# Data input for WallDYN runs

Generally, WallDYN needs to be able to determine the incident particle energy and impact angle for each species to derive sputter and reflection yields. The data requested below assumes sheath acceleration for ions and constant energy for charge exchange species. Also, WallDYN distinguishes between background species and dynamic species. Background species usually are the majority plasma species e.g. D, H, He whereas the dynamic species are the ones for which migration and surface dynamics are computed e.g. Be/W.

Although not explicitly need (the wall element areas would be enough) it would be good to have the 3D geometry (triangles or quads depending on you meshing) usually via a vertex buffer and an index buffer where every three/four indices pointing into the vertex buffer make up a triangle/quad.

Internally WallDYN (like trim/tridyn) uses A (10-10m) for length, eV for energy

For each dynamic species (e.g. Be, W)

1. Pure element number density (#/A3)
2. Sputter/reflection/sublimation yield database

**I can take care of the above**

For each wall element (can be any number of „binned“ triangles/quads that make up the actual wall geometry)

1. Te, Ti
2. Constant flux of background ion species
3. Constant flux of background CX-species
4. Constant energy of background CX species
5. Impact angle of species (usually different for gyrating vs neutral particles)
6. Wall temperature (if sublimation and/or chemical erosion is requested)
7. Wall element area (usually computed by WallDYN based on 3D geometry data)
8. Initial composition (atomic fractions) of the species in the wall tile

**The above would have to come from the Oakridge side**

The redistribution matrix should best be raw particle numbers, any normalization can be done internally by WallDYN. So for each species and given number of neutral particles launched per triangle/quad the matrix entries should be:

M\_qi\_wsrc\_wdest = Number of particles launched from wall element index “wsrc” that impact on wall element “wdest” at charge state qi (qi includes 0 i.e. deposited as neutral)

File format wise I would vote for JSON, its human readable can easily be parsed in python and as standardized files go it has low formatting overhead (compared to XML for instance).