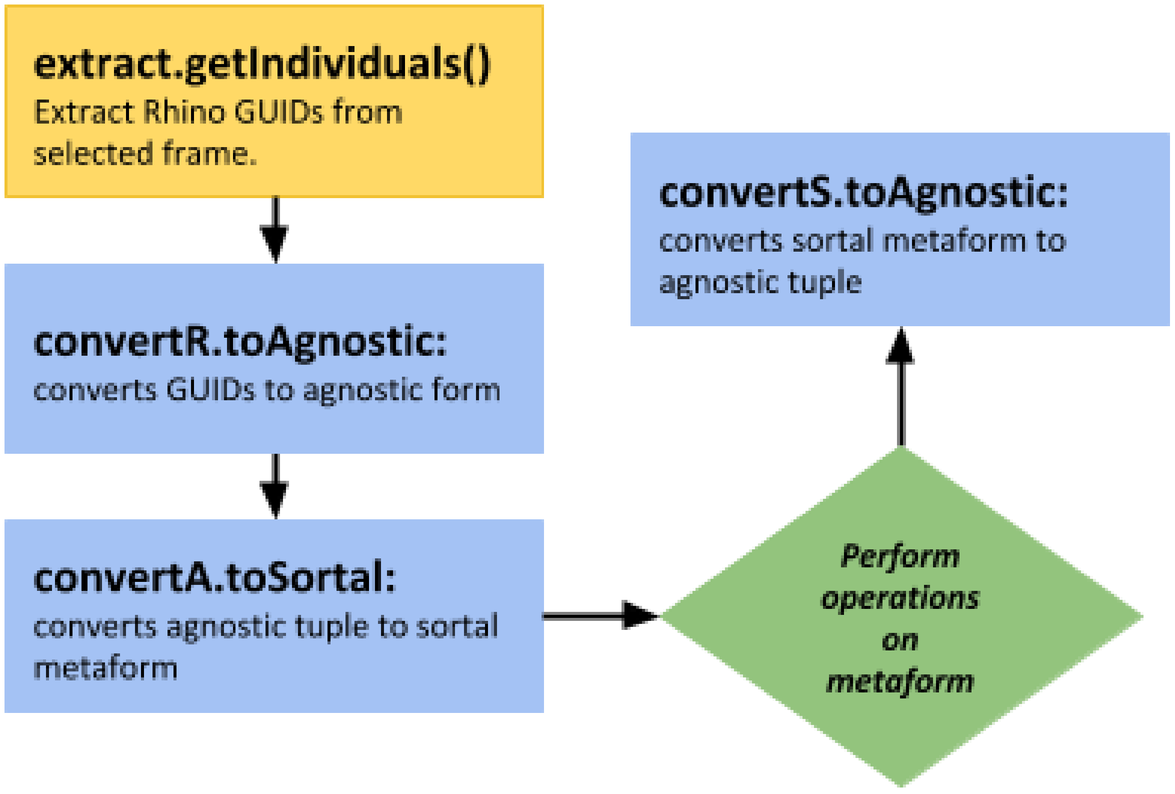
Annex B: Conversion Classes

The conversion codes are stored in the demo scripts -> 'convert' (extract, convertR) and 'sortalgi' -> 'sortal_lib_api' -> 'convert' (convertA, convertS) folder. They handle extraction and conversion between Rhino GUID, agnostic and sortal formats. There are four functions in this folder, namely:

- extract
- convertR
- convertA
- convertS

A typical workflow involving these functions may be as follows:


extract.getIndividuals()
Extract Rhino GUIDs from selected frame.
convertR.toAgnostic:
converts GUIDs to agnostic form
convertA.toSortal:
converts agnostic tuple to sortal metaform
Perform operations on metaform
convertS.toAgnostic:
converts sortal metaform to agnostic tuple

In the following example, the shape is extracted as a list of GUIDs corresponding to the objects inside the frame selected. The reference point (the insertion point of the frame block instance) and the layer name of the layer where the objects are drawn, are also returned.

```
shape, refPointMain, layerNameMain = extract.getIndividuals('Select the frame
containing the main form')
```

This shape is then converted from Rhino GUIDs to an agnostic tuple.

```
# Converts list of Rhino GUIDs to agnostic form and sorted GUIDs
shapeAgnostic, shapeIds = convertR.toAgnostic(shape, refPointMain)
```

The agnostic tuple ‘shapeAgnostic’ is fed to the method ‘find_rule_appns’, where it is converted to a sortal metaform.

```
shapeSortal = convertA.toSortal(shapeAgnostic)
```

The resulting sortal metaform undergoes operations:

```
convertS.toAgnostic(shapeApps[index2].perform(shapeSortal.duplicate())
```

where ‘shapeApps[index2]’ is an rule application instance. The sortal metaform, ‘shapeSortal’, is converted to an agnostic tuple right after the operation.

extract

extract’s method, ‘getIndividuals()’, takes a string as input. It uses this string input to prompt for the frame of a shape, for example:

```
lhs, refPointL, layerNameL = extract.getIndividuals('Select the frame
containing the LHS')
```

It prompts for selection of a frame block instance which houses Rhino objects inside, and returns a list of GUIDs extracted from the frame; the reference point or insertion point of the frame block instance; and the layer name where the frame and shape are located.

convertR

convertR is a class of functions that converts Rhino geometry to agnostic tuple form. The specific method to this is called ‘toAgnostic()’, which takes as input a list of Rhino GUIDs and a reference point (either a GUID or a tuple of coordinates). In the case of the demo scripts, the reference point is usually the insertion point of the frame block instance they were extracted from.

Reference points are necessary when defining rules, so that the LHS and RHS can be converted to agnostic tuple form in reference to the frames they are housed in. When no reference point is given, the default value of the reference point inside convertR is the origin (0, 0, 0). It returns a tuple describing the shape in agnostic form and the sorted list of Rhino GUIDs that describe the shape.

The class convertR needs to first be instantiated before using it. For example:

```
conversion = convertR()
```

```
agnosticTuple, rhinoObjs = conversion.toAgnostic( rhinoList, refPt)
```

where ‘rhinoList’ is a list of Rhino GUIDs and ‘refPt’ is a GUID representing a Point 3D geometry. The variable ‘agnosticTuple’ stores the returned agnostic tuple form of ‘rhinoList’, and ‘rhinoObjs’ contains the GUIDs of the Rhino geometry included in the agnostic tuple.

In the case of using a frame to obtain a reference point, as in the ‘create_rule’ function:

```
lhs, refPointL, layerNameL = extract.getIndividuals('Select the frame
containing the LHS')
```

```
lhsIndiv, lhsObjects = convertR.toAgnostic(lhs, refPointL)
```

Note that the outputted list of Rhino GUIDs is considered as ‘sorted’ because the conversion class will only generate agnostic tuple forms for the geometric sortal objects that have been initialized in ‘set_up_all.py’. For example, if the disjunctive sort defined by ‘set_up_all.py’ contains points but not line segments, and the frame selected contains a shape with points and line segments, then convertR.toAgnostic will not return any line segments in the list of GUIDs returned.

convertA

‘convertA.toSortal()’ takes the agnostic tuple describing a shape as an input, and returns a sortal metaform. Its parsing and creation of sortal individuals and their attributes is based on the disjunctive sort defined when ‘set_up_all.py’ is run. To use this method:

```
lhsMetaform = convertA.toSortal(lhsIndiv)
```

convertS

‘convertS.toAgnostic’ takes the metaform as input, and returns the agnostic interpretation of the metaform. This example is found in ‘find_rule_appns.py’:

```
match = convertS.toAgnostic(shapeApps[index2].matchForm)
```

where ‘shapeApps[index2].matchForm’ is a metaform, and ‘match’ stores the generated agnostic tuple form of the metaform.