```
// Example protocol demonstrating the use of the Sample
function
protocol Sample // this is the name of the protocol that will
be called in a workflow or other antha element

// we need to import the wtype package to use the LHComponent
type
// the mixer package is required to use the Sample function
import (
    "github.com/antha-lang/antha/antha/anthalib/wtype"
    "github.com/antha-lang/antha/antha/anthalib/mixer"
)


// Input parameters for this protocol (data)
Parameters (
    // antha, like golang is a strongly typed language in which
the type of a variable must be declared.
    // In this case we're creating a variable called
SampleVolume which is of type Volume;
    // the type system allows the antha compiler to catch many
types of common errors before the programme is run
    // the antha type system extends this to biological types
such as volumes here.
    // functions require inputs of particular types to be
adhered to
    SampleVolume Volume
)

// Data which is returned from this protocol, and data types
Data (
    // Antha inherits all standard primitives valid in golang;
    //for example the string type shown here used to return a
textual message
    Status string
)


// Physical Inputs to this protocol with types
Inputs (
    // the LHComponent is the principal liquidhandling type in
antha
    // the * signifies that this is a pointer to the component
rather than the component itself
    // most key antha functions such as Sample and Mix use
*LHComponent rather than LHComponent
    // since the type is imported from the wtype package we
need to use  *wtype.LHComponent rather than simply *LHComponent
    Solution *wtype.LHComponent
)
```

```
// Physical outputs from this protocol with types
Outputs (
    // An output LHComponent variable is created called Sample
    Sample *wtype.LHComponent
)

Requirements {

}

// Conditions to run on startup
Setup {

}

// The core process for this protocol, with the steps to be
performed
// for every input
Steps {

    // the Sample function is imported from the mixer library
    // in the mixer library the function signature can be
found, here it is:
    // func Sample(l *wtype.LHComponent, v wunit.Volume)
*wtype.LHComponent {
    // The function signature  shows that the function requires
a *LHComponent and a Volume and returns an *LHComponent
    Sample = mixer.Sample(Solution,SampleVolume)

    // The Sample function is not sufficient to generate liquid
handling instructions alone,
    // We would need a Mix command to instruct where to put the
sample

    // we can also create data outputs as a string like this
    Status = SampleVolume.ToString() + " of " +
Solution.ToString() + " sampled"

}
// Run after controls and a steps block are completed to
// post process any data and provide downstream results
Analysis {
}

// A block of tests to perform to validate that the sample was
processed
//correctly. Optionally, destructive tests can be performed to
validate
//results on a dipstick basis
```

```
Validation {

}
```