System Configuration Files

All the configurable parameters of the system are set using *system configuration files* (***.scf**). These are plain text files, and they use the following basic syntax—formulated in EBNF:

```
component ::= "[" identifier ("." identifier)* "]"
property ::= identifier "=" (constant | "{" option (";" option)* "}")
option ::= (Default | User)

Default ::= "DEFAULT" ":" definition
User ::= "USER" ":" definition

comment ::= "--" text
```

Typical Uses

System components are identified by square brackets, and the settings of the component properties are specified between braces under that section.

The following table provides a few typical examples.

Type	Examples
Component definition	`[system]`
	`[database]`
	`[database.tables]`
Property definition	`block_size = {SYSTEM: 4096; USER: 1024}`
	`users.count = {USER: 150}`
	`aligned = true`
	`cloud_use = {DEFAULT: Local}`
	`self_certificate = false`

Each component or property has to be on a separate line, and line breaks are assumed to terminate the definition.

The following is an example of a **valid** layout:

```
[database]
table_count = 50
record_size = {DEFAULT: 1024; USER: 4096}
```

The following is an example of an **invalid** layout: a section id and a property name are on the same line, and the settings of the ğroperties are on the next lines.

```
[disks] block_size
= 2048
```

```
  replicated =
  true
```

If a component specified does not exist or the value set for a property is not within a valid range or one of the predefined constants, the definition is ignored.

Setting	Error
`[bicycles]`	The component doesn't exist
`frizzles = {DEFAULT: anything; USER: honky-dory}`	The property doesn't exist, and the settings are not recognized
`disk_size = {ADMIN: sys_admin}`	The option does not exist

Configurable Settings

The following table enumerates the components that exist in the system and their properties along with the setting data types and allowable values. Each property can get a *default* value which then can be overridden with a *user* value, as stated in the above EBNF specification for options.

Component	Property	Type	Acceptable Values
system	*id*	string	Any arbitrary identifier assigned by the user
	locale	string	Valid locale settings are identical to those of all operating systems, e.g. *en-US*, *fr-FR*, *it-IT*, etc.
	cloud_use	enumerated	Specifies whether the system will use cloud as storage. Can be either *Local* or *Global*. If *Local*, cloud access and certificates have to be managed locally. If *Global*, the system will manage the certificates from a pool
system.databases	*count*	integer	The number of databases in the system
	backup	boolean	Whether the system databases will be backed up
	auto_resize	boolean	Whether the system databases will be resized automatically. If false, then the resize logic has to be specified with the next property
	resize_by	integer	The resize value. Has to be a multiple of 1024
database	*id*	string	Any arbitrary identifier assigned by the user. Has to be the name of a specific database
	type	enumerated	Type of the database. Has to be either *Flat* or *Relational*. If *Flat*, no database keys are maintained by the system
	replicated	boolean	Specifies whether the databases will be replicated on multiple sites
	user_count	integer	The number of users that will be accessing the database
database.tables	*count*	integer	Number of tables in the database

Component	Property	Type	Acceptable Values
	block_aligned	boolean	Whether the table will be aligned on the disk for faster access. If *true* is specificed the next value must also be set
	block_size	integer	The block size to be used to align the records in the table. Must be a multiple of 1024