This section defines the data files which specify what instruments are available. The precise definition of the file syntax is provided in Backaus-Naur form in figure 1. It must

a) begin with a "type" field consisting of `type=` followed by a description of a type which the system knows. This can be a constant, shorthand for one of the default generic types (at the moment: `MIDI`) or a file name (ending in `.ck`). In the case of a file name the system will check the first line for `// type=`. It will then attempt to match the remainder of the line to a constant defined in the `Server.ck` file, at which point it should add the file to the virtual machine and instantiate an object. Note that this code will have to be added to `Server.ck` when a new instrument is added by a `.ck`.

b) follow this with a name, specified by `name=`*name*

c) follow this with 0 or more translation lines.

A file with 0 translation lines would cause the server to instantiate the object specified, and may therefore be of some use.

⟨*file*⟩	::=	⟨*type*⟩ ⟨*linebreak*⟩ ⟨*name*⟩ ⟨*linebreak*⟩ (⟨*translation*⟩ ⟨*linebreak*⟩) *
⟨*type*⟩	::=	‘‘`type=`’’⟨*type-string*⟩ ⟨*linebreak*⟩
⟨*type-string*⟩	::=	‘‘`MIDI`’’
⟨*name*⟩	::=	‘‘`name=`’’⟨*name-string*⟩ ⟨*linebreak*⟩
⟨*name-string*⟩	::=	‘`[a-zA-Z][a-zA-Z0-9_]*`’
⟨*translation*⟩	::=	⟨*osc-message-desc*⟩ ‘`=`’ ⟨*output-message-desc*⟩ ⟨*linebreak*⟩
⟨*osc-message-desc*⟩	::=	‘`"`’⟨*osc-addr-pat*⟩ ⟨*osc-typetag*⟩ ‘`"`’
⟨*osc-addr-pat*⟩	::=	(‘`/`’ ⟨*osc-string*⟩)+
⟨*osc-typetag*⟩	::=	‘`, `’ ⟨*osc-type*⟩+
⟨*osc-type*⟩	::=	‘`i`’ | ‘`f`’ | ‘`s`’ | ‘`b`’
⟨*osc-string*⟩	::=	‘`[^\0]+`’
⟨*output-message-desc*⟩	::=	⟨*midi-message*⟩ | future message types
⟨*midi-message*⟩	::=	⟨*midi-stat-byte*⟩ ‘`,`’ ⟨*midi-data-byte*⟩ ‘`,`’ ⟨*midi-data-byte*⟩
⟨*midi-stat-byte*⟩	::=	128–255
⟨*midi-data-byte*⟩	::=	0–127 | ⟨*osc-arg*⟩
⟨*osc-arg*⟩	::=	‘`$`’[0-9]+

Figure 1: Data file grammar