

# FlightStream Analysis Interface Module (AIM) Manual

John Dannenhoffer (Syracuse) and Marshall Galbraith MIT

January 15, 2026



---

0.1 Introduction	1
0.1.1 FlightStream AIM Overview	1
0.1.2 Geometry Requirements and Assumptions	1
0.1.2.1 Sectional Geometry Requirements	1
0.2 Attribution	3
0.3 FlightStream Fluid Properties	3
0.4 FlightStream Data Transfer	4
0.4.1 Data transfer to FlightStream (FieldIn)	4
0.4.2 Data transfer from FlightStream (FieldOut)	4
0.5 AIM Units	4
0.5.1 JSON String Dictionary	5
0.6 AIM Inputs	5
0.7 AIM Outputs	8
0.8 AIM Execution	9
0.9 Vortex Lattice Surface	9
0.9.1 Single Value String	10
0.10 Vortex Lattice Control Surface	11
0.10.1 Single Value String	11
Bibliography	13
Index	15



## 0.1 Introduction

### 0.1.1 FlightStream AIM Overview

A module in the Computational Aircraft Prototype Syntheses (CAPS) has been developed to interact (primarily through input files) with Research in Flight's FlightStream [1]. FlightStream predicts the aerodynamic performance of a vehicle via a panel method, which makes it very fast.

An outline of the AIM's inputs and outputs are provided in [AIM Inputs](#) and [AIM Outputs](#), respectively.

Geometric attributes recognized by the AIM are provided in [Attribution](#).

### 0.1.2 Geometry Requirements and Assumptions

The FlightStream coordinate system assumption (X – downstream) needs to be followed.

The FlightStream AIM supports two types of geometry representations:

1. Solid body (SOLIDBODY) or manifold sheet body (SHEETBODY)  
This geometry format requires the use of a surface meshing AIM that is linked to the Surface\_Mesh input (see [AIM Inputs](#)).
2. Sectional geometry  
Sectional geometry information used to generate FlightStream CCS geometry file (see CCS inputs in [AIM Inputs](#)). Currently supported CCS bodies are:
  - General Components (Fuselage, Fairings, Pods, etc.)
  - Lifting Surface Components (Wing, Fin, etc.) with Standard Deflection Control Surfaces
  - Annular Components
  - Revolution Components

#### 0.1.2.1 Sectional Geometry Requirements

Within **OpenCSM** there are a number of airfoil generation UDPs (User Defined Primitives). These include NACA 4 series, a more general NACA 4/5/6 series generator, Sobieczky's PARSEC parameterization and Kulfan's CST parameterization. All of these UDPs generate **EGADS FaceBodies** where the *Face*'s underlying *Surface* is planar and the bounds of the *Face* is a closed set of *Edges* whose underlying *Curves* contain the airfoil shape.

#### Important Airfoil Geometry Assumptions

- There must be a *Node* that represents the *Leading Edge* point
- For a sharp trailing edge, there must be a *Nodes* at the *Trailing Edge*
- For a blunt trailing edge, the airfoil curve may be open, or closed by a single *Edge* connecting the upper/lower *Nodes*
- For a *FaceBody*, the airfoil coordinates traverse counter-clockwise around the *Face* normal. The **OpenCSM REORDER** operation may be used to flip the *Face* normal.
- For a *WireBody*, the airfoil coordinates traverse in the order of the loop

**Note:** Additional spurious *Nodes* on the upper and lower *Edges* of the airfoil are acceptable.

The FlightStream CCS surfaces are generated from a set of *FaceBodys* with the same capsGroup attributes, and the geometric details extracted from the geometry. Attempts are made to orient and sort *FaceBody*, but users are encouraged to check the airfoil orientation in the CCS input file when setting up a new problem.

It should be noted that general construction in either **OpenCSM** or even **EGADS** will be supported as long as the topology described above is used. But care should be taken when constructing the airfoil shape so that a discontinuity (i.e., simply  $C^0$ ) is not generated at the *Node* representing the *Leading Edge*. This can be done by fitting a spline through the entire shape as one and then intersecting the single *Edge* to place the LE *Node*.

### CCS General Components

The *FaceBody* sections for general components and can be oriented in any direction with any shape. However, sections may not have perpendicular normal vectors.

### CCS Lifting Surface Components

The *FaceBody* must contain at least two edges and two nodes, but may contain any number of *Edges* otherwise. If the *FaceBody* contains more nodes, the node with the smallest *x* value is used to define the leading edge, the node with the largest *x* defines the trailing edge. The airfoil may have a single *Edge* that defines a straight blunt trailing edge. The airfoil shapes in the CCS file are generated by sampling the *Curves* of each section.

Trailing edge control surfaces can be added to the above example by making use of the vlmControlName attribute (see [Attribution](#) regarding the attribution specifics). To add a **RightFlap** and **LeftFlap** to the the naca UDP entries are augmented with the following attributes.

```
# left tip
udprim      naca      Thickness thick      Camber      camber
attribute vlmControl_LeftFlap 80 # Hinge line is at 80% of the chord (control surface between hinge and
trailing edge)
...

# root
udprim      naca      Thickness thick      Camber      camber
attribute vlmControl_LeftFlap 80 # Hinge line is at 80% of the chord (control surface between hinge and
trailing edge)
attribute vlmControl_RightFlap 80 # Hinge line is at 80% of the chord (control surface between hinge and
trailing edge)
...

# right tip
udprim      naca      Thickness thick      Camber      camber
attribute vlmControl_RightFlap 80 # Hinge line is at 80% of the chord (control surface between hinge and
trailing edge)
...
```

Note how the root airfoil contains two attributes for both the left and right flaps. Only trailing edge control surfaces are currently supported by FlightStream.

In the pyCAPS script the [AIM Inputs](#), **CCS\_Control**, must be defined.

```
flap = {"deflectionAngle" : 10.0}

flightstream.input.CCS_Control = {"LeftFlap": flap, "RightFlap": flap}
```

Notice how the information defined in the **flap** variable is assigned to the vlmControlName portion of the attributes added to the \*.csm file.

### CCS Annular Components

Annular components are made up of multiple annular airfoil sections. The same section rules for Lifting Surface components apply. However, blunt trailing edges are currently not supported by FlightStream.

### CCS Revolution Components

Revolution components are made up of a single airfoil sections along with an axis of rotation. The same section rules for Lifting Surface components apply. However, blunt trailing edges are currently not supported by FlightStream. The axis of rotation body must be a LINE and have the same capsGroup attribute as the other sections, as well as a capsType=Axis string attribute.

## 0.2 Attribution

The following list of attributes drives the FlightStream geometric definition.

- **capsGroup** For sectional CCS geometry, this string attribute labels the *FaceBody* as to which AVL Surface the section is assigned. This should be something like: *Main\_Wing*, *Horizontal\_Tail*, etc. This informs the AVL AIM to collect all *FaceBodies* that match this attribute into a single AVL Surface.  
Note: If a capsGroup contains only one section then the section is treated as a slender body, and only the numChord and spaceChord in the "AVL\_Surface" ([Vortex Lattice Surface](#)) input will be used.
- **vlmControl"Name"** This string attribute attaches a control surface to the Lifting Surface CCS geometry *FaceBody*. The hinge location is defined as the double value between 0 or 1.0. The range as percentage from 0 to 100 will also work. The name of the control surface is the string information after vlmControl (or vlmControl\_). For Example, to define a control surface named Aileron the following are identical (*attribute vlmControlAileron 0.8* or *attribute vlmControl\_Aileron 80*). Multiple *vlmControl* attributes, with different names, can be defined on a single *FaceBody*.
- **vlmNumSpan** This attribute may be set on any given airfoil cross-section to overwrite the number of spanwise elements placed on the surface (globally set - see keyword "numSpanPerSection" and "numSpanTotal" in [Vortex Lattice Surface](#)) between two sections. Note, that the AIM internally sorts the sections in ascending y (or z) order, so care should be taken to select the correct section for the desired intent.
- **vlmNumChord** This attribute may be set on any given airfoil cross-section to overwrite the number of chord-wise elements placed on the surface (globally set - see keyword "numChord" in [Vortex Lattice Surface](#)). vlmNumChord must be the same if set on multiple sections of a surface.
- **capsLength** This attribute defines the length units that the \*.csm file is generated in and is not optional for FlightStream. The FlightStream input grid will be scaled to either the default length of METER or the user specified length unit (see [AIM Units](#)).
- **capsReferenceChord** and **capsReferenceSpan** [Optional] These attributes may exist on any *Body*. Their value will be used as the reference moment lengths in FlightStream's input file with their units assumed to be consistent with the attribute "capsLength". These values may be alternatively set through an input value, "ReferenceChord" (see [AIM Inputs](#))
- **capsReferenceArea** [Optional] This attribute may exist on any *Body*. Its value will be used as the reference area in FlightStream's input file with its units assumed to be consistent with the attribute "capsLength". This value may be alternatively set through an input value, "ReferenceArea" (see [AIM Inputs](#))
- **capsReferenceX**, **capsReferenceY**, and **capsReferenceZ** [Optional] These attribute may exist on any *Body*. Their value will be used as the reference moment lengths in FlightStream's input file with their units assumed to be consistent with the attribute "capsLength". These values may be alternatively set through an input value, "ReferenceX" (see [AIM Inputs](#))
- **capsReferenceX**, **capsReferenceY**, and **capsReferenceZ** [Optional] These attribute may exist on any *Body*. Their value will be used as the reference moment lengths in FlightStream's input file with their units assumed to be consistent with the attribute "capsLength". These values may be alternatively set through an input value, "ReferenceX" (see [AIM Inputs](#))

## 0.3 FlightStream Fluid Properties

Structure for the load tuple = ("Property", "Value").

- **density = "(no default)"**  
Reference density

- **pressure = "(no default)"**  
Reference pressure
- **sonic\_velocity = "(no default)"**  
Reference speed of sound
- **temperature = "(no default)"**  
Reference temperature
- **viscosity = "(no default)"**  
Reference viscosity

## 0.4 FlightStream Data Transfer

### 0.4.1 Data transfer to FlightStream (FieldIn)

- **"Displacement"**  
Retrieves nodal displacements (as from a structural solver) and updates FlightStream surface mesh.

The FlightStream AIM has the ability to transfer surface data (e.g. pressure distributions) to and from the AIM using the conservative and interpolative data transfer schemes in CAPS.

### 0.4.2 Data transfer from FlightStream (FieldOut)

- **"RelPressure"**  
Loads the relative to freestream pressure distribution from FlightStream vtk file. This distribution may be scaled based on  $\text{Pressure} = \text{Pressure\_Scale\_Factor} * \text{RelPressure}$ , where "Pressure\_Scale\_Factor" is an AIM input ([AIM Inputs](#))
- **"AbsPressure"**  
Loads the absolute pressure distribution from FlightStream vtk file. This distribution may be scaled based on  $\text{Pressure} = \text{Pressure\_Scale\_Factor} * \text{AbsPressure}$ , where "Pressure\_Scale\_Factor" is an AIM input ([AIM Inputs](#))

## 0.5 AIM Units

FlightStream expects units for all inputs, and by default the AIM uses SI units, i.e.

- mass : kg
- length : meter
- time : seconds
- temperature : K A unit system may be optionally specified during AIM instance initiation to use a different set of base units. A unit system may be specified via a JSON string dictionary for example: `unitSys = {"mass": "lb", "length": "feet", "time": "seconds", "temperature": "R"}`



### 0.5.1 JSON String Dictionary

The key arguments of the dictionary are described in the following:

- **mass = "None"**  
Mass units - e.g. "kilogram", "k", "slug", ...
- **length = "None"**  
Length units - FlightStream support: "inch", "millimeter", "feet", "mile", "meter", "kilometer", "mils", "micron", "centimeter", "microinch"
- **time = "None"**  
Time units - e.g. "second", "s", "minute", ...
- **temperature = "None"**  
Temperature units - e.g. "Kelvin", "K", "degC", ...

## 0.6 AIM Inputs

The following list outlines the FlightStream inputs along with their default values available through the AIM interface.

- **ProjName = "flightstream\_CAPS"**  
Name for files generated by flightstream AIM.
- **FlightStream = "FlightStream"**  
The name of the FlightStream executable. May include full path.
- **Mach = NULL**  
Mach number
- **Alpha = NULL**  
Angle of attack
- **Beta = NULL**  
Sideslip angle
- **SweepRangeMach = NULL**  
Use FlightStream Sweeper Toolbox to solve for a range of Mach numbers.  
SweepRangeMach is a vector of length 3 with [start, end, delta] Mach numbers.  
Only one of 'Mach', 'SweepRangeMach', or 'SweepRangeVelocity' must be specified.
- **SweepRangeVelocity = NULL**  
Use FlightStream Sweeper Toolbox to solve for a range of velocities.  
SweepRangeVelocity is a vector of length 3 with [start, end, delta] velocities.  
Only one of 'Mach', 'SweepRangeMach', or 'SweepRangeVelocity' must be specified.
- **SweepRangeAlpha = NULL**  
Use FlightStream Sweeper Toolbox to solve for a range of angles of attack.  
SweepRangeAlpha is a vector of length 3 with [start, end, delta] angles.  
Only one of 'Alpha' or 'SweepRangeAlpha' must be specified.
- **SweepRangeBeta = NULL**  
Use FlightStream Sweeper Toolbox to solve for a range of sideslip angles.  
SweepRangeBeta is a vector of length 3 with [start, end, delta] angles.  
Only one of 'Beta' or 'SweepRangeBeta' must be specified.

- **SweepClear\_Solution\_After\_Each\_Run = False**  
Re-initialize the solution between each analysis when using FlighStream Sweeper Toolbox.
- **SweepReference\_Velocity\_Equals\_Freestream = False**  
Use the updated reference velocity when using FlighStream Sweeper Toolbox to sweep over Mach or Velocity.
- **SweepExport\_Surface\_Data = NULL**  
Export surface data while using FlighStream Sweeper Toolbox. Only one file format may be specified. Available options:
  - DAT
  - VTK
  - CSV
  - TXT
- **Fluid\_Properties = NULL**  
Reference fluid properties. Altitude must be NULL. See [FlightStream Fluid Properties](#).
- **Altitude = NULL (default)**  
Altitude used to compute Fluid Properties. The Fluid\_Properties input must be NULL.
- **Solver\_Model = NULL (default)**  
One of: "INCOMPRESSIBLE", "SUBSONIC\_PRANDTL\_GLAUERT", "TRANSONIC\_FIELD\_PANEL", "↔ SUPersonic\_LINEAR\_DOUBLET", "TANGENT\_CONE", "MODIFIED\_NEWTONIAN"
- **Wake\_Termination = NULL (default)**  
Wake termination x-location
- **Stabilization = 1.0 (default)**  
Stabilization value for Solver\_Model "INCOMPRESSIBLE", "SUBSONIC\_PRANDTL\_GLAUERT", or "↔ TRANSONIC\_FIELD\_PANEL"
- **Wall\_Collision\_Avoidance = false (default)**  
Enable wake Wall Collision Avoidance
- **CCS\_GeneralSurface = NULL (default)**  
General surface information for CCS geometry. See [Vortex Lattice Surface](#) for additional details.
- **CCS\_LiftingSurface = NULL (default)**  
Lifting surface information for CCS geometry. See [Vortex Lattice Surface](#) for additional details.
- **CCS\_Control = NULL (default)**  
Control surface information for CCS geometry. See [Vortex Lattice Control Surface](#) for additional details.
- **CCS\_Revolution = NULL (default)**  
Body of Revolution information for CCS geometry. Only one airfoil section and one body for the axis of revolution must be provided. See [Vortex Lattice Surface](#) for additional details.
- **CCS\_Annular = NULL (default)**  
Nacelle and Annular body information for CCS geometry. See [Vortex Lattice Surface](#) for additional details.
- **ReferenceChord = NULL**  
This sets the reference chord for used in force and moment calculations. Alternatively, the geometry (body) attribute (see [Attribution](#)) "capsReferenceChord" maybe used to specify this variable (note: values set through the AIM input will supersede the attribution value).
- **ReferenceSpan = NULL**  
This sets the reference span for used in force and moment calculations. Alternatively, the geometry (body) attribute (see [Attribution](#)) "capsReferenceSpan" maybe used to specify this variable (note: values set through the AIM input will supersede the attribution value).

- **ReferenceArea = NULL**  
This sets the reference area for used in force and moment calculations. Alternatively, the geometry (body) attribute (see [Attribution](#)) "capsReferenceArea" maybe used to specify this variable (note: values set through the AIM input will supersede the attribution value).
- **ReferenceX = NULL**  
This sets the reference X for moment calculations. Alternatively, the geometry (body) attribute (see [Attribution](#)) "capsReferenceX" maybe used to specify this variable (note: values set through the AIM input will supersede the attribution value).
- **ReferenceY = NULL**  
This sets the reference Y for moment calculations. Alternatively, the geometry (body) attribute (see [Attribution](#)) "capsReferenceY" maybe used to specify this variable (note: values set through the AIM input will supersede the attribution value).
- **ReferenceZ = NULL**  
This sets the reference Z for moment calculations. Alternatively, the geometry (body) attribute (see [Attribution](#)) "capsReferenceZ" maybe used to specify this variable (note: values set through the AIM input will supersede the attribution value).
- **ReferenceVelocity = NULL**  
This sets the reference velocity
- **Pressure\_Scale\_Factor = 1.0**  
Value to scale Pressure data when transferring data. Data is scaled based on  $\text{Pressure} = \text{Pressure\_Scale\_Factor} * \text{Pressure}$ .
- **ConvergenceTol = 1.0e-5 (default)**  
Solver convergence tolerance
- **SolverIterations = NULL**  
Maximum number of solver iterations
- **SolverConvergence = NULL**  
Solver convergence tolerance
- **Export\_Solver\_Analysis = NULL**  
List of file types to export. Available options:
  - Tecplot
  - VTK
  - CSV
  - BDF
  - Force\_Distributions
- **FlightScriptPre = NULL**  
List of flight script commands injected in script.txt prior to INITIALIZE\_SOLVER
- **FlightScriptPost = NULL**  
List of flight script commands to append at the end of script.txt
- **Hidden = True**  
Hide FlightStream GUI during execution on Windows
- **Max\_Parallel\_Threads = NULL**  
Maximum number of threads for FlightStream analysis
- **SaveFSM = False**  
Save a FlightStream .fsm file after analysis
- **Mesh\_Morph = False**  
Project previous surface mesh onto new geometry.
- **Surface\_Mesh = NULL**  
A Surface\_Mesh link.

## 0.7 AIM Outputs

The following list outlines the FlightStream outputs available through the AIM interface.

Analysis settings:

- **Alpha** = Angle of attack
- **Beta** = Sideslip angle
- **Mach** = Freestream mach
- **Velocity** = Freestream velocity
- **Density** = Freestream density
- **Pressure** = Freestream pressure
- **SonicSpeed** = Freestream speed of sound
- **Temperature** = Freestream temperature
- **Viscosity** = Freestream viscosity

Aerodynamic coefficients:

- **Cx** = X-force coefficient
- **Cx** = X-force coefficient
- **Cx** = X-force coefficient
- **CL** = Lift coefficient
- **CDi** = Induced drag coefficient
- **CDo** = Skin friction drag coefficient
- **CMx** = X-moment coefficient
- **CMy** = Y-moment coefficient
- **CMz** = Z-moment coefficient

NOTE: Coefficients are normalized by the input reference velocity, not the freestream velocity.

Component aerodynamic coefficients are also available as dynamic outputs for a single point analysis.

## 0.8 AIM Execution

If auto execution is enabled when creating an FlightStream AIM, the AIM will execute FlightStream just-in-time on Linux with the command line:

```
FlightStream script.txt > flightstreamOut.txt
```

and on Windows with the command:

```
FlightStream -hidden -script script.txt > flightstreamOut.txt
```

In both cases the FlightStream executable is assumed to be in the PATH environment variable.

The analysis can also be explicitly executed with `caps_execute` in the C-API or via `Analysis.runAnalysis` in the pyCAPS API.

Calling `preAnalysis` and `postAnalysis` is NOT allowed when auto execution is enabled.

Auto execution can also be disabled when creating an FlightStream AIM object. In this mode, `caps_execute` and `Analysis.runAnalysis` can be used to run the analysis, or FlightStream can be executed by calling `preAnalysis`, `system`, and `posAnalysis` as demonstrated below with a pyCAPS example:

```
print ("\n\npreAnalysis.....")
flightstream.preAnalysis()

print ("\n\nRunning.....")
flightstream.system("FlightStream.exe -hidden -script script.txt"); # Run via system call in inputs analysis
                           directory

print ("\n\npostAnalysis.....")
flightstream.postAnalysis()
```

## 0.9 Vortex Lattice Surface

Structure for the Vortex Lattice Surface tuple = ("Name of Surface", "Value"). "Name of surface" defines the name of the surface in which the data should be applied. The "Value" can either be a JSON String dictionary (see Section \ref jsonStringVLMSurface) or a single string keyword string (see Section \ref keyStringVLMSurface). @section jsonStringVLMSurface JSON String Dictionary If "Value" is a JSON string dictionary (eg. "Value" = {"numChord": 5, "spaceChord": 1.0, "numSpan": 10, "spaceSpan": 0.5}) the following keywords ( = default values) may be used:

- **groupName = "(no default)"**  
Single or list of *capsGroup* names used to define the surface (e.g. "Name1" or ["Name1","Name2",...]). If no groupName variable is provided an attempted will be made to use the tuple name instead;
- **numChord = 10**  
The number of chordwise horseshoe vortices placed on the surface. Note: The chordwise count may be overridden using the *vlmNumChord* BODY attribute on a section.
- **numSpanTotal = 0**  
Total number of spanwise horseshoe vortices placed on the surface. The vortices are 'evenly' distributed across sections to minimize jumps in spacings. *numSpanPerSection* must be zero if this is set.  
Note: The local spanwise count may be overridden using the *vlmNumSpan* BODY attribute on a section.
- **numSpanPerSection = 0**  
The number of spanwise horseshoe vortices placed on each section the surface. The total number of spanwise vortices are (numSection-1)\*numSpanPerSection. The vortices are 'evenly' distributed across sections to minimize jumps in spacings. *numSpanTotal* must be zero if this is set.  
Note: The local spanwise count may be overridden using the *vlmNumSpan* BODY attribute on a section.

- **sortVec = [0.0, 0.0, 0.0]**  
Vector for sorting airfoil sections.  
By default, section normals are used for sorting.  
However, for LINE sections, a sorting direction is necessary if sorting is desired.
  
- **growthTypeChord = NULL (int)**  
Chordwise growth type:
  - 1 = Uniform spacing
  - 2 = Successive
  - 3 = Dual-sided successive
  - 4 = Reverse Successive
  
- **growthRateChord = NULL (double)**  
Chordwise growth rate
  
- **periodicityChord = NULL (int)**  
Chordwise periodicity of the refinement pattern
  
- **growthTypeSpan = NULL (int)**  
Spanwise growth type:
  - 1 = Uniform spacing
  - 2 = Successive
  - 3 = Dual-sided successive
  - 4 = Reverse Successive
  
- **growthRateSpan = NULL (double)**  
Spanwise growth rate
  
- **periodicitySpan = NULL (int)**  
Spanwise periodicity of the refinement pattern
  
- **blendTrailingEdge = false**  
Blend blunt trailing edges into sharp trailing edges (does not increase chord)
  
- **startAngle = 0**  
Starting angle for revolved surface
  
- **startAngle = 360 (default)**  
End angle for revolved surface

### 0.9.1 Single Value String

If "Value" is a single string the following options maybe used:

- (NONE Currently)

## 0.10 Vortex Lattice Control Surface

Structure for the Vortex Lattice Control Surface tuple = ("Name of Control Surface", "Value"). "Name of control surface" defines the name of the control surface in which the data should be applied. The "Value" must be a JSON String dictionary (see Section \ref jsonStringVLMSection). @section jsonStringVLMSection JSON String Dictionary If "Value" is a JSON string dictionary (e.g. "Value" = {"deflectionAngle": 10.0}) the following keywords ( = default values) may be used:

- **deflectionAngle = 0.0**  
Deflection angle of the control surface.
- **hingeHeight = 0.5**  
Control hinge fraction of height. Valid range [0, 1].
- **slotGap = 0.001**  
Control surface inner/outer slot gap as fraction of span. Valid range [0, 1].

### 0.10.1 Single Value String

If "Value" is a single string, the following options maybe used:

- (NONE Currently)





# Bibliography

[1] Altair flightstream. [1](#)



# Index

AIM Execution, [9](#)

AIM Inputs, [5](#)

AIM Outputs, [8](#)

AIM Units, [4](#)

Attribution, [3](#)

FlightStream Data Transfer, [4](#)

FlightStream Fluid Properties, [3](#)

Introduction, [1](#)

Vortex Lattice Control Surface, [11](#)

Vortex Lattice Surface, [9](#)