```
scala> // import the SCSCPClient and OpenMath libraries
scala> import info.kwarc.mmt.odk.SCSCP.Client.SCSCPClient
scala> import info.kwarc.mmt.odk.OpenMath._

scala> // establish a connection
scala> val client = SCSCPClient("scscp.gap-system.org")
client: info.kwarc.mmt.odk.SCSCP.Client.SCSCPClient =
info.kwarc.mmt.odk.SCSCP.Client.SCSCPClient@48d61b48

scala> // get a list of supported symbols
scala> client.getAllowedHeads
res0: List[info.kwarc.mmt.odk.OpenMath.OMSymbol] =
List(OMSymbol(Size,scscp_transient_1,None,None),
OMSymbol(Length,scscp_transient_1,None,None),
OMSymbol(LatticeSubgroups,scscp_transient_1,None,None),
OMSymbol(NrConjugacyClasses,scscp_transient_1,None,None),
OMSymbol(AutomorphismGroup,scscp_transient_1,None,None),
OMSymbol(Multiplication,scscp_transient_1,None,None),
OMSymbol(Addition,scscp_transient_1,None,None),
OMSymbol(IdGroup,scscp_transient_1,None,None),
OMSymbol(Phi,scscp_transient_1,None,None),
OMSymbol(SCSCPStartTracing,scscp_transient_1,None,None),
OMSymbol(SCSCPStopTracing,scscp_transient_1,None,None),
OMSymbol(Identity,scscp_transient_1,None,None),
OMSymbol(IsPrimeInt,scscp_transient_1,None,None),
OMSymbol(Factorial,scscp_transient_1,None,None), OMSymbol(Determinant,scscp_t...

scala> // We make a simple example: Apply the identity function to an integer 1
scala> val identitySymbol = OMSymbol("Identity", "scscp_transient_1", None, None)
identitySymbol: info.kwarc.mmt.odk.OpenMath.OMSymbol =
OMSymbol(Identity,scscp_transient_1,None,None)
scala> val identityExpression = OMApplication(identitySymbol, List(OMInteger(1, None)), None,
None)
identityExpression: info.kwarc.mmt.odk.OpenMath.OMApplication =
OMApplication(OMSymbol(Identity,scscp_transient_1,None,None),List(OMInteger(1,None)),Non
e,None)
scala> client(identityExpression).fetch().get
res1: info.kwarc.mmt.odk.OpenMath.OMExpression = OMInteger(1,None)

scala> // We also try to compute 1 + 1
scala> val additionSymbol = OMSymbol("Addition", "scscp_transient_1", None, None)
additionSymbol: info.kwarc.mmt.odk.OpenMath.OMSymbol =
OMSymbol(Addition,scscp_transient_1,None,None)
```

```
scala> val additionExpression = OMApplication(additionSymbol, OMInteger(1, None) ::
OMInteger(1, None) :: Nil, None, None)
additionExpression: info.kwarc.mmt.odk.OpenMath.OMApplication =
OMApplication(OMSymbol(Addition,scscp_transient_1,None,None),List(OMInteger(1,None),
OMInteger(1,None)),None,None)
scala> client(additionExpression).fetch().get
res2: info.kwarc.mmt.odk.OpenMath.OMExpression = OMInteger(2,None)

scala> // and close the connection
scala> client.quit()
```