**WORKING DRAFT**

# MULTI−PHASE/LIQUID SPRAY EXTENSION TO CGNS

**Revised: 12/1/2000**

Some commentary...

At this point, the primary interest is in defining the technical information to be stored. I tried incorporating it in a loose SIDS−ish representation; this needs work, but hopefully it gets the message across. Its meant to be generalized for multiple needs, i.e., should work for things like Lagrangian or Eulerian approach; spherical/non−spherical particle geometries; solid particles/liquid droplets &/or gas bubbles; steady or time−accurate CFD calculations.

For the multi−phase analysis, I assume the user may have packet &/or field data to store, which is put under each zone.

A "Packet" refers to a group of particles (solid particles, liquid droplets, or gas bubbles) which are typically used in a Lagrangian approach, and each packet exists at a specific x, y, z location in the domain at a specific time. There may be multiple packets in a zone, or none at all. The items in the data list for the packet (e.g., VelocityX, etc.) are single−valued.

Field data array (i.e., at each i, j, k in each zone) also may be required in a Lagrangian calculation, typically to record droplet source terms for each variable which are solved for (e.g., u, v, p, t, ..). This information is may be stored off if the CFD solution is marching to a steady solution. This should the follow the SIDS standard (DataArray_t, I believe).

Also, additional field data information (at each i, j, k in each zone) is required for the Eulerian approach. A velocity, volume fraction, temperature, etc., is associated for each particle in each cell, which needs to be recorded.

For time−accurate calculations, these data arrays & packets may need to be written off both at time *(t)* and *(t−1)*. This should be checked with the time−accurate portion of the SIDS for compatibility.

Regarding **ParticleType_t:** Most people only deal with spherical particles, hence the variable, ParticleRadius, is included. But, one may have non−spherical particles, so nParticleLengthScale & ParticleLengthScale are available for a user−defined instance of the particle geometry. If nParticleLengthScale =1, then the ParticleLengthScale can be defined as the radius. (So, yes, this seems redundant. Should we get rid of ParticleRadius ? )
(BTW, I believe particle orientation may also be useful to some folks, particularly for radiation; but I'm not accounting for that here.)

Any standardized data name identifier in the SIDS (Appendix A) may be fair game for the single−valued packet data, or the field data. Some typical ones are shown. But I think Volume, VolumeFraction may be new and required.

The data/format for the models (for evaporation, atomizations, etc.) used may get quite tricky. I don't know industry standards for these, and they are likely to be very code specific. If there are industry standards, we should list them, but definitely allow for user to enter a Descriptor/ text line or label (char32). Multiple models may be invoked, or none. Or not apply, depending on the phase. I'd like to worry about them later.

**Now in SIDS−ish terms, we define:**

ApproachType_t: = Enumeration(Eulerian, Lagrangian)

PhaseType_t:= Enumeration(Gas, Liquid, Solid)

CompostionType_t:= Enumeration(H2, O2, H2O, ... possibly any from chemistry/SIDS list of species)

```
ParticleType_t:={
        int ParticleID;                                                (r)
        char*32 ParticleName;      (Descriptor?)                       (r)
        real ParticleRadius;                                           (o)
        int nParticleLengthScale;                                      (o)
        DataArray_t<R4, 1, nParticleLengthScale > ParticleLengthScale; (o)
        CompositionType_t CompositionType;                             (r)
        PhaseType_t PhaseType;                                         (r)
}


PacketType_t:={
        int PacketID;                                                  (r)
        char*32 PacketName;        (Descriptor?)                       (r)
        ParticleType_t ParticleType                                    (r)
        real ParticlesPerPacket;                                       (r)
        List(DataArray_t<>         VelocityX, VelocityY, VelocityZ,
                                   CoordinateX, CoordinateY, CoordinateZ,
                                   MomentumX, MomentumY, MomentumZ
                                   Volume, Temperature, Density,
                                   etc... );                           (r)
}

Eulerian_t:{
        ParticleType_t ParticleType;
        Data_Array_t:              VolumeFraction,
                                   VelocityX, VelocityY, VelocityZ,
                                   MomentumX, MomentumY, MomentumZ,
                                   etc....);
}

Lagrangian_t:{
        ParticleType_t ParticleType;
        PacketType_t PacketType;
        Data_Array_t:              VelocityX, VelocityY, VelocityZ,
                                   MomentumX, MomentumY, MomentumZ,
                                   H2, O2,
                                   etc....);
}


ModelType_t: {                                       (o)
        Atomization_t:                               (o)
        Breakup_t_t:                                 (o)
        Cavitation_t:                                (o)
        Coalescence_t:                               (o)
        Collision_t:                                 (o)
        Evaporation_t:                               (o)
        }
```

```
MultiPhase_t:{
        ApproachType_t ApproachType;                (r)
        ModelType_t: ModelType;                     (o)
        Data(ApproachType)                          (r)
}
```