForward.SingleEngineICUS.Day)
        Debug.Print("Single Engine ICUS Night: " & objPilot.AircraftTypes.BroughtForward.SingleEngineICUS.Night)
        Debug.Print("Single Engine Dual Day: " & objPilot.AircraftTypes.BroughtForward.SingleEngineDual.Day)
        etc.
End Select
```

DFTDatabase.PilotAircraftTypes.BroughtForward Property	
Description	Returns a DFTDB.PilotFlightHours object that contains all of the starting hours when objPilot.Settings.PastExperienceHoursTotalling = dbPastExperienceTotalsBroughtForward and the past experience / brought forward hours is not known for each aircraft type flown. See the DFTDB.PilotSettings.PastExperienceHoursTotalling property for further information.

DFTDB.PilotAircraftType Class

The PilotAircraftType class contains the past experience / brought forward hours for a particular aircraft type. This class is only relevant when objPilot.Settings.PastExperienceHoursTotalling = dbPastExperienceAircraftTypeTotalsBroughtForward and for the logbook versions of DFT.

DFTDatabase.PilotAircraftType.TheType Property	
Description	The aircraft type name. For example, C101, C121.
Size / Limitations	20 Characters. Also defined by property DFTDB.DFTDatabase.Constants.AircraftTypeLength.

DFTDatabase.PilotAircraftType.PastExperience Property	
Description	Returns a DFTDB.PilotFlightHours object that contains all of the aircraft type's brought forward for Instructor, ICUS, Command, Co-Pilot, Dual, Instrument Ground and Instrument Flight.

DFTDB.PilotCurrencyChecks Class

The DFTDB.PilotCurrencyChecks class contains a collection of pilot currency checks - a collection of procedures that a pilot must frequently renew and maintain experience with. The currency checks are unique to a particular pilot. Each pilot can have a maximum of 50 currency checks.

To enumerate through the pilot's currency checks:

```
For Each objCheck As DftDB.PilotCurrencyCheck In objPilot.CurrencyChecks
    Debug.Print("Check: " & objCheck.Name)
    Debug.Print("Expiry Date: " & objCheck.ExpiryDate)
    Debug.Print("Comments: " & objCheck.Comments)
Next
```

DFTDB.PilotCurrencyCheck Class

The DFTDB.PilotCurrencyCheck class details a pilot's currency check. The currency check includes a name, expiry date and comments. The expiry date indicates when the currency check is due for renewal.

DFTDatabase.PilotCurrencyCheck.Name Property	
Description	Returns the name of the currency check.
Size / Limitations	20 Characters. Also defined by property DFTDB.DFTDatabase.Constants.PilotCurrencyNameLength.

DFTDatabase.PilotCurrencyCheck.ExpiryDate Property	
Description	Indicates when the currency check is due for renewal. Although this value is a string it typically contains a date value written in valid date format. i.e. 1/1/2010, 1 Jan 2010 etc.. It may contain a non-date value.
Size / Limitations	20 Characters. Also defined by property DFTDB.DFTDatabase.Constants.PilotCurrencyExpiryDateLength.

DFTDatabase.PilotCurrencyCheck.Comments Property	
Description	Returns any comments related to the currency check.
Size / Limitations	50 Characters. Also defined by property DFTDB.DFTDatabase.Constants.PilotCurrencyCommentsMaxLength.

DFTDB.PilotHours Class

The DFTDB.PilotHours class provides access to a pilot's duty, flight hours, instrument approaches, night flight, night takeoffs, night landings, instrument flight and instrument ground for a particular day. This data is displayed on the **Day Hours** tab.

The DFT8+Logbook package automatically sums the flight hours entered into the logbook and displays the total flight hours for the day on the **Day Hours** tab. The hours cannot be edited from the **Day Hours** tab - only from the **Logbook** tab. The Personal Logbook package also writes the daily flight hours however it does not display the hours on the **Day Hours** tab.

Flight hours can be entered in decimal format where the decimal part represents the fraction of 1 hour. For example, 1.5 equates to 1 hour and 30 minutes. Similarly, 1.2 equates to 1 hour and 12 minutes.

For all DFT packages, the flight, night flight, instrument ground and instrument flight time totals are accessed from the DFTDB.PilotHours class.

DFTDatabase.PilotHours.TheDate Property	
Description	The date to which the pilot's hours refers.

DFTDatabase.PilotHours.Duty(1 to DFTDB.DftDatabase.Constants.MaxDuties) Property	
Description	Property array of DFTDB.TimeOnOff objects each containing a starting and ending time in 24 hour time format. The number of DFTDB.TimeOnOff objects in the array is dependant upon the customer's settings and is typically 2. Use the DFTDB.DftDatabase.Constants.MaxDuties value to determine the maximum index into the array. Because the time does not contain a date component, ending times that are less than the starting time are assumed to be for the following day. For example, a duty period from 2200 to 0500 indicates that 2200 occurred on the date specified by the TheDate property and 0500 occurred on the following day. The IsSet property can be used to determine that the starting or ending time contains a valid value - i.e. both values are not 0000. Duty times must be entered in sequence and an entry cannot be made after a blank (0000 - 0000) duty time pair. i.e. the following cannot be entered: Duty1: 0000-0000 Duty2: 0100-0200

DFTDatabase.PilotHours.FlightTime Property	
Description	Returns the total flight time that the pilot has flown for the day. For the standard (non-logbook) version of DFT this value is entered in the Day Hours tab. For the logbook version this value is summed from the logbook entries for the day.
Size / Limitations	Single/float data type with precision to one decimal place. Valid range: 0 to 24.

DFTDatabase.PilotHours.InstrumentFlightTime Property	
Description	Returns the total instrument flight time for the day. For the standard (non-logbook) version of DFT this value is entered in the Day Hours tab. For the logbook version this value is summed from the instrument flight logbook entries for the day. This value is displayed on the Day Hours tab as Flight/Sim in the Instrument group.
Size / Limitations	Single/float data type with precision to one decimal place. Valid range: 0 to 24.

DFTDatabase.PilotHours.InstrumentGround Property	
Description	Returns the total instrument ground / simulator time for the day. For the standard (non-logbook) version of DFT this value is entered in the Day Hours tab. For the logbook version this value is summed from the instrument ground logbook entries for the day. This value is displayed on the Day Hours tab as Ground in the Instrument group.
Size / Limitations	Single/float data type with precision to one decimal place. Valid range: 0 to 24.

DFTDatabase.PilotHours.InstrumentFlightCheck Property	
Description	Indicates that an instrument flight check was performed.

DFTDatabase.PilotHours.NightTakeOffs Property	
Description	The total number of night take offs performed for the day.
Size / Limitations	Integer data type. Valid range: 0 to 99.

DFTDatabase.PilotHours.NightLandings Property	
Description	The total number of night landings performed for the day.
Size / Limitations	Integer data type. Valid range: 0 to 99.

DFTDatabase.PilotHours.InstrumentApproach() Property	
Description	Array property that indicates whether a particular instrument approach has been performed. The array can be indexed by the DFTDB.DefaultInstrumentApproachEnum for each instrument approach. The enumeration contains the following values: 1. dftInstrumentApproachILS - ILS instrument approach 2. dftInstrumentApproachVOR - VOR instrument approach 3. dftInstrumentApproachNDB - NDB instrument approach 4. dftInstrumentApproachGPSArrival - GPS Arrival / DGA instrument approach These approaches are visible on Day Hours tab in the Instrument Checks group.

DFTDatabase.PilotHours.GPSApproach(1 to 2) Property	
Description	Array property that indicates whether a GPS approach has been performed. The DFTDB.PilotSettings.GPSUnits property indicates the number of GPS units that have been setup for the pilot and the maximum index into the property. See DFTDB.PilotSettings.GPSUnits for further details on how the GPS units can be configured.

DFTDB.PilotLogbookEntry Class

The DFTDB.PilotLogbookEntry class provides access to the pilot's logbook hours, including instructor, ICUS, Dual, Co-Pilot and Command day and night hours, flight details, aircraft, instrument flight and instrument ground hours. These hours are visible on the **Logbook** tab on the pilot screen and only relevant to the logbook versions of DFT. A pilot can enter multiple logbook entries for one day.

DFTDatabase.PilotLogbookEntry.TheDate Property	
Description	The date of the logbook entry.

DFTDatabase.PilotLogbookEntry.Index Property	
Description	The index / entry number for a particular day. The date and index property are used together to uniquely identify each logbook entry in the database.
Size / Limitations	Integer data type. Valid range: 1 to 32767.

DFTDatabase.PilotLogbookEntry.Aircraft Property	
Description	Indicates the aircraft flown for the flight. See DFTDB.AircraftSet for further details on the aircraft objects.

DFTDatabase.PilotLogbookEntry.PilotInCommand Property	
Description	Indicates the pilot in command for the flight. Typically returns “Self”.
Size / Limitations	30 Characters. Also defined by property DFTDB.DFTDatabase.Constants.PilotInCommandLength.

DFTDatabase.PilotLogbookEntry.OtherPilotOrCrew Property	
Description	The names of any other pilots or crew that took part in the flight.
Size / Limitations	40 Characters. Also defined by property DFTDB.DFTDatabase.Constants.OtherPilotOrCrewLength.

DFTDatabase.PilotLogbookEntry.Details Property	
Description	The details of the flight. Typically returns the flight route.
Size / Limitations	100 Characters. Also defined by property DFTDB.DFTDatabase.Constants.DetailsLength.

DFTDatabase.PilotLogbookEntry.Instructor(1 to DFTDB.DftDatabase.Constants.MaxInstructor) Property	
Description	Property array indicates the instructor / specialist hours completed for the flight. The array dimension is typically 3. One DFT customer has 5 instructor / specialist fields. The DFTDB.DftDatabase.Constants.MaxInstructor field will return the appropriate dimension of the array.
Size / Limitations	Single/float data type with precision to one decimal place. Valid range: 0 to 24.

DFTDatabase.PilotLogbookEntry.ICUS / Dual / Command / CoPilot Properties	
Description	Returns the flight hours for ICUS / Dual / Command or Co-Pilot. Each property returns the flight hours for the day and night component. Each property is associated with either the **Single-Engine Aircraft** section or **Multi-Engined Aircraft** section of the logbook based on the aircraft selected and the value of the DFTDB.Aircraft.Engines property.
Size / Limitations	Single/float data type with precision to one decimal place. Valid range: 0 to 24.

DFTDatabase.PilotLogbookEntry.InstrumentFlight Property	
Description	Indicates the number of instrument flight hours completed as part of the flight.
Size / Limitations	Single/float data type with precision to one decimal place. Valid range: 0 to 24.

DFTDatabase.PilotLogbookEntry.InstrumentGround Property	
Description	Indicates the number of instrument flight hours completed on the ground in a simulator.
Size / Limitations	Single/float data type with precision to one decimal place. Valid range: 0 to 24.

Aviation Software

Code Example Listing

Accessing Pilot Data

```
Sub Main()

    Dim objPilot As DftDB.Pilot

    Dim objDatabase As New DftDB.DftDatabase
    objDatabase.OpenDatabase("C:\Program Files\Dft8", "AccessKey",
        strRegisteredCompanySerialNumber:="")

    Debug.Print("")
    Debug.Print("--------------------------------")
    Debug.Print(" PILOT PERSONAL DATA & SETTINGS ")
    Debug.Print("--------------------------------")
    Debug.Print("")

    For Each objPilot In objDatabase.Pilots
        Debug.Print("Name: " & objPilot.Name)
        Debug.Print("Password: " & objPilot.Password)

        Debug.Print("Personal Details:")
        Debug.Print("Full name: " & objPilot.Details.FullName)
        Debug.Print("Seniority: " & objPilot.Details.Seniority)
        Debug.Print("Address: " & objPilot.Details.Address)
        Debug.Print("Commenced: " & objPilot.Details.CommencedEmployment)
        Debug.Print("License Number: " & objPilot.Details.LicenseNumber)
        Debug.Print("Base / Location: " & objPilot.Details.BaseOrLocation)
        Debug.Print("Parent / Guardian: " & objPilot.Details.ParentOrGuardian)
        Debug.Print("Phone: " & objPilot.Details.Phone)
        Debug.Print("Fax: " & objPilot.Details.Fax)
        Debug.Print("Comments: " & objPilot.Details.Comments)

        Dim objSettings As DftDB.PilotSettings = objPilot.Settings
        Debug.Print("Settings:")
        Debug.Print("CAO Flight Rules: " & objSettings.CAOSetup)
        Debug.Print("Flight Rules: " & objSettings.FlightRules.ToString)
        Debug.Print("Duty Calculation: " & objSettings.DutyCalculationMethod.ToString)
        Debug.Print("Number of GPS Units: " & objSettings.GPSUnits)
        Debug.Print("GPS Unit 1: " & objSettings.GPSName(1))
        Debug.Print("GPS Unit 2: " & objSettings.GPSName(2))

        Debug.Print("")
    Next
```

Accessing Aircraft and Aircraft Type Data

```
'Database contains logbook information
Dim bLogbookMode As Boolean = objDatabase.LogbookMode

'Pilots can be accessed by an ordinal/index value.
'In this case get the first pilot in the database
objPilot = objDatabase.Pilots(1)

'Cache the settings so that the database is not continually accessed because of calls to
'objPilot.Settings
Dim objPilotSettings As DftDB.PilotSettings = objPilot.Settings

If bLogbookMode Then

    Debug.Print("")
    Debug.Print("-------------------------")
    Debug.Print(" AIRCRAFT DATA (LOGBOOK) ")
    Debug.Print("-------------------------")
    Debug.Print("")

    For Each objAircraft As DftDB.Aircraft In objDatabase.Aircraft
        Debug.Print("Call Sign: " & objAircraft.Registration)
        Debug.Print("Type: " & objAircraft.TheType)
        Debug.Print("Engines: " & objAircraft.Engines.ToString)
        Debug.Print("Engine Type: " & objAircraft.EngineType.ToString)
        Debug.Print("IsDeletable: " & objAircraft.IsDeletable)
        Debug.Print("")
    Next

    Debug.Print("")
    Debug.Print("---------------------------------")
    Debug.Print(" PILOT PAST EXPERIENCE (LOGBOOK) ")
    Debug.Print("---------------------------------")
    Debug.Print("")

    Debug.Print("Pilot: " & objPilot.Name)
    Debug.Print("Brought Forward Totals from Date: " &
           objPilotSettings.PastExperienceReferenceDate.ToShortDateString)
    Debug.Print("Brought Forward Totals Method: " &
           objPilotSettings.PastExperienceHoursTotalling.ToString)

    Select Case objPilotSettings.PastExperienceHoursTotalling
        Case dbPastExperienceAircraftTypeTotalsBroughtForward
            For Each objAircraftType As DftDB.PilotAircraftType In objPilot.AircraftTypes
                Debug.Print("Starting Hours for Aircraft Type " & objAircraftType.TheType & ":")
                Debug.Print("Single Engine ICUS Day: " *
                    objAircraftType.PastExperience.SingleEngineICUS.Day)
                Debug.Print("Single Engine ICUS Night: " *
                    objAircraftType.PastExperience.SingleEngineICUS.Night)
                Debug.Print("Single Engine Dual Day: " *
                   objAircraftType.PastExperience.SingleEngineDual.Day)
```

```
                PrintPilotFlightHours(objDatabase, objAircraftType.PastExperience)
            Next
        Case dbPastExperienceTotalsBroughtForward
            PrintPilotFlightHours(objDatabase, objPilot.AircraftTypes.BroughtForward)
    End Select

Else

    Debug.Print("")
    Debug.Print("-------------------------------------------------------")
    Debug.Print(" PILOT AIRCRAFT TYPES & PAST EXPERIENCE (NON-LOGBOOK) ")
    Debug.Print("-------------------------------------------------------")
    Debug.Print("")

    Debug.Print("Pilot: " & objPilot.Name)
    Debug.Print("Total Flight Hours: " & objPilot.Settings.StartingFlightHours)
    Debug.Print("Total Flight Hours as at: " &
         objPilot.Settings.StartingFlightHoursReferenceDate)

    For Each objAircraftType As DftDB.PilotAircraftTypeDFT In objPilot.AircraftTypesDFT
        Debug.Print("Type: " & objAircraftType.TheType)
        Debug.Print("Engine Type: " & objAircraftType.EngineType.ToString)
        Debug.Print("Engine Configuration: " & objAircraftType.EngineConfiguration.ToString)
        Debug.Print("Starting Hours: " & objAircraftType.StartingHours.ToString("0.0"))
        Debug.Print("Starting Hours Reference Date: " &
             objAircraftType.StartingHoursReferenceDate.ToShortDateString)
        Debug.Print("")
    Next

End If
```

Accessing Pilot Currency Checks

```
Debug.Print("")
Debug.Print("-----------------------")
Debug.Print(" PILOT CURRENCY CHECKS ")
Debug.Print("-----------------------")
Debug.Print("")

Debug.Print("Pilot: " & objPilot.Name)

For Each objCheck As DftDB.PilotCurrencyCheck In objPilot.CurrencyChecks
    Debug.Print("Check: " & objCheck.Name)
    Debug.Print("Expiry Date: " & objCheck.ExpiryDate)
    Debug.Print("Comments: " & objCheck.Comments)
    Debug.Print("")
Next
```

Accessing Duty, Flight and Instrument Checks Data

```
Debug.Print("")
Debug.Print("--------------")
Debug.Print("   DFT DATA   ")
Debug.Print("--------------")
Debug.Print("")

Dim objPilotHours As DftDB.PilotHours
Dim dtFirstEntryDate As Date = objPilotSettings.FirstEntryDate
Dim dtLastEntryDate As Date = objPilotSettings.LastEntryDate
Dim dtCurrentDate As Date = dtFirstEntryDate

Debug.Print("Pilot: " & objPilot.Name)
Debug.Print("Date of First Entry: " & dtFirstEntryDate.ToShortDateString)
Debug.Print("Date of Last Entry: " & dtLastEntryDate.ToShortDateString)

While dtCurrentDate <= dtLastEntryDate
    objPilotHours = objPilot.Days(dtCurrentDate).Hours
    ' If this is a record with actual data entries (i.e. is not a blank entry)
    If objPilotHours.FlightTime <> 0 Or
        objPilotHours.DutyDay <> 0 Or
        objPilotHours.InstrumentFlightTime <> 0 Then

        Debug.Print("Date: " & dtCurrentDate)

        'Typically objDatabase.Constants.MaxDuties returns 2 - but some customers have 3 or 4
        'sets of duty on/off times
        For intDutyIndex As Integer = 1 To objDatabase.Constants.MaxDuties
            'Indicates the Duty has the time on and time off set i.e. it is not 0000 and 0000.
            If objPilotHours.Duty(intDutyIndex).IsSet Then
                Debug.Print("Duty " & intDutyIndex & " On:  " &
                objPilotHours.Duty(intDutyIndex).TimeOn)      'format is 24 hour format i.e. 1230
                Debug.Print("Duty " & intDutyIndex & " Off: " &
                objPilotHours.Duty(intDutyIndex).TimeOff)     'format is 24 hour format i.e. 1230
            Else
                'There are no further duty times entered
                Exit For
            End If
        Next

        Debug.Print("Flight Time: " & objPilotHours.FlightTime)
        If Not bLogbookMode Then
            For intAircraftTypeFlightHours As Integer = 1 To
                                                    objDatabase.Constants.MaxDFTAircraft
                Debug.Print("Flight Time (" &
                    objPilot.AircraftTypesDFT(intAircraftTypeFlightHours).TheType & "): " &
                    objPilotHours.AircraftFlightHours(intAircraftTypeFlightHours))
            Next
        End If

        Debug.Print("Instrument Flight Time: " & objPilotHours.InstrumentFlightTime)
```

```
        Debug.Print("Instrument Ground: " & objPilotHours.InstrumentGround)
        Debug.Print("Instrument Flight Check: " & objPilotHours.InstrumentFlightCheck)
        Debug.Print("Night Flight: " & objPilotHours.NightFlightTime)
        Debug.Print("Night Takeoffs: " & objPilotHours.NightTakeOffs)
        Debug.Print("Night Landings: " & objPilotHours.NightLandings)

        Debug.Print("ILS: " & objPilotHours.InstrumentApproach(dftInstrumentApproachILS))
        Debug.Print("VOR: " & objPilotHours.InstrumentApproach(dftInstrumentApproachVOR))
        Debug.Print("NDB: " & objPilotHours.InstrumentApproach(dftInstrumentApproachNDB))
        Debug.Print("DGA: " & objPilotHours.InstrumentApproach(dftInstrumentApproachGPSArrival))

        If objPilotSettings.GPSUnits >= 1 Then
            Debug.Print(objPilotSettings.GPSName(1) & ": " & objPilotHours.GPSApproach(1))
        End If
        If objPilotSettings.GPSUnits >= 2 Then
            Debug.Print(objPilotSettings.GPSName(2) & ": " & objPilotHours.GPSApproach(2))
        End If

        Debug.Print("")

    End If
    dtCurrentDate = dtCurrentDate.AddDays(1)
End While
```

Accessing Logbook Data

```
    Debug.Print("")
    Debug.Print("----------------------")
    Debug.Print("     LOGBOOK DATA     ")
    Debug.Print("----------------------")
    Debug.Print("")

    Debug.Print("Pilot: " & objPilot.Name)

    dtCurrentDate = dtFirstEntryDate
    While dtCurrentDate <= dtLastEntryDate

        'If there are logbook entries for the day
        For Each objEntry As DftDB.PilotLogbookEntry In objPilot.Logbook(dtCurrentDate)
            Debug.Print("Date: " & objEntry.TheDate.ToShortDateString)
            Debug.Print("Entry Number: " & objEntry.Index)
            Debug.Print("Aircraft: " & objEntry.Aircraft.Registration)
            Debug.Print("Pilot In Command: " & objEntry.PilotInCommand)
            Debug.Print("Other Pilot or Crew: " & objEntry.OtherPilotOrCrew)
            Debug.Print("Details: " & objEntry.Details)
            For intInstructorIndex As Integer = 1 To objDatabase.Constants.MaxInstructor
                Debug.Print("Instructor " & intInstructorIndex & ": " &
                    objEntry.Instructor(intInstructorIndex))
            Next
            Debug.Print("ICUS Day: " & objEntry.ICUS.Day)
            Debug.Print("ICUS Night: " & objEntry.ICUS.Night)
            Debug.Print("Dual Day: " & objEntry.Dual.Day)
            Debug.Print("Dual Night: " & objEntry.Dual.Night)
            Debug.Print("Command Day: " & objEntry.Command.Day)
            Debug.Print("Command Night: " & objEntry.Command.Night)
            Debug.Print("CoPilot Day: " & objEntry.CoPilot.Day)
            Debug.Print("CoPilot Night: " & objEntry.CoPilot.Night)
            Debug.Print("Instrument Flight: " & objEntry.InstrumentFlight)
            Debug.Print("Instrument Ground: " & objEntry.InstrumentGround)

            Debug.Print("")
        Next

        dtCurrentDate = dtCurrentDate.AddDays(1)
    End While

    'Close the database - ensuring that all files have been closed.
    objDatabase.CloseDatabase()

End Sub
```

```
Private Sub PrintPilotFlightHours( _
    ByVal objDatabase As DftDB.DftDatabase, _
    ByVal objFlightHours As DftDB.PilotFlightHours)

    For intInstructorIndex As Integer = 1 To objDatabase.Constants.MaxInstructor
        Debug.Print("Instructor " & intInstructorIndex & ": " &
            objFlightHours.Instructor(intInstructorIndex))
    Next

    Debug.Print("SingleEngine ICUS Day: " & objFlightHours.SingleEngineICUS.Day)
    Debug.Print("SingleEngine ICUS Night: " & objFlightHours.SingleEngineICUS.Night)
    Debug.Print("SingleEngine Dual Day: " & objFlightHours.SingleEngineDual.Day)
    Debug.Print("SingleEngine Dual Night: " & objFlightHours.SingleEngineDual.Night)
    Debug.Print("SingleEngine Command Day: " & objFlightHours.SingleEngineCommand.Day)
    Debug.Print("SingleEngine Command Night: " & objFlightHours.SingleEngineCommand.Night)
    Debug.Print("SingleEngine CoPilot Day: " & objFlightHours.SingleEngineCoPilot.Day)
    Debug.Print("SingleEngine CoPilot Night: " & objFlightHours.SingleEngineCoPilot.Night)

    Debug.Print("MultiEngine ICUS Day: " & objFlightHours.MultiEngineICUS.Day)
    Debug.Print("MultiEngine ICUS Night: " & objFlightHours.MultiEngineICUS.Night)
    Debug.Print("MultiEngine Dual Day: " & objFlightHours.MultiEngineDual.Day)
    Debug.Print("MultiEngine Dual Night: " & objFlightHours.MultiEngineDual.Night)
    Debug.Print("MultiEngine Command Day: " & objFlightHours.MultiEngineCommand.Day)
    Debug.Print("MultiEngine Command Night: " & objFlightHours.MultiEngineCommand.Night)
    Debug.Print("MultiEngine CoPilot Day: " & objFlightHours.MultiEngineCoPilot.Day)
    Debug.Print("MultiEngine CoPilot Night: " & objFlightHours.MultiEngineCoPilot.Night)

    Debug.Print("Instrument Flight: " & objFlightHours.InstrumentFlight)
    Debug.Print("Instrument Ground: " & objFlightHours.InstrumentGround)

End Sub
```