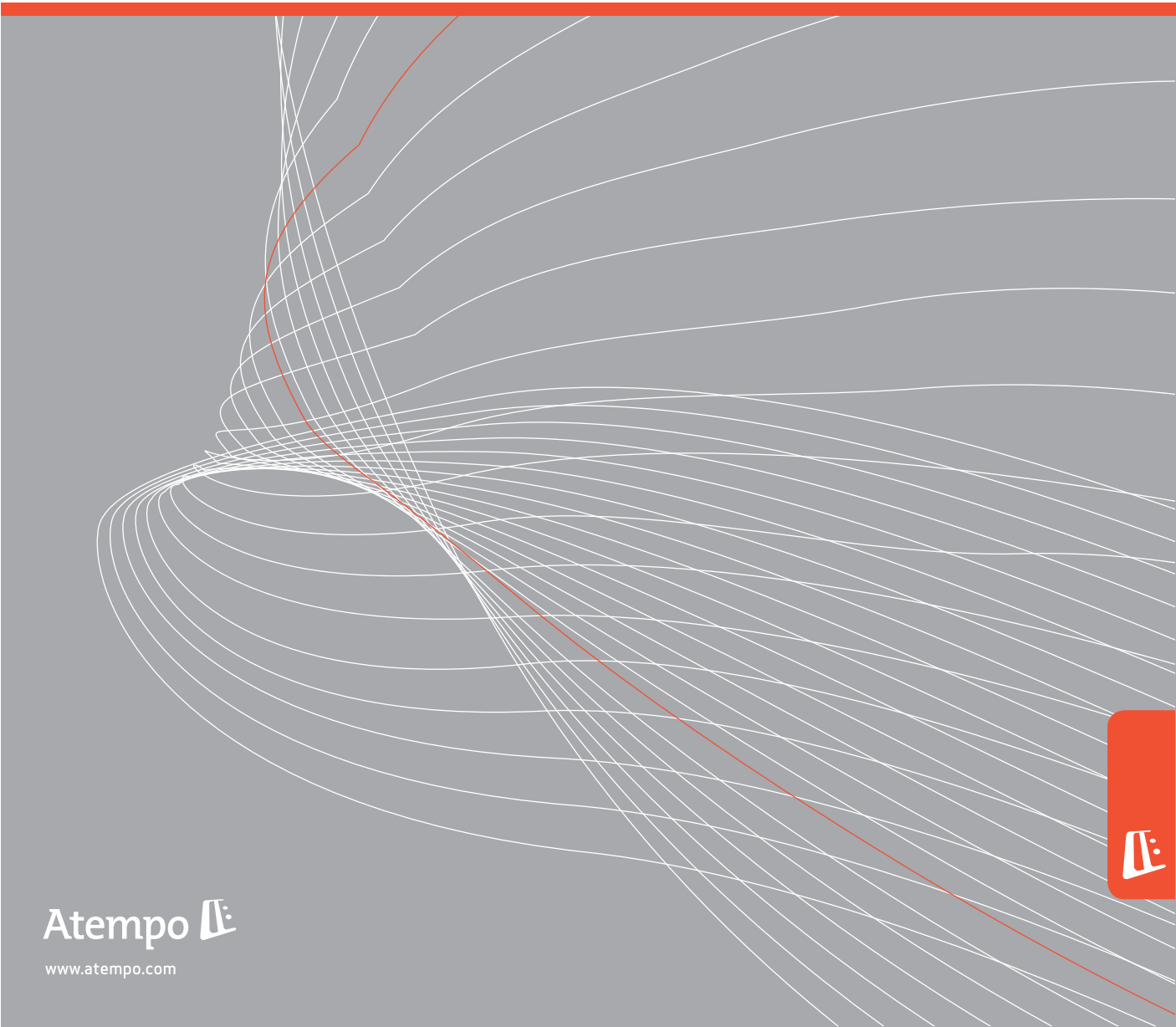
Version 4.2 SP8

Ref: shel42e_sp8_rev1_20111028

Time Navigator Shell Scripting



Atempo

www.atempo.com

Atempo Copyrights

Atempo Web Support

If you have subscribed a maintenance contract and you encounter a problem with **Time Navigator**, consult the Atempo Web Support at:
http://support.atempo.com

If you are unable to solve your problem or find the information you need, Atempo and its partners will help you. Web support includes an interactive interface through which you can log your inquiries directly and follow them up. Support is also available by email and by phone.

When calling the Atempo Technical Support team, please provide your maintenance contract number. We also recommend that for all Support inquiries you generate the Environment Report. For more details on the Environment Report, see the Troubleshooting chapter of the **Time Navigator Administration Guide**.

Your Comments are Welcome

We value and appreciate your opinion as a **Time Navigator** user and reader of our documentation. As we write, revise and evaluate our guides, your comments are the most important input we receive. Please do not hesitate to send us any remarks you have to the following address: **documentation@atempo.com**

Table of Contents

Introduction

Overview

tina_shell Language

Manipulation of Objects

Objects and their Attributes

Appendix

Introduction

Welcome

Thank you for choosing **Time Navigator**, superior software developed by Atempo. The prime automated solution for backing up, archiving and restoring data. Its flexible and dynamic architecture brings both performance and security to your data storage need.

Time Navigator Shell Scripting is a command interpreter that allows the administrator to configure **Time Navigator**.

The Guide

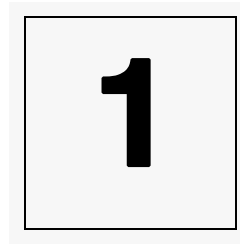
The guide is organized as follows:

- Chapter 1 gives a short description of **Time Navigator Shell Scripting**: the application fields and the function modes, such as the fundamental principles of the API **Time Navigator**.
- Chapter 2 describes the syntax of the language `tina_shell`.
- Chapter 3 presents the manipulation methods of the objects using their attributes and also using the on-line help.
- Chapter 4 describes all the objects and their different attributes. It also describes the links between the objects.

The Appendix gives the conventions in order to homogenize scripts. It also gives examples of `tina_shell` scripts and provides the list of `tina_shell` commands.

CHAPTER 1

Overview



This chapter presents **Time Navigator Shell Scripting** by giving a short description of the **Time Navigator** API (Application Programming Interface), and by providing a few examples.

Introduction

Time Navigator Shell Scripting is composed of a command shell using a programming interface of the **Time Navigator** API, and its related documentation.

The command shell is used in command line, `tina_shell` being the name of the binary. The `tina_shell` command allows users to administer a **Time Navigator** configuration by developing procedures in simple language. This enables interaction with the product administration database, the catalog.

It contains a number of commands which are intentionally restricted, allowing the handling of shell variables, together with **Time Navigator** objects.

A good knowledge of **Time Navigator**, and its administration interface `tina_adm` is required to use `tina_shell`.

☞ For more information, refer to the **Time Navigator Administration Guide** and **Time Navigator Restore Guide**.

Time Navigator Shell Scripting is a **Time Navigator** option of which the `tina_shell` binary is installed under the same name as the other executables. It is found in the `tina/Bin` directory.

Application Fields

Time Navigator Shell Scripting can be used in the following contexts (the list is non-exhaustive):

- Automated deployment of the **Time Navigator** configuration on one or more sites having one or more backup servers. This deployment rests on the use of " masters " of installation and configuration.
- Advanced interfacing with **Time Navigator** and a follow-up management tool of exploitation.
- Advanced interfacing with a software scheduler.
- Automation of regular tasks: preventive recycling, etc.
- Collection of information to create personalized reports and for statistical analysis.

Warning An application of **Time Navigator Shell Scripting** is considered a client of the **Time Navigator** server. Only the "client" functionalities are available. For security reasons, the management of user profiles and changing the password of the user `tina` is impossible.

Warning The availability of a new scheduler object in the API will require recompilation of scripts.

API Time Navigator

The API **Time Navigator** is a programming interface composed of a library of functions written in C. It offers a great number of disposable functionalities via **Administration Console** and **Restore & Archive Manager**. This is object oriented programming where the information and functions of **Time Navigator** are structured in class objects.

To use **Time Navigator Shell Scripting**, it is preferable to be familiar with certain concepts of the API, class objects and their attributes.

Object Classes

Time Navigator API uses the concepts of *classes* and *attributes*.

Sixteen object classes are available to users.

Classes correspond to objects that users can handle in `tina_adm`, for example the objects `Host`, `Drive`, ...

Here are the classes listed alphabetically:

- `AccessGroup`
- `Alarm`
- `Application`
- `Archive`
- `Backup`
- `BackupClass`
- `Cartridge`
- `Catalog`
- `Drive`
- `Folder`
- `Host`
- `HostGroup`
- `Job`
- `Library`
- `Strategy`
- `User`

Object Attributes

Each class is characterized by a certain number of attributes.

The value of each attribute can either be read (`get`) or modified (`set`).

For example, three out of twenty-eight attributes of the class `Host` are:

- `HostName` : its name.
- `HostType` : its type (HP, SGI, etc.).
- `HostEnable` :: its status (enabled/disabled).

☞ For more information about object attributes, refer to “Objects and their Attributes”, page 43.

Object Operations

Classes can be instantiated and handled by means of the six following operations:

- `create`: creates an object in the **Time Navigator** catalog.
- `open`: opens an existing object in the **Time Navigator** catalog.
- `get`: gets the value of one or several attributes.
- `set`: modifies the value of one or several attributes.
- `close`: closes an object previously opened by `open` or created with `create`.
- `delete`: deletes an object existing in the **Time Navigator** catalog.

☞ For more information about object manipulation, refer to “Manipulation of Objects”, page 29.

Functional Modes and Starting Up of tina_shell

Interactive Mode or Shell Mode

In the interactive mode, the commands are immediately interpreted and executed by `tina_shell`.

Launch the shell mode by using the following command:
`tina_shell [-catalog catalog] [-language language] [-version] [-help]`

The parameters are:

- `-catalog catalog` specifies the server catalog if more than one catalog is available.
- `-language Chinese|English|French|Korean|Spanish` defines the language.
- `-version` is the installed version of the product.
- `-help` displays on-line help.

The Programming Mode

In the programming mode, `tina_shell` executes a group of commands initially defined in a file (script).

Launch the shell mode by using the following command:

```
tina_shell -file file [-catalog catalog] [-language language]
[-version] [-help]
```

The parameters are:

- `-file` specifies the path of the command file. The path can be either absolute or relative. The absolute path is OS dependent. For example:
  - Relative path is `-file Test.tsh`
  - Windows absolute path is `-file c:\Test.tsh`
  - Unix absolute path is `-file /tmp/Test.tsh`
- `-catalog catalog` specifies the server catalog if more than one catalog is available.

You can add arguments to perform specific processes that are read by the command file.

Note It is recommended that you document scripts. The character # placed at the beginning of a line allows comments to be included.

☞ Refer to “Examples of tina_shell Scripts”, page 126 for examples of scripts.

To avoid confusion between parameters used in tina_shell and those for scripts you must:

- Enter the parameters just behind the tina_shell command.
  For example :

```
tina_shell param1 param2 -file fic.tsh -catalog cata
```

or

- Enter the parameters anywhere so long as they are behind one or more specific options:
  For example :

```
tina_shell -file fic.tsh -myoption param1 param2 param3 -catalog
cata
```

Quitting tina_shell

There are two ways to quit `tina_shell`:

- The `quit` command, which is interactive and asks for a confirmation.
  It can only be used in shell mode and always outputs a 0 status.
- The `exit` command, which takes an integer or an integer variable as an argument which is an error code used as output status.
  The command is useful in script mode.

CHAPTER 2

tina_shell Language

2

This chapter explains how variables, objects, conditional branches and loops are managed in `tina_shell`.

tina_shell Variables

Variable Types

`tina_shell` manages the following six variables types:

- `int` : integer variable between (-2^{32} /2) and ($+(2^{32}/2)-1$) which is equivalent to -2,147,483,648 to + 2,147,483,647
- `string` : string of characters
- `intlist` : integer list
- `stringlist` : list of strings of characters
- `handle` : link to a **Time Navigator** object (see the section *Time Navigator Objects*)
- `handlelist` : list of links to a **Time Navigator** object

Variable Naming Conventions

In all our examples, variable types are explicitly provided in the variable names as follows:

- `int` :INT_MY_VARIABLE
- `string` :STR_MY_VARIABLE
- `intlist` :INT_LIST_MY_VARIABLE
- `stringlist` :STR_LIST_MY_VARIABLE

- handle :HDL_MY_VARIABLE
- handlelist :HDL_LIST_MY_VARIABLE

Existing Variables

Several variables are created when tina_shell is launched:

- ARGV : stringlist (5 strings) complete chain of command.

  ARGV[0 is the binary name, usually tina_shell (this name can be different if there is a symbolic link or if the binary one was renamed).

  ARGV[1 is the first argument of the command.
- EMPTY_STRING : empty string (string).
- NULL STRING string filled with zeroes
- EMPTY_STRING_LIST forces a string attribute on an empty list.
- EMPTY_HANDLE_LIST forces a handle attribute on an empty list.
- EMPTY_INT_LIST forces an integerstring attribute on an empty list.
- TRUE : (int) = 1
- FALSE : (int) = 0
- INVALID_HANDLE = unknown handle
- NULL_HANDLE = unknown handle

General Variable Operations

Creating Variables

You can create variables using the following command :

```
variable type_name value
```

Where type is the variable type to be created (int, stringlist ...), name is the name of the variable to be created, and value is a value to be attributed to this variable.

The line: tina_shell > variable INT_NUMBER 3
creates a variable of integer type called INT_NUMBER with a value of 3.

```
tina_shell > variable stringlist STR_LIST_WORDS sea sand sun
```

creates a variable called STR_LIST_WORDS, which is a list of strings of characters into which the three strings are stored in order.

To create variables which are not lists, the last argument of this command can also be a type variable compatible with the declared type.

Example

```
tina_shell > variable int INT_A 3
tina_shell > variable int INT_B INT_A
tina_shell > echo INT_B
3
tina_shell
```

In this example, the value of Variable INT_A is 3. You then create a variable INT_B of the int type whose value is INT_A, in other words 3.

Note If you want to assign a value that is a standard string, but that the string is also a variable name, insert this string between quotation marks so that it will not be interpreted as a variable.

Warning The variables of the handle type are created when objects are opened. The variable command handle HANDLENULL null allows only the creation of a handle Null.

Deleting Variables

You can remove one or more variables with the following command:

```
erase variable1 variable2
```

- `tina_shell > erase INT_NUMBER STR_LIST_WORDS`

  Removes the variables INT_NUMBER and STR_LIST_WORDS.

Viewing All the Variables

View the variables by using the following command:

```
show
```

This command displays all the existing variables at any given moment; for example, the default variables and the created variables. It displays two parts: the variable part (on the left), and the attribute part (on the right).

Example

```
tina_shell > show
variables                                    attributes
ARGV              3 string(s) list           HostName =              &STR_NAME
EMPTY_STRING      string                     HostType =              &INT_TYPE
NULL_STRING       string                     HostEnable =            &INT_STATUS
TRUE              1
FALSE             0
INVALID_HANDLE    Unknown handle
NULL_HANDLE       Unknown handle
STR_LIST          3 string(s) list
INT_NUMBER        6
tina_shell > _
```

With this example, two variables were created*:*
- `INT_NUMBER` *of type* `int` *whose value is 6, and*
- `STR_LIST` *of type* `stringlist` *containing 3 elements*.

Three attributes were directed*:*
- `HostName` *whose value is recovered in the variable* `STR_NAME`*,*
- `HostType` *whose value is recovered in the variable* `INT_TYPE`*, and*
- `HostEnable` *whose value is recovered in the variable* `INT_STATUS`.

Displaying the Variable Values

You can display the value of the variables using the following command:

```
echo variable1 variable2
```

Used without arguments, this command jumps a line. In this case, the command `echo` simply displays the arguments, unless an argument is a variable name. In this case, the value of the variable is displayed instead of its name.

Example To display the variable value INT_NUMBER:

```
tina_shell > echo INT_NUMBER
3
tina_shell >
```

Note If you want to display a character string which is also a name of variable, insert it between quotation marks ("). In addition, if you want to display a double quote, insert a backslash before the double quote.

Specific Variable Operations

Type int Variables

Increment and Decrement

To increment or decrement variables, use the following commands:

```
increment name value_increment
```

```
decrement name value_decrement
```

Where `name` is the name of the integer type onto which the operation is performed.

Where `value_increment` and `value_decrement` set the increment or decrement values. This can be specified by a raw integer value or by an integer variable of type `int`.

In the following example, the two cases are encountered:

Example

```
tina_shell > variable int INT_A 2
tina_shell > variable int INT_B 1
tina_shell > increment INT_A 3
tina_shell > echo INT_A
5
tina_shell > increment INT_A INT_B
6
tina_shell > decrement INT_A 3
tina_shell > echo INT_A
3
tina_shell > decrement INT_A INT_B
tina_shell > echo INT_A
2
tina_shell >
```

Multiply

The `multiply` command works with integer variables.

```
multiply variable1 variable2 name
```

Where `variable1` and `variable2` are two variables of type `int` containing the values to be multiplied.

Where `name` is the name of the variable created to contain the result.

```
tina_shell > variable int INT_A 2
tina_shell > variable int INT_B 3
tina_shell > multiply INT_A INT_B INT_RESULT
tina_shell > echo INT_RESULT
6
```

Percent

The `percent` command works like the `multiply` command, except that it calculates a percentage.

Example

```
tina_shell > variable int INT_A 3
tina_shell > variable int INT_B 4
tina_shell > percent INT_A INT_B INT_PERCENT
tina_shell > echo INT_A / INT_B = INT_PERCENT %
tina_shell > echo INT_PERCENT
75
tina_shell >
```

String Type Variables

Concatenation

The concatenation of 2 variables of type `string` is carried out with the following command:

```
concat STR_A STR_B
```

`STR_A` is the variable name of `string` type to be concatenated. If the variable does not exist, `STR_A` is litterally taken as the source. If the variable exists and is of the `string` type, the variable value is the same as the source.

`STR_B` is the variable name of `string` type, target of the concatenation. If the variable does not exist, it is created with the value of `STR_A`. If the variable exists, the value of `STR_A` is concatenated with it.

Example

```
tina_shell > variable string STR_A1 /Bin
tina_shell > variable string STR_A2 /tina
tina_shell > variable string STR_A3 /usr
tina_shell > concat STR_A1 STR_A2
tina_shell > echo STR_A2
/tina/Bin
tina_shell > concat STR_A2 STR_A3
tina_shell > echo STR_A3
/usr/tina/Bin
tina_shell >
```

Conversions

`tina_shell` allows the conversion of a string to an integer and vice versa.

Example

```
tina_shell > variable string STR_STR1 5
tina_shell > variable int INT_A STR_STR1
tina_shell > increment INT_A 1
tina_shell > variable string STR_STR2 INT_A
tina_shell > echo STR_STR1 INT_A STR_STR2
5 6 6
tina_shell >
```

Conversion of an integer into a character string: there are no requirements.

Conversion a character string into an integer: the character string representing a number must not contain any punctuation or spaces. For example, write 100000 and not 100,000 or 100 000.

Note If you specify a string as "1 000", the result is interpreted as "1". If you specify "x1000", where "x" is an empty space, the result is "0", without an error message.

List type variables

All list type variables (`intlist`, `stringlist`, `handlelist`) are tables.

The first element of a `tina_shell` table (or list) is the element **number 0**.

Access to Table Elements

A list element can be accessed using the character '**[**'.

Example
```
tina_shell > variable stringlist STR_LIST sea sand sun
tina_shell > echo INT_LIST
sea sand sun
tina_shell > echo INT_LIST[0
sea
tina_shell >
```

Counting Table Elements

The `item` command displays the number of elements in a table:

```
item variable number
```

Where `variable` is a variable of type `stringlist` containing a table where you want to count the number of elements.

Where `number` is a variable of type `int` which is created to contain the result of the count.

Example
```
tina_shell > variable stringlist STR_LIST sea sand sun
tina_shell > item STR_LIST INT_NUMBER
tina_shell > echo INT_NUMBER
3
tina_shell >
```

Adding an Element to a Table

You can add an element to the end of a list using the `add` command.

```
add variable in name
```

Where `variable` corresponds to the value to be added.

Where `name` is the the list into which the value of the value is added.

The variable types must be consistent.

If the variable `name` does not exist, it is created according to the type of variable to be added.

Example

```
tina_shell > variable string STR_STR1 sea
tina_shell > variable string STR_STR2 sun
tina_shell > add STR_STR1 in STR_LIST
tina_shell > add STR_STR2 in STR_LIST
tina_shell > echo STR_LIST
sea sun
```

Binary/Decimal Conversion

The two commands `mask` and `unmask` operate on tables of integers.

They allow you to easily handle attributes whose value is a bit mask.

```
mask variable1 variable2
```

The `mask` command generates a bit mask from all the integers present in a list.The first argument of this command is an integer list, the second one being a variable name which will be created of integer type that contains the mask value.

```
unmask variable1 variable2
```

Conversely, the `unmask` command generates an integer list from an integer representing a bit mask.

The first argument of this command is an integer representing a mask, the second is a variable name which is created of integer list type that contains the mask components.

Example The attribute `StrategyIncrPhaseTime` corresponds to the starting time of the

incremental backup. This attribute can take the following attributes:

```
tina_shell > help StrategyIncrPhaseTime
         1 : Heure 00
         2 : Heure 01
         4 : Heure 02
         8 : Hour  03
        16 : Hour  04
        32 : Hour  05
        64 : Hour  06
       128 : Hour  07
       256 : Hour  08
       512 : Hour  09
      1024 : Hour  10
      2048 : Hour  11
      4096 : Hour  12
      8192 : Hour  13
     16384 : Hour  14
     32768 : Hour  15
     65536 : Hour  16
    131072 : Hour  17
    262144 : Hour  18
    524288 : Hour  19
   1048576 : Hour  20
   2097152 : Hour  21
   4194304 : Hour  22
   8388608 : Hour  23
   or any combination of these in a mask
tina_shell > []
```

The `mask` command makes it possible to code into a single value any combination of hours, creating a checkmark in the Edition window of the Backup Strategy of **Administration Console**. This coding is equivalent to an addition of the various values corresponding to the time phases.
To launch the incremental backup at 6:00., 14:00. and 19:00.

```
tina_shell > variable intlist INT_LIST 64 16384 524288
tina_shell > mask LIST INT_MASK
tina_shell > echo INT_MASK
540736
```

Conversely, the `unmask` command decodes a value to provide the list of all the time phases. If the time phases notched in **Administration Console** correspond to value 540736, it is necessary to decode this value so that the time phases are transparent.

```
tina_shell > unmask MASK1 INT_LIST2
tina_shell > echo INT_LIST2
64 16384 524288
```

These values correspond to the hour phases 06, 14 et 19.

Conditional Branches

Four commands are used to perform tests with the conditional execution of some script branches.

if and endif Commands

The `if` command starts a test branch which must end by using the `endif` command.

The `if` command only tests variables of non-list type and compares them with other variables of the same type or with raw values.

```
if INT_A == 2
...
endif

if INT_A == INT_B
...
endif

if STR_NAME == speedy
...
endif
```

elif and else Commands

`elif`

`else`

These commands are optional and complete the last two commands.

```
if INT_A == 1
...
elif INT_A == 2
...
elif INT_A == 3
...
else
...
endif
```

Comparison Operations

The supported comparison operators are:

== Equal

!= Unequal

>= Greater than or equal

`<=` Less than or equal

`>` Greater than

`<` Less than

You can nest as many comparison levels as required.

Note You cannot do a comparison between two Handles, except for an equal comparison between two NullHandles.

Loops

`tina_shell` loops are managed by the following commands:

```
foreach name variable_list

endfor
```

These commands allow a variable to browse all the elements of a `list` type variable.

For example, if you want to browse all the strings of characters of the list `ARGV` created when `tina_shell` was started and containing all the parameters of the command line, you write:

Example

```
tina_shell > variable stringlist STR_LIST_WORDS sea sand sun
tina_shell > foreach STR_ELEM in STR_LIST_WORDS
tina_shell > echo STR_ELEM
tina_shell > endfor
sea
sand
sun
tina_shell >
```

You can nest as many loop levels as required.

Other Operations

Value Entry

A dialogue can be established between the user and `tina_shell`, using the command:

`input variable`

Where `variable` corresponds to the variable which is created. The variable must be a `string` type.

This command allows you to interact with the program by entering a string which is placed in the named variable.

If the type of variable required is an integer, conversion is achieved by using the `variable` command.

For more information about conversions of string variables, refer to “Conversions”, page 17.

Example

```
tina_shell >echo enter your user name:
tina_shell >input STR_NAME
? MyName
tina_shell >echo HELLO STR_NAME
hello MyName
tina_shell >
```

In this example, you enter your name which is recovered in the variable STR_NAME, and reused to display a greeting.

Access to Environment Variables

envget Command

```
envget variable1 variable2
```

This command retrieves the value of the environment variable specified in `variable1`, and places the value in `variable2` to be created as a `string` type. You can get this value directly or by using another variable.

Example

```
tina_shell > envget TINA_HOME STR_PATH
tina_shell > echo STR_PATH
/usr/tina
tina_shell >
```

Example

```
tina_shell > variable string STR_ENV_NAME TINA_HOME
tina_shell > envget STR_ENV_NAME STR_PATH
tina_shell > echo STR_PATH
/usr/tina
tina_shell >
```

envput Command

```
envput variable1 variable2
```

This command places the value contained in `variable2` in the environment variable named in `variable1`.

Example

```
tina_shell > variable string STR_ENV_NAME TINA_HOME
tina_shell > variable string STR_ENV_NAMEVAR_VALUE /tina
tina_shell > envput STR_VAR_NAME STR_VAR_VALUE
-or-
tina_shell > envput TINA_HOME /tina
```

Access to files

fileget Command

```
fileget file_path name [first_line number_of_lines]
```

This command retrieves the content of a text file in a `stringlist` type variable. The argument `file_path` corresponds to the file path from where you want to get the content. The variable `name` is created from this.

By default, the entire content of the file is retrieved, but it is possible to retrieve only a part of the file by using `[first_line number_of_lines]`.
These lines are numbered from zero. For example, to retrieve 3 lines starting at the third line, enter `[2 3]`. This retrieves lines 3, 4 and 5.

Example

```
tina_shell > fileget /usr/tina/Conf/catalogs LIST
tina_shell > foreach STRING in LIST
tina_shell > echo STRING
tina_shell > endfor
catalogs :
name = demo,
server = "speedy",
console = "speedy:0",
comment = "catalog of demonstration"
tina_shell >
```

fileput Command

```
fileput file_path name [first_line number_of_lines]
```

This command creates a text file containing the elements of a `stringlist` type variable. The argument `file_path` corresponds to the file path that one wants to create. The variable `name` is of the stringlist type and contains the list of values to be integrated into the file.

By default, the entire content of the variable is retrieved. It is possible to retrieve only part of the variable by using `[first_line number_of_lines]`. The lines are numbered from zero. For example, to retrieve 3 lines starting at the third line, enter `[2 3]`. This retrieves lines 3, 4 and 5.

Example
```
tina_shell > variable stringlist STR_LIST_PARAMS
parameter:language=English binary:tina_adm catalog:*
parameter:host_icon_size=14
tina_shell > fileput /usr/tina/Conf/parameters STR_LIST_PARAMS
tina_shell > exit 0
```

The parameters now contains the following text:
parameter:language=English
binary:tina_adm catalog:* parameter:host_icon_size=14

Time Commands

date command

```
date variable name
```

This command converts the number of seconds since January 1, 1970, contained in `variable`, to a date in the form of a meaningful string and written into the variable `name`. This command is very useful because the value of the number attribute is in number of seconds.

Example
```
tina_shell > time INT_TIME
tina_shell > date INT_TIME STR_DATE
tina_shell > echo STR_DATE
lun 19 fev 09:57 :56 2001
tina_shell >
```

time Command

```
time name
```

This command places into an integer variable the number of seconds since January 1, 1970.

Example
```
tina_shell > time INT_TIME
tina_shell > echo INT_TIME
982596030
tina_shell >
```

wait Command

```
wait number_seconds|variable
```

This command stops the program during a number of specified seconds (`number_seconds`), either directly, or through the use of a variable (`variable`). This allows, for example, solving problems of real time.

Example

```
tina_shell > wait 5
tina_shell > variable int INT_TEMPO 10
tina_shell > wait INT_TEMPO
tina_shell >
```

Functions

It is possible to write the functions that allow the use of factorized script code. Access to these functions is by three commands:

function_begin

The command `function_begin name parameters` begins function declaration. There must be at least one parameter.

Note The defined arguments of this command are purged after the execution of the function.

---

function_end

The command `function_end` ends the function declaration.

function_execute

The command `function_execute name parameters` executes the function. The parameters specified when calling this function are replaced by the parameters defined in the function declaration.

Note You can use variables that were not exclusively created for this function. You can also create variables in a function and use them again, after the execution of this function.

Short Mode

For a faster use in interactive mode, `tina_shell` interprets shortened commands (ex : `as` instead of `assign`).

A shortened command is interpreted because there is no ambiguity.
For example, the command `de` cannot be used because it could be interpreted as the command `decrement` or the command `delete`.

Warning We do not recommend using the short mode in a script, because the appearance of new commands could generate incompatibilities.

CHAPTER 3

Manipulation of Objects

3

Principles

You can act on many object classes of **Time Navigator**. The majority of these objects are visible in **Administration Console**. They are contained in the `tina_shell` variables, of the type `handle`. These variables make it possible to handle the **Time Navigator** objects in `tina_shell`.

`tina_shell` allows users to handle **Time Navigator** objects using the six following operations:

- Creation : `create`
- Opening : `open .`
- Recovering information : `get`
- Modifying information : `set`
- Close : `close`
- Deletion : `delete`

A view of the catalog contents in the **Time Navigator** server can be seen in `tina_adm`, as well as in `tina_shell`.

The catalog can be seen by two different interfaces. The modes are:

- Graphic mode
- Text mode (which can be programmed)

However, each handling in one of the interfaces ends its equivalent in the other, except for the `open` and `close` commands which are implicit in `tina_adm`. Once started, `tina_adm` automatically creates an image for each object of the catalog and deletes them all when you quit the application. Whereas in `tina_shell`, you must explicitly specify the objects whose images you want to keep and those whose images you want to delete.

On-line Help

```
help
```

This command allows access to on-line help.

- Used without arguments, it provides a list of all the commands, followed by a short description of the syntax, which also contains all the available object classes.
- Used with class object names, it provides specific help on the class which is listed in descending order in a table of the properties of the attributes related to the class.
- Used with an attribute name like an argument, it provides help specific to the attribute (displays a general list of possible values for the attribute).

Example: On-line help obtained by the command `help Host`

```
tina_shell > help Host
   Attributes                    Actions                    Type
   ----------                    -------                    ----
   HostName                      OPEN  CREATE   set    get  (string)
   HostType                       --   CREATE   set    get  (int)
   HostEnable                     --   create   set    get  (int)
   HostReportUnavailability       --   create   set    get  (int)
   HostServer                     --     --     --     get  (int)
   HostDiskSpace                  --   create   set    get  (int)
   HostKey                        --     --     set    get  (string)
   HostDrive                      --     --     --     get  (handle list)
   HostLibrary                    --     --     --     get  (handle list)
   HostRobot                      --     --     --     get  (handle list)
   HostHostGroup                  --     --     set    get  (handle)
   HostAlarm                      --     --     --     get  (handle list)
   HostBackupMaster               --   create   set    get  (handle)
   HostBackupMastered             --     --     --     get  (handle list)
   HostGraphicalMaster            --   create   set    get  (int)
   HostProtocolType               --   create   set    get  (int)
   HostProtocolTinaVersionMaj     --   create   set    get  (int)
   HostProtocolTinaVersionMin     --   create   set    get  (int)
   HostProtocolTinaVersionIndice  --   create   set    get  (int)
   HostProtocolNdmpVersionMaj     --   create   set    get  (int)
   HostProtocolNdmpVersionMin     --   create   set    get  (int)
   HostProtocolNdmpVersion        --   create   set    get  (int)
   HostProtocolNdmpUser           --   create   set    get  (string)
   HostProtocolNdmpPassword       --   create   set    --   (string)
   HostProtocolSymapiVersionMaj   --   create   set    get  (int)
   HostProtocolSymapiVersionMin   --   create   set    get  (int)
   HostSoftwareVersion            --     --     --     get  (string)
   HostStorageNode                --   create   set    get  (int)
tina_shell >
```

Attribute Column

This column lists all the corresponding attributes for each class of objects.

☞ For more information regarding different attributes, “Objects and their Attributes”, page 43.

Action Column

The four columns gathered under the *Action* heading reveal information on the actions (or operations).

Each column relates to an operation: `open`, `create`, `set`, or `get`. The operations `closed` and `delete` do not require attributes.

- If the name of the operation located opposite an attribute is in uppercase letters, this attribute is MANDATORY to carry out this operation.
- If the name of the operation located opposite an attribute is in lowercase letters, this attribute is defined, but it is OPTIONAL to carry out this operation.
- If no operation name appears opposite an attribute, this attribute is not DEFINED for this operation.

For example, for the creation operation, the `HostName` attributes and the `HostType` are mandatory. The `HostEnable` attribute is not mandatory but can be specified because it is defined, and the `HostDrive` attribute is prohibited because it is not defined.

The attributes of all the classes function in a similar way.

Type Column

This column specifies the variable type. There are 6 types:

`int` : integer variable
`string` : string of characters
`handle` : points to a **Time Navigator** object
`intlist` : table of integers
`stringlist` : table of character strings
`handlelist` : table of handles

Accessing Catalog Objects

Preparation

Handles by Access Type

With `tina_shell`, you must explicitly specify the object handles that you want and those that you want to erase.

Certain operations require at least an attribute and/or a handle (the image `tina_shell` of the object of the catalog), while others generate a variable of the `handle` type.

The list below summarizes the situation for each of the six operations:

	Requires attribute (s)	Requires a handle	Generates a handle
`create`	*		*
`open`	*		*
`set`	*		*
`get`	*		*
`delete`	*		
`close`	*		

Assigning attributes

The attributes can either be used, or modified at the time of the request to the `tina_shell` commands . For the four operations requiring one or more attributes (first column), you must assign a value to each attribute. The access to these attributes is done by read/write in a table of attributes which must be prepared in advance.

This preparation is carried out by means of the command `assign`.

Assign attributes for the create, open and set Commands

```
assign attribute value
```

Where `attribute` corresponds to the name of the concerned attribute.

Where `value` represents the value of the attribute that you want to assign.

The name and the purpose of the attributes to be specified depend on the class and the object created.

For example, before host creation, you must specify the name and the type.

```
tina_shell > assign HostName speedy
tina_shell > assign HostType 12
```

Entering these commands is the same as filling in the **Type** and **Host Name** fields in the Host Creation window of **Administration Console**.

☞ For more information about attributes related to different object classes, refer to "Objects and their Attributes", page 43.

Assign attributes for the get Command

```
assign attribute &name
```

Where `attribute` corresponds to the name of the attribute.

Where `name` is the name of the variable from which you get the value.

The name must be preceeded by the character "&" to indicate that you want to get the value.

Example

```
assign HostName &STR_NAME
assign HostType &INT_TYPE
assign HostEnable &INT_ENABLE
```

In the example above, you retrieve the values of the `STR_NAME`, `INT_TYPE`, and `INT_ENABLE` variables that correspond to the host name, the type and its current status (enabled/disabled), respectively.

The attribute table is reset to zero after each call of a command, but it can be directly removed with the following command:

```
reset
```

Note The results of the attribute table for the command `assign` can be viewed with command `show` (right column).

Assign attributes for the list Command

```
assign attribute value
```

Where `attribute` corresponds to the name of the concerned attribute.

Where `value` represents the value of the attribute that you want to assign.

Assigning an attribute to the `list` command is optional. This command only displays the objects corresponding to the assigned attributes.

Example

```
assign JobListAge 86400
assign JobListActivity 2
```

In the above example, along with the `list` command, you can view only the historic jobs which occurred in the last 24 hours.

☞ Refer to section “list Command”, page 39 for a detailed description of the `list` command.

Creation and removal of objects

Create Command

After you have assigned a value to the attributes, you must create an object to which they apply. This is the function of the command `create` :

```
create class variable
```

The `create` command creates an object of the `class` class by using the information of the assigned attributes. The `variable` parameter is the name of a variable of type `handle` which is created and which is used as a link between the user and the object created for further operations. This command resets the attributes.

Example *Creation of a Host*

```
tina_shell > assign HostName speedy
tina_shell > assign HostType 12
tina_shell > create Host HDL_SPEEDY
tina_shell >
```

The `create` command makes the creation of the object effective. This is the same as clicking on **OK** in the Host Creation window of **Administration Console**. The `SPEEDY` variable can then be used for each of the four operations requiring a handle.

Delete Command

```
delete variable
```

The `delete` command takes a variable of the handle type like an argument and deletes the corresponding object in the catalog, and its image in `tina_shell`.

Warning This operation cannot be undone.

Example *Deletion of the host Speedy*

```
tina_shell > delete HDL_SPEEDY
tina_shell >
```

Opening and Closing Objects

Open Command

```
open class variable
```

Opens an object of the `class` class by using the information of the assigned attributes. `variable` is the name of a variable of type `handle` which is created and which is used as a link between the user and the object opened in `tina_shell`.

For example, the on-line help of the Host class shows that only one attribute is necessary to open a host: the `HostName` attribute. The opening of a system is therefore carried out in two lines:

Example

```
tina_shell > assign HostName popeye
tina_shell > open Host HDL_POPEYE
tina_shell >
```

`popeye` is the object name in the catalog. HDL_`POPEYE` is the variable name of the type handle that is created, pointing to the catalog object.

Note This operation has no effect on the catalog.

Close command

```
close variable
```

The object disappears from `tina_shell` at the same time as the variable `variable`. There is no effect on the catalog.

Example

```
tina_shell > close HDL_POPEYE
tina_shell >
```

Erase Command

```
erase variable
```

Allows users, in a single command, to delete variables as well as attribute assignments. The `erase` command removes a variable of the type handle, but does not erase the object image from `tina_shell`. If two variables of `handle` type point to the same object, closing one of them makes the second one invalid and the latter must no longer be used, except for the `erase` command.

Reading Attributes

Get Command

```
get variable
```

The `get` command retrieves attribute information. The assigned attributes do not have to specify a value. The information can be a variable name that is the container for the retrieved value.

The variable must be of the type handle and with the correct attributes of the corresponding class. These variables are created with the corresponding attribute type.

Behind an attribute, you must define a destination (in the form of a variable) to receive the information. The destination is preceeded by an ampersand "&".

Example

```
tina_shell > assign HostEnable &INT_HOSTENABLE
tina_shell > assign HostType &INT_HOSTTYPE
tina_shell > get HDL_POPEYE
tina_shell > echo INT_HOSTENABLE INT_HOSTTYPE
1 32
tina_shell >
```

Modifying Attributes

Set Command

```
set variable
```

The `set` command modifies the attribute information. It is equivalent to editing, and requires the assignment of at least one attribute (authorized for this command) taking a type handle variable as an argument, which is the image of the object that you want to modify.

Example

```
tina_shell > assign HostEnable 1
tina_shell > set HDL_POPEYE
tina_shell >
```

In this example, the host moves from a disabled status to an enabled status.

Note For boolean parameters, you can assign `true` or `false` to the variables.

The number and purpose of attributes to be specified depend on the class of the object that is opened.

Listing Objects for a Class

list Command

```
list class in variable
```

The list object function generates a list of handles of all objects of `class` class known to the catalog. `variable` is the name of the `handlelist` type variable that is created. The following classes are supported:

- `AccessGroup`
- `Alarm`
- `Application`
- `BackupClass`
- `Cartridge`
- `Drive`
- `Folder`
- `Host`
- `Job`
- `Library`
- `Network`
- `User`

Example

```
tina_shell > list Host in HDL_LIST_HOST
tina_shell > echo HDL_LIST_HOST
(2 Host(s) handle list)
tina_shell >
```

You assign attributes to the `list` command to view objects corresponding to certain criteria.

☞ See the section “Assign attributes for the list Command”, page 35 for details.

The following attributes are supported:

Job

- `JobListActivity` — Lists jobs according to the specified activity.
  1 Lists active jobs.
  2 Lists historic jobs.
  3 Lists active and historic jobs.
- `JobListAge` — Lists jobs that have been run during the specified number of seconds.

Alarm

- `AlarmListSeverityList` — Lists alarms according to the specified severities.
  1 Lists minor alarms.
  2 Lists major alarms.
  3 Lists critical alarms.

Existence test

exist Command1

```
exist class variable
```

Where `class` corresponds to the assigned attributes.

Where `variable` is the name of an `int` type variable created to contain the result of the command, `true (1)` if the object exists and `false (0)` if it does not exist.

The existence test function (`exist` command) tells you whether an object exists in the catalog, using information on assigned attributes.

Hereafter is a list of the supported objects. To test these objects, use the attributes written between brackets.

- `AccessGroup` (`AccessGroupName`)
- `Alarm` (`AlarmId`)
- `Application` (`ApplicationName`)
- `Archive` (`ArchiveName`, `ArchiveFolder`)
- `BackupClass` (`BackUpClassPath`, `BackupClassHostGroup`)
- `Cartridge` (`CartrigeName`)

- Drive (DriveName DriveHost)
- DriveConnection (DriveConnectionHost, DriveConnectionDevice)
- Folder (FolderName)
- Host (HostName)
- Job (JobId)
- Library (LibraryName, LibraryHost)
- Network (NetworkName)
- Schedule (ScheduleName or ScheduleNameUtf8)
- ScheduleRule (ScheduleRuleId)
- Strategy (StrategyName, StrategyHostGroup)
- User (UserName)

Warning After you use the Existence test, the attribute used for the test is not reset to zero. As a result, opening after the test an attribute whose type is different from that of the attribute used for the test will cause an error.
To reset the attribute used for the Existence test to zero, use the `reset` command.

Example
```
tina_shell > assign ApplicationName aria.cat
tina_shell > exist Application INT_CAT
tina_shell > echo CAT
1
tina_shell >
```
This example states that there is an application whose name is `aria.cat`.

It is also possible to test for the existence of children applications

Example
```
tina_shell > assign ApplicationParent HDL_parent.app
tina_shell > assign ApplicationName Aria.cat
tina_shell > exist Application INT_CAT
tina_shell > echo CAT
1
tina_shell >
```
This example tests for the existence of an application name.

It takes into account the ApplicationParent attribute to see if the children application exists amongst the children of the parent application.

Example

➤ **Create a host Dilbert of type SGI and enable it**

1. Call the attributes using the `assign` command.

```
tina_shell > assign HostName dilbert

tina_shell > assign HostType 12

tina_shell > assign HostEnable TRUE
```

2. Enter the following command:

```
tina_shell > create Host HDL_DILBERT
```

The host Dilbert is created. It appears in **Administration Console** and is enabled.

The `assign` command takes on the first argument, an attribute name, and the second argument, the value of this attribute.

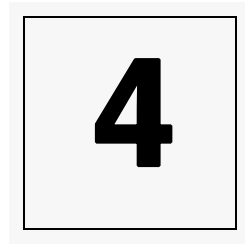
While entering the command `help HostType`, the user sees that 12 corresponds to the type SGI. So the value 12 is used as the value of the attribute `HostType`.

The `TRUE` value was given as the value of the attribute `HostEnable`.

The `TRUE` value was part of the four variables created at the launching of `tina_shell`, and its value is 1.

CHAPTER 4

Objects and their Attributes



Principles

The **Time Navigator** catalog contains object classes that you can manipulate using `tina_shell` variables of the type handle. To do this, you must know the attributes which characterize the corresponding object classes.

Each attribute corresponds to a **Time Navigator** function in **Administration Console**.

☞ For more information, refer to the **Time Navigator Administration Guide**.

All the attributes are not defined for each operation (`open`, `create`, `get` and `set`). For each object, on-line help provides an attribute table that gives the possible operations. This table of attribute properties is mentioned again in this chapter for each object described.

☞ For more information about the property table, see “On-line Help”, page 30.

The objects have links between them to maintain the **Time Navigator** functions. This is why it is necessary to present these objects in a functional order.

☞ To have global view of the relation between objects, refer to “Relations between objects”, page 123.

Platform related Objects

Host Object

The `Host` object represents a host on the network that is managed by **Time Navigator**. A host belongs to a unique group of platforms. One or more drives (`Drive`) and/or libraries (`Library`) can be attached. They can also be associated to alarms.

Attributes	Actions				Type
HostAlarm	- -	- -	- -	get	(handle list)
HostBackupMaster	- -	create	set	get	(handle)
HostBackupMastered	- -	- -	- -	get	(handle list)
HostComment		create	set	get	(string)
HostCommentUtf8		create	set	get	(string)
HostDiskSpace	- -	create	- -	get	(int)
HostDrive	- -	- -	- -	get	(string)
HostEnable	- -	create	set	get	(int)
HostHostGroup	- -	- -	set	get	(handle)
HostKey	- -	- -	set	get	(string)
HostLibrary	- -	- -	- -	get	(handle list)
HostName	open	create	set	get	(string)
HostProtocolNdmpPassWord	- -	create	set	- -	(string)
HostProtocolNdmpUser	- -	create	set	get	(string)
HostProtocolNdmpVersion	- -	create	set	get	(int)
HostProtocolNdmpVersionMaj	- -	create	set	get	(int)
HostProtocolNdmpVersionMin	- -	create	set	get	(int)
HostProtocolTinaVersionIndice	- -	create	set	get	(int)
HostProtocolTinaVersionMaj	- -	create	set	get	(int)
HostProtocolTinaVersionMin	- -	create	set	get	(int)
HostProtocolType	- -	create	set	get	(int)
HostReportUnavailability	- -	create	set	get	(int)
HostSecuredAgent	- -	create	set	get	(int)
HostServer	- -	- -	- -	get	(int)
HostSoftwareVersion	- -	- -	- -	get	(string)
HostStorageNode	- -	create	set	get	(int)
HostType	- -	create	set	get	(int)

Attributes

- `HostAlarm` — List of handles of alarms associated with host. If no alarm is attached to the host, the list is blank.
- `HostBackupMaster` — Specifies if the system has a backup master:
  -3 No backup master
  -4 The backup master is the server.

To determine if there is a backup master, perform the test with a handle containing an integer (-3 and -4):

```
assign HostBackupMaster &HBM
get MyHost
variable handle BCKMASTER_NON -3
variable handle BCKMASTER_SERV -4
if HBM == BCKMASTER_NON
if HBM == BCKMASTER_SERV
```

☞ For more information about the object operations, refer to the previous Chapters “tina_shell Language”, page 11 and “Manipulation of Objects”, page 29.

- `HostBackupMastered` — Handle list of hosts with masters.
- `HostComment` — Comment on host.
- `HostCommentUtf8` — Comment on host in UTF-8 format.
- `HostDiskSpace` — Theoretical capacity of disk (in MB) specified at host creation. This value has no relationship with actual disk capacity.
- `HostDrive` — List of handles of drives associated with host. If no drives are attached to the host, the list is blank.
- `HostEnable` — Host enabled/disabled:
  `0` Host disabled (default)
  `1` Host enabled
- `HostHostGroup` — Handles of host group to which the host belongs. If the attribute is absent on creation of the host, the host is created within a new host group.
- `HostKey` — Key of the server system (same as the key requested during `tina_init running`).

- `HostLibrary` — List of handles of libraries associated with host. If no library is attached to the host, the list is blank.
- `HostName` — Name of the host. Corresponds to the result of the command `hostname` or `uname -n` on this host.
- `HostProtocolNdmpPassword` — NDMP User password
  It is true only if `HostProtocolType` = 2.
- `HostProtocolNdmpUser` — NDMP User
  It is true only if `HostProtocolType` = 2.
- `HostProtocolNdmpVersion` — This attribute must no longer be used. It remains supported to ensure compatibility but is replaced by the `HostProtocolNdmpVersionMaj` and `HostProtocolNdmpVersionMin` attributes.
- `HostProtocolNdmpVersionMaj` — First digit of the NDMP version.
  It is true only if `HostProtocolType` = 2.
- `HostProtocolNdmpVersionMin` — Second digit of the TiNa protocol version.
  It is true only if `HostProtocolType` = 2.
- `HostProtocolTinaVersionIndice` — Third digit of the TiNa protocol version.
- `HostProtocolTinaVersionMaj` — First digit of the TiNa protocol version.
- `HostProtocolTinaVersionMin` — Second digit of the TiNa protocol version.
- `HostProtocolType` — Protocol type:
  1 TiNa (default)
  2 NDMP
  You can define several protocols by
  combining the values in a mask.
  (Ex: 3=1+2 equivalent to NDMP and TiNa).
- `HostReportUnavailability` — Specifies if an alarm must be logged when the host cannot be reached:
  `0` Alarm logged (default)
  `1` No alarm logged
- `HostSecuredAgent` — Specifies if an agent is security-compliant or not.
- `HostServer` — Server or client system:
  0 Client system (default)
  1 Server system1

- `HostSoftwareVersion` — Gets the **Time Navigator** version installed on the host. The format is `Maj.Min.Indice.Patch`.
- `HostStorageNode` — Specifies if the host is a storage node
  0 The host is not a storage node (default).
  1 The host is a storage node.
- `HostType` — Type of host. The type list of systems evolves frequently, consult the On-line Help by entering `help HostType` in the `tina_shell` window.

☞ For script examples concerning the `Host` object, refer to “Getting and Displaying Host Names”, page 126 and “Enabling Hosts, Applications and Drivers”, page 127.

Application Object

The `Application` object is an application controlled by **Time Navigator**. It belongs to a unique group of platforms (`HostGroup`) and must always be attached to a host (`Host`). It can also be associated with alarms.

Attributes	Actions				Type
ApplicationAlarm	- -	- -	- -	get	(handle list)
ApplicationCryptPassword	- -	create	set	get	(string)
ApplicationEnable	- -	create	set	get	(int)
ApplicationEngenioRootDir		create	set	get	(string)
ApplicationEngenioRootDirUtf8		create	set	get	(string)
ApplicationEnvironment	- -	create	set	get	(string list)
ApplicationFileList	- -	create *	- -	get	(string)
ApplicationHost	- -	create	set	get	(handle)
ApplicationHostGroup	- -	create	set	get	(handle)
ApplicationName	open	create	set	get	(string)
ApplicationNdmpMountPath		create	- -	get	(string)
ApplicationNdmpMountPath	- -	create *	- -	get	(string)
ApplicationNdmpOSCryptpassword	- -	create	set	get	(string)
ApplicationNdmpOsPassword	- -	create	set	- -	(string)
ApplicationNdmpOsUser	- -	create	set	get	(string)
ApplicationNdmpPassword		create	set	- -	(string)
ApplicationNdmpServer	- -	create *	- -	get	(string)
ApplicationNdmpUser		create	set	get	(string)
ApplicationNetDiskServer	- -	create	- -	get	(string)
ApplicationOwner	- -	create *	set	get	(string)
ApplicationOwnerUtf8	- -	create *	set	get	(string)
ApplicationOwnerCryptPassword	- -	create	set	get	(string)
ApplicationOwnerPassword	- -	create *	set	- -	(string)
ApplicationOwnerPasswordUtf8	- -	create *	set	- -	(string)
ApplicationParentApp	- -	create	- -	get	(handle)
ApplicationPassword	- -	create *	set	- -	(string)
ApplicationPasswordUtf8	- -	create *	set	- -	(string)
ApplicationReplicaDestApp	- -	- -	- -	get	(handle list)
ApplicationSnapHost		create	set	get	(handle)
ApplicationSnapNdmpFiler		create	set	get	(string)
ApplicationSnapNdmpTmpDir	- -	create	set	get	(string)
ApplicationSnapPassword		create	set	get	(string)
ApplicationSnapShot		create	set	get	(string)
ApplicationSnapType		create	set	get	(string)
ApplicationSnapTypeEngenio		create	set	get	(int)
ApplicationSnapUser		create	set	get	(string)
ApplicationSnapUser		create	set	- -	(string)
ApplicationSnapVersion		create	set	get	(int)

Attributes	Actions				Type
ApplicationType	- -	create	- -	get	(int)
ApplicationUserName	- -	create *	set	get	(string)
ApplicationUserNameUtf8	- -	create *	set	get	(string)
ApplicationVCB2OsCryptPassword	- -	create	set	get	(string)
ApplicationVCB2OsPassword	- -	create	set	- -	(string)
ApplicationVCB2OsPasswordUtf8	- -	create	set	- -	(string)
ApplicationVCB2Server	- -	create	set	get	(string)
ApplicationVCB2OsUser	- -	create	set	get	(string)
ApplicationVCB2OsUserUtf8	- -	create	set	get	(string)

* usage depends on the application type.
Refer to **Administration Console** to see which attributes are required for the creation of an application.

Attributes

`ApplicationName` — Name of the application.

`ApplicationAlarm` — List of handles of alarms associated with the application. If no alarms are attached to the application, the list is blank.

`ApplicationCryptPassword` — Application password is encrypted for security reasons during file retrieval and transmission.

`ApplicationEnable` — Application enabled/disabled.

0 Application disabled

1 Application enabled

`ApplicationEngenioRootDir` — Positions the mounting point

`ApplicationEngenioRootDirUtf8` — Positions the mounting point (path in UTF-8 format)

`ApplicationEnvironment` — List of environment variables written in the form "`name=value`".

`ApplicationFileList` — Path of the file containing the **FileList** application.

It is true only if the application is of the List type (`ApplicationType` = 6).

`ApplicationHost` — Handle of the host to which the application is attached.

`ApplicationHostGroup` — Handle of the platform group for the application. If this attribute is absent on creation of the application, the application is created within a new platform group.

ApplicationName	Name of the application.
ApplicationNdmpMountPath	NDMP mount path.
ApplicationNdmpMountPath	Path of the directory where volumes of the file server are assembled. It is true only if the application is of the NDMP type (ApplicationType = 11).
ApplicationNdmpOsCryptpassword	NDMP application user password encryption.
ApplicationNdmpOsPassword	NDMP application user password.
ApplicationNdmpOsUser	NDMP application user.
ApplicationNdmpPassword	NDMP user password.
ApplicationNdmpServer	Name of the server NDMP to back up. It is true only if the application is of the NDMP type (ApplicationType = 11)
ApplicationNdmpUser	NDMP user.
ApplicationNetDiskServer	Remote machine where the mapped drives to back up are located.
ApplicationOwner	Owner of the application.
ApplicationOwnerPassword	Password of the application owner.
ApplicationOwnerPasswordUtf8	Password of the application owner in Utf8.
ApplicationPassword	Password of privileged user.
ApplicationPasswordUtf8	Password of privileged user in Utf8.
ApplicationSnapHost	Application host.
ApplicationSnapNdmpFiler	Snapshot NDMP file server.
ApplicationSnapNdmpTmpDir	Completes the description of an NDMP snapshot for a temporary storage place.
ApplicationSnapPassword	Snapshot password.
ApplicationSnapShot	Application snapshot.
ApplicationSnapType	Application type.
ApplicationSnapTypeEngenio	Application snapshot type.
ApplicationSnapUser	Snapshot user.
ApplicationSnapVersion	Snapshot version.

`ApplicationName`	Name of the application.
`ApplicationType`	Type of application. The list of application types changes frequently, refer to the on-line help by entering `help ApplicationType` in the `tina_shell` window.
`ApplicationUserName`	Name of the privileged user.
`ApplicationUserNameUtf8`	Name of the privileged user in Utf8.
`ApplicationOwnerCryptPassword`	Application owner password is encrypted for security reasons during file retrieval and transmission.
`ApplicationParentApp`	Parent application name (see note).
`ApplicationReplicaDestApp`	Destination of replicated application.
`ApplicationVCB2OsCryptPassword`	vCenter encrypted user password.
`ApplicationVCB2OsPassword`	Password of the vCenter User.
`ApplicationVCB2OsPasswordUtf8`	Password of the vCenter User in Utf8.
`ApplicationVCB2Server`	Name of the machine on which the Virtual Center is installed.
`ApplicationVCB2OsUser`	Name of the user who has administration rights on the Virtual Center, or at least rights to back up, snapshot and restore.
`ApplicationVCB2OsUserUtf8`	Name of the user in Utf8 who has administration rights on the Virtual Center, or at least rights to back up, snapshot and restore.

Note When a parent application is replicated as a child application, the child application inherits certain parameters of the parent application. It is necessary to use `ApplicationParentApp` to specify the relationship between child and parent.

Note A comment field has been added for Applications and their Hosts. The field is accessible through `tina_adm` and the Hosts and Apps edit files, the API and `tina_shell` and can be seen with `tina_config`.

☞ For an example of a script concerning the `Application` object, refer to “Enabling Hosts, Applications and Drivers”, page 127.

HostGroup Object

The `HostGroup` object represents a group of hosts or applications. It is an operational set of hosts or applications with the same backup strategies and classes.
The `HostGroup` object is accessible indirectly by one of the platforms forming it. Simply retrieve the value of the `HostHostGroup` attribute from one of the platforms of the group. The value retrieved is the handle of the group.

Attributes	Actions				Type
HostGroupApplication	- -	- -	- -	get	(handle list)
HostGroupBackupClass	- -	- -	- -	get	(handle list)
HostGroupHost	- -	- -	- -	get	(handle list)
HostGroupStrategy	- -	- -	- -	get	(handle list)

Attributes

- `HostGroupApplication`
  List of handles of applications belonging to the host group. The list may contain just one application, or even be empty.
- `HostGroupBackupClass`
  List of handles of classes associated with the group of hosts or applications.
- `HostGroupHost`
  List of handles of hosts belonging to the group. The list may contain just one host, or even be empty.
- `HostGroupStrategy`
  List of strategy handles associated with the group of platforms. The list may contain between one and four backup strategies, or even none.

Device Related Objects

Drive Object

The `Drive` object represents a physical unit. There are three main associations of a drive:

- A library (`DriveAccessGroupLibrary` attribute): It is compulsory to specify the physical position of the drive in the library. It is possible to have a drive physically connected to a machine other than the machine that controls the library.
- One or more groups of users (attribute `DriveAccessGroup`), who can have access to manual operations, for example, copying, labelling, and reading.

- One of more cartridge pools (attribute `DriveAccessGroup`), that defines which cartridges can be loaded in the drive.

Attributes	Actions				Type
DriveAccessGroup	- -	- -	set	get	(handle list)
DriveAccessGroupLibrary	- -	- -	set	get	(handle)
DriveAlarm	- -	- -	- -	get	(handle list)
DriveCartridge	- -	- -	- -	get	(handle)
DriveConnectList	- -	create	set	get	(handle list)
DriveConnectType	- -	create	set	get	(int)
DriveEnable	- -	create	set	get	(int)
DriveFileSize	- -	create	set	get	(int)
DriveFull	- -	- -	set	get	(int)
DriveHost	open	create	- -	get	(handle)
DriveLastCleaning	- -	create	set	get	(int)
DriveLoader	- -	- -	- -	get	(int)
DriveLogicalIndexInLibrary	- -	create	set	get	(int)
DriveName	open	create	- -	get	(string)
DriveNbClean	- -	- -	- -	get	(int)
DriveNbLoad	- -	- -	- -	get	(int)
DriveNetwork	- -	create	- -	get	(handle)
DriveNextCleaning	- -	create	set	get	(int)
DrivePilot	- -	create	set	get	(string)
DriveSerialNumber	- -	create	set	get	(string)
DriveStatus	- -	create	set	get	(int)
DriveTapeLifeTime	- -	create	set	get	(int)
DriveTimeUsed	- -	- -	- -	get	(int)
DriveType	- -	create	- -	get	(int)
DriveUsrName	- -	create	set	get	(string)
DriveUsrPasswd	- -	create	set	- -	(string)
DriveVolumeRead	- -	- -	- -	get	(string)
DriveVolumeWritten	- -	- -	- -	get	(string)
DriveWorking	- -	- -	- -	get	(int)

Attributes

- `DriveAccessGroup` — List of handles of access groups associated with the drive.
- `DriveAccessGroupLibrary` — Handle of library type access group associated with the library controlling drive. This attribute is only valid if the loader is automated (`DriveLoader`= 2).
- `DriveAlarm` — List of handles of alarms associated with the drive.
- `DriveCartridge` — `"Null"` handle or handle of cartridge present in drive.

- `DriveConnectList` — List of handles of Host/Drive links associated with the drive.

- `DriveConnectType` — Drive connection type.
  1 Local (default: drive connected locally to a host).
  2 SAN (the drive is connected to a network).

  Note:
  Setting DriveConnectType to the value DriveConnectTypeSan requires setting the attribute DriveNetwork, to indicate what host the drive is connected to.

  To set DriveConnectType to the value DriveConnectTypeLocal, first make sure the drive is connected to only one host.

- `DriveEnable` — This attribute must no longer be used. It remains supported to ensure compatibility but is replaced by the DriveStatus attribute.

- `DriveFileSize` — File size in MB (from 256 MB to 200 GB) if the drive is a Disk Drive.

- `DriveFull` — Drive holding a cartridge:
  0 Drive empty
  1 Drive full
  To empty a drive, set the attribute value to 0.
  You cannot set the attribute value to 1.

- `DriveHost` — Handle of host the drive is attached to.
  Can only be used if the drive is local (`DriveConnectType=1`).
  Must not be used with the `DriveConnectList` and `DriveNetwork` attributes.

- `DriveLastCleaning` — Time since last cleaning (in hours).

- `DriveLoader` — Cartridge loader:
  1 Manual loader
  2 Automated loader

- `DriveLogicalIndexInLibrary` — Logical index in library, default is 0.

- `DriveName` — Name of drive.

- `DriveNbClean` — Number of times the drive has been cleaned.
- `DriveNbLoad` — Number of times a cartridge has been mounted in the drive.
- `DriveNetwork` — Handle of the network to which the drive is connected. Can only be used if the drive is connected to a network (`DriveConnectType=2 or 3`) Must not be used with the `DriveHost` attribute.
- `DriveNextCleaning` — Time till next cleaning (in hours).
- `DrivePilot` — Drive device descriptor.

WARNING This attribute must not be used with the `DriveConnectList` attribute.

Note The syntax depends on the drive type:
Disk Drive
Path of the directory where the Disk Drive cartridges are located.
Drive
UNIX: path of special file associated with drive. Windows: notation "c?b?t?l?".

- `DriveSerialNumber` — Drive serial number.
  To reset the drive serial number, set the attribute `DriveSerialNumber` to `NULL_STRING`. You cannot set this attribute to any other value.
- `DriveStatus` — Status of the drive.
  0 Disabled.
  1 Enabled.
  2 Maintenance.
- `DriveTapeLifeTime` — Lifetime of cartridge (in points).
- `DriveTimeUsed` — Number of seconds the drive has been used for.
- `DriveType` — Type of drive. The list of drive types change frequently, refer to the on-line help by entering `help DriveType` in the `tina_shell` window.

- `DriveUsrName` — Name of a user with access rights for a Disk Drive defined on a network drive.
  Only useful if the drive is a Disk Drive.
- `DriveUsrPasswd` — Password of the user with access rights for a Disk Drive defined on a network drive.
  Only useful if the drive is a Disk Drive.
- `DriveVolumeRead` — Amount of information that the drive has read.
- `DriveVolumeWritten` — Amount of information that the drive has written .
- `DriveWorking` — Drive activity:
  1 Reading
  2 Writing
  3 Rewinding
  4 Skip forward
  0 No activity
  -1 Exception

☞ For an example of a script concerning the `Drive` object, refer to “Enabling Hosts, Applications and Drivers”, page 127.

DriveConnection Object

The `DriveConnection` object represents the connections between a drive and one or several hosts. This object is not added in the catalog. It is actually represented by the associated drive using the `DriveConnectList` drive attribute. It is closely related to the `Drive` object: its own attributes constitute complementary drive attributes.

To modify one of the attributes, you must first create a new `DriveConnection` object, which has no incidence on the catalog. Then, you must modify the `DriveConnectList` drive attribute by creating a new list that consists of the previous connections that do not change and those you want to modify.

Attributes	Actions				Type
DriveConnectionDevice	- -	create	- -	get	(string)
DriveConnectionDrive	- -	- -	- -	get	(handle)
DriveConnectionHost	- -	create	- -	get	(handle)
DriveConnectionProtocol	- -	create	- -	get	(int)
DriveConnectionProtocolNdmMaj	- -	create	- -	get	(int)
DriveConnectionProtocolNdmMin	- -	create	- -	get	(int)
DriveConnectionProtocolNdmNetAddr	- -	create	- -	get	(string)
DriveConnectionProtocolNdmPasswd	- -	create	- -	- -	(string)
DriveConnectionProtocolNdmUser	- -	create	- -	get	(string)
DriveConnectionProtocolSymMaj	- -	create	- -	get	(int)
DriveConnectionProtocolSymMin	- -	create	- -	get	(int)
DriveConnectionProtocolTnaIndice	- -	create	- -	get	(int)
DriveConnectionProtocolTnaMaj	- -	create	- -	get	(int)
DriveConnectionProtocolTnaMin	- -	create	- -	get	(int)
DriveConnectionStatus	- -	- -	set	get	(int)

Attributes

- `DriveConnectionDevice` Drive device descriptor.

Note The syntax of the device descriptor depends on the type of drive and the operating system:
Unix: Special file path associated with the drive.
Windows: notation "c?b?t?l?".

- `DriveConnectionDrive` Handle of the drive.

- `DriveConnectionHost` Handle of the host the drive is connected to.
- `DriveConnectionProtocol` Protocol used for connection:
  1TiNa (default)
  4NDMP
- `DriveConnectionProtocolNdmMaj` First digit of the NDMP protocol version.
- `DriveConnectionProtocolNdmMin` Second digit of the NDMP protocol version.
- `DriveConnectionProtocolNdmNetAddr` Network name of the drive.
- `DriveConnectionProtocolNdmPasswd` NDMP User password.
  It is true only if HostProtocolType = 2.
- `DriveConnectionProtocolNdmUser` NDMP User.
  It is true only if HostProtocolType = 2.
- `DriveConnectionProtocolSymMaj` Value of the major version of the Sym API protocol used by Time Navigator.
- `DriveConnectionProtocolSymMin` Value of the minor version of the Sym API protocol used by Time Navigator.
- `DriveConnectionProtocolTnaIndice` Value of the indice of the version of the TiNa protocol used by Time Navigator.
- `DriveConnectionProtocolTnaMaj` Value of the major version of the TiNa protocol used by Time Navigator.
- `DriveConnectionProtocolTnaMin` Value of the minor version of the TiNa protocol used by Time Navigator.
- `DriveConnectionStatus` Status of the drive-host connection:
  1: the drive-host connection is enabled
  2: the drive-host connection is disabled

Network Object

The Network object represents the networks that the drives can be attached to.

Attributes	Actions				Type
NetworkDrive	- -	- -	- -	get	(handle list)
NetworkName	open	create	- -	get	(string)
NetworkType	- -	create	- -	get	(int)

Attributes

- `NetworkDrive` — List of handles of drives attached to the network.
- `NetworkName` — Name of the network.
- `NetworkType` — Type of network:
  1 SAN
  2 `LAN` (not yet implemented)

Library Object

The `Library` object is associated with a system (`Host`), alarms (`Alarm`), and an access group (`AccessGroup`).

Warning When deleting a `Library` object, delete the `AccessGroup` object it is associated to.

Attributes	Actions				Type
LibraryAccessGroup	- -	create	- -	get	(handle)
LibraryAlarm	- -	- -	- -	get	(handle list)
LibraryCartridge	- -	- -	- -	get	(handle list)
LibraryCartridgeInMBox	- -	- -	- -	get	(handle list)
LibraryCleaningNb	- -	create	set	get	(int)
LibraryCleaningNbMax	- -	create	set	get	(int)
LibraryCleaningTape	- -	create	set	get	(int)
LibraryCleaningTapeSlot	- -	create	set	get	(int)
LibraryGeoNbCol	- -	create	set	get	(int)
LibraryGeoSlot0Position	- -	create	set	get	(int)
LibraryGeoSlot1Position	- -	create	set	get	(int)
LibraryHost	open	create	- -	get	(handle)
LibraryLocations	- -	- -	- -	get	(handle list)
LibraryMailbox	- -	- -	- -	get	(int)
LibraryPilot	open	create	set	get	(string)
LibraryShared	- -	create	set	get	(int)
LibrarySupportBarCode	- -	create	set	get	(int)
LibraryType	- -	create	- -	get	(int)

Attributes

- `LibraryAccessGroup` — Handle of type Library access group associated with the library.
- `LibraryAlarm` — List of handles of alarms associated with the library.
- `LibraryCartridge` — List of cartridges.
- `LibraryCartridgeInMBox` — List of the cartridges known by the catalog and located in the mailboxes of the library.
- `LibraryCleaningNb` — Number of cleaning operations performed.

- `LibraryCleaningNbMax` — Maximum number of cleaning operations.
- `LibraryCleaningTape` — Library with a cleaning cartridge:
  0 No
  1 Yes
- `LibraryCleaningTapeSlot` — Number of the cleaning cartridge slot.
- `LibraryGeoNbCol` — Number of columns of the library.
- `LibraryGeoSlot0Position` — Slot 0 display
  1: Slot 0 appears in top left-hand corner
  2: Slot 0 appears in top right-hand corner
  3: Slot 0 appears in bottom left-hand corner
  4 : Slot 0 appears in bottom right-hand corner
- `LibraryGeoSlot1Position` — Slot 1 display
  1: Slot 1 is in the same column as Slot 0
  2: Slot 1 is on the same line as Slot 0
- `LibraryHost` — Handle of host to which library is attached. This is mandatory to open a library.
- `LibraryLocations` — List of the mailbox locations (appears as a list of handles of object LibraryLocation).
- `LibraryMailbox` — Indicates whether there is a mailbox in the library.
  0: No mailbox.
  1: There is a mailbox.
- `LibraryPilot` — Library device descriptor.

Note The syntax depends on the unit type and the operating system:
UNIX: Special file path associated with the drive,
Windows: notation "c?b?t?l?".

- `LibraryShared` — Specifies if the library is shared.
  0: the library is shared
  1: the library is not shared
- `LibrarySupportBarCode` — Library supporting bar codes:
  0: No
  1: Yes

- `LibraryType` — Type of library. The list of library types change frequently, refer to on-line help by entering `help LibraryType` in the `tina_shell` window.

Note In **Administration Console**, you must define a library type, a name, a device descriptor and a list of accessible drives. In `tina_shell`, however, the definition of the name and list of drives are done via an access group (`AccessGroup`) of library type.

LibraryLocation object

The LibraryLocation object represents the location of a library.

Use the LibraryLocations attribute of the Library object to access the LibraryLocation object.

Attributes	Actions				Type
LibraryLocationAccessible	- -	- -	- -	get	(int)
LibraryLocationBarCode	- -	- -	- -	get	(string)
LibraryLocationCartridge	- -	- -	- -	get	(handle)
LibraryLocationCleaning	- -	- -	- -	get	(int)
LibraryLocationEnable	- -	- -	set	get	(int)
LibraryLocationLock	- -	- -	set	get	(int)
LibraryLocationName	- -	- -	- -	get	(string)
LibraryLocationType	- -	- -	- -	get	(int)

Attributes

- LibraryLocationAccessible

  Status of the location
  1: the location is accessible
  0: the location is not accessible

- LibraryLocationBarCode

  Barcode of the cartridge (if the location is full).

- LibraryLocationCartridge

  Handle of the cartridge (if the location is full and if the cartridge is known by the catalog).

- LibraryLocationCleaning

  1: the location is a cleaning slot
  0: the location is not a cleaning slot

- LibraryLocationEnable

  1: the location is enabled
  0: the location is disabled

- LibraryLocationLock

  1: the location is locked
  0: the location is not locked

- LibraryLocationName

  Name of the location.

- LibraryLocationType

  1: slot
  2: drive
  3: mailbox
  4: picker
  0: other type of location

AccessGroup Object of Library type

The `AccessGroup` object represents a group allowing the association of users with drives.

A group can be one of the following types:

- **Library: The library type group associates a Library with one or more drives.**
- Cartridge pool: The cartridge pool associates a user with one or more drives. In the latter case, it is possible to define a retention period.
- User: The user group associates one or more users with one or more drives.

Attributes	Actions				Type
AccessGroupAlarm	- -	- -	- -	get	(handle list)
AccessGroupDrive	- -	- -	set	get	(handle list)
AccessGroupLibrary	- -	- -	- -	get	(handle)
AccessGroupLibrarySerialNumber	- -	create	- -	get	(string)
AccessGroupName	open	create	- -	get	(string)
AccessGroupType	- -	create	- -	get	(int)

Attributes

Only the attributes of the `AccessGroup` object Library type are listed below:

- `AccessGroupAlarm` — List of handles of alarms associated with access group.
- `AccessGroupDrive` — List of handles of drives associated with access group.
- `AccessGroupLibrary` — Handle of library associated with access group. It is true only if the access group is of library type `AccessGroupType = 1)`.
- `AccessGroupLibrarySerialNumber` — Library serial number.
- `AccessGroupName` — Name of access group (Name of Library).
- `AccessGroupType` — Type of access group:
  - **1** **Library type**
  - 2 User type
  - 3 Pool cartridge type

Cartridge Pool Related Objects

AccessGroup Object of Cartridge Pool Type

The `AccessGroup` object represents a group allowing the association of users with drives.

A group can be of the following type:

- Library: The library type group associates a library with one or more drives.
- User: A group of users associates one or several users with one or more drives.
- **Cartridge pool: The cartridge pool associates a user with one or more drives.**

Attributes	Actions				Type
AccessGroupAlarm	- -	- -	- -	get	(handle list)
AccessGroupComment	- -	create	set	get	(string)
AccessGroupCommentpPropagate	- -	- -	set	- -	(int)
AccessGroupCommentUtf8	- -	create	set	get	(string)
AccessGroupDrive	- -	- -	set	get	(handle list)
AccessGroupName	open	create	- -	get	(string)
AccessGroupPolicy	- -	create	set	get	(int)
AccessGroupRetUnit	- -	create	set	get	(int)
AccessGroupRetValue	- -	create	set	get	(int)
AccessGroupType	- -	create	- -	get	(int)
AccessGroupUser	- -	- -	- -	get	(handle list)

Attributes

Only the attributes of the `AccessGroup` object cartridge pool type are listed below:

- `AccessGroupAlarm` — List of handles of alarms associated with access group.
- `AccessGroupComment` — ASCII comment stored in the catalog that allows identification of a cartridge pool.
- `AccessGroupCommentPropagate` — To be used if the AccessGroupComment or AccessGroupCommentUtf8 is set. It allows to assign the comment to the cartridges that already exist.
  0 No
  1 Yes
- `AccessGroupCommentUtf8` — UTF8 comment stored in the catalog that allows identification of a cartridge pool.

- `AccessGroupDrive` — List of handles of drives associated with access group.
- `AccessGroupName` — Name of access group (name of the cartridge pool).
- `AccessGroupPolicy` — Management policy for the cartridge pool. It is true only if the access group is of the cartridge pool type (`AccessGroupType = 3`):
  1 Infinite pool
  2 Cyclical pool
- `AccessGroupRetUnit` — Retention unit of a cartridge. It is true only if the access group is of the cartridge pool type (`AccessGroupType = 3`) and the management of the pool cycle (`AccessGroupPolicy =2`):
  1 Day
  2 Week
  3 Month
  4 Year
- `AccessGroupRetValue` — Retention period [0-99] of a cartridge. It is true only if the access group is the cartridge pool type (`AccessGroupType = 3`) and the management of the pool cycle (`AccessGroupPolicy =2`).
- `AccessGroupType` — Type of access group:
  1 Library type
  2 User type
  3 Cartridge pool type
- `AccessGroupUser` — List of handles of users associated with the user group. The list contains a single element in the case of a pool of cartridges (`AccessGroupType = 3`). It is true only if the access group is a user type or a cartridge pool (`AccessGroupType` = 2 or 3).

Note Total definition of a cartridge pool must be via association with a user. The name of this user is used for the prefix of labels and the list of cartridges present in the pool, via a list of handles.

☞ For a script example implementing an AccessGroup object of a cartridge pool type, refer to “Getting a Cartridge List via a Cartridge Pool”, page 134.

User Object of Cartridge Pool Type

The `User` object represents a user that can be associated with:

- **An access group** (`AccessGroup`) **of cartridge pools**: the user represents the label of cartridges (the prefix set in the definition of a pool of cartridges). The user has a list of cartridges belonging to him/her, this is therefore the list of cartridges in the pool.
- An access group (`AccessGroup`) of the User type: represents the users of the operating system.

Attributes	Actions				Type
UserAccessGroup	- -	create	set	get	(handle)
UserAlarm	- -	- -	- -	get	(handle list)
UserCartridge	- -	- -	- -	get	(handle list)
UserName	open	create	- -	get	(string)

Attributes

- `UserAccessGroup` — Handle of the access group associated with the user. Here access group type pool of cartridges. The handle is Null if no access group is attached.
- `UserAlarm` — List of handles of alarms associated with the user.
- `UserCartridge` — List of handles of cartridge(s) associated with the user.
- `UserName` — Label Prefix for the pool of cartridges. The syntax corresponds to UNIX .

Warning Do not confuse users attached to user groups of type "cartridge pool" with users attached to user groups of type "user".

User Related Objects

AccessGroup Object of User Type

The `AccessGroup` object represents a group that allows users to be associated with drives.

A group can be of the following types:

- Library: The Library type group associates a library with one or more drives.
- **Users: A user group associates one or several users with one or more drives.**
- Cartridge Pool: Cartridge pool associates a user with one or more drives.

Note The users represent the users of the operating system. They cannot be created in `tina_shell`. The creation of the users can only be done from **Administration Console**.

Note Cartridges associated with the user, in this context, correspond to local archiving cartridges.

Attributes	Actions				Type
AccessGroupAlarm	- -	- -	- -	get	(handle list)
AccessGroupDrive	- -	- -	set	get	(handle list)
AccessGroupName	open	create	- -	get	(string)
AccessGroupType	- -	create	- -	get	(int)
AccessGroupUser	- -	- -	- -	get	(handle list)

Attributes

- `AccessGroupAlarm`
  List of handles of alarms associated with access group.

- `AccessGroupDrive`
  List of handles of drives associated with access group.

- `AccessGroupName` Name of access group (Name of user group).

- `AccessGroupType`
  Type of access group:

1	Library type
2	**User type**
3	Cartridge pool type.

- `AccessGroupUser`
  List of handles of users associated with the user group. It is true only if the access group is a user type or a cartridge pool (`AccessGroupType` = 2 or 3).

User Object of User Type

The `User` object represents a user that can be of the following types:

- An access group (`accessgroup`) of cartridge pool type: in this case, it is the cartridge label (a set prefix in the definition of the cartridge pool).
- **An access group** `(accessgroup)` **of User type**: in this case, it is the operating system users.

Attributes	Actions				Type
UserAccess	- -	- -	set	get	(int)
UserAccessArchiving	- -	- -	set	get	(int)
UserAccessBackup	- -	- -	set	get	(int)
UserAccessCartridge	- -	- -	set	get	(int)
UserAccessGeneral	- -	- -	set	get	(int)
UserAccessGroup	- -	create	set	get	(handle)
UserAccessOther	- -	- -	set	get	(int)
UserAlarm	- -	- -	- -	get	(handle list)
UserCartridge	- -	- -	- -	get	(handle list)
UserGid	- -	- -	- -	get	(int)
UserIsPrivileged	- -	- -	- -	get	(int)
UserName	open	create	- -	get	(string)
UserPassword	- -	create	set	- -	(string)
UserUid	- -	- -	- -	get	(int)
UserAccessRights	- -	- -	set	get	(int list)

Attributes

- `UserAccessGroup` — Handle of access group associated with the user. Here a User type Access group. The Handle is Null if no access group is attached.
- `UserAccessRights` — List of all the existing access rights in **Time Navigator** indicating whether each right is enabled or not for the user.
- `UserAlarm` — List of handles of alarms associated with a user.
- `UserCartridge` — List of handles of cartridge(s) associated with a user.
- `UserGid` — Gid (Identifier of the Group).
- `UserIsPrivileged` — Indicates if the user is the privileged user or not.
- `UserName` — User name.
- `UserPassword` — User password.
  Can only be modified by its own user or by another user who has the User group Management rights.

- `UserUid` Uid (Identifier of the User).

The following attributes concern the access rights and correspond to the Control window of **Administration Console**. Before being used in `tina_shell`, these rights must be defined in **Administration Console**. You can then recover their values in `tina_shell` using the command `get`, and reassign these using the command `set` during the creation of a new configuration in **Time Navigator**.

The values of these attributes are not directly readable, but their handling can be useful such as in the case where you want to automate the creation of the same configuration **Time Navigator** on several servers using a `tina_shell` script.

- `UserAccess`
- `UserAccessArchiving`
- `UserAccessBackup`
- `UserAccessCartridge`
- `UserAccessGeneral`
- `UserAccessOther`

Warning Do not confuse the type `User` associated to a group of type `Cartridge pool` and those associated to a group of type `User`.

Note The users represent the users of the operating system. They cannot be created in `tina_shell`. The creation of the users can only be done from **Time Navigator**.

Data Related Objects

Catalog Object

The `catalog` object represents the heart of the administration catalog. The information obtained are the cells that are displayed in **Administration Console**, the composition of the cache, free space (MB) and details of the cache (Disk / memory).

Attributes	Actions				Type
CatalogCacheVolumeRead	- -	- -	- -	get	(int)

Attributes	Actions				Type
CatalogCacheVolumeWrite	- -	- -	- -	get	(int)
CatalogCartLabelPrefix	- -	- -	set	get	(string)
CatalogCurrentUserName	- -	- -	- -	get	(string)
CatalogCurrentUserNameUtf8	- -	- -	- -	get	(string)
CatalogDefaultUsrAccessrRghts	- -	- -	set	get	(int list)
CatalogDiskCacheVolumeFree	- -	- -	- -	get	(int)
CatalogDiskCacheVolumeTotal	- -	- -	- -	get	(int)
CatalogInstance	- -	- -	- -	get	(int)
CatalogMaxOnDemandBackup	- -	- -	set	get	(int)
CatalogMemoryCacheVolumeFree	- -	- -	- -	get	(int)
CatalogMemoryCacheVolumeTotal	- -	- -	- -	get	(int)
CatalogName	open	- -	- -	get	(string)
CatalogObject	- -	- -	- -	get	(int)
CatalogVolumeFree	- -	- -	- -	get	(int)
CatalogVolumeTotal	- -	- -	- -	get	(int)

Attributes

- CatalogCacheVolumeRead — Size of the cache used for reading data, in MB.
- CatalogCacheVolumeWrite — Size of the cache used for writing data, in MB.
- CatalogCartLabelPrefix — String that prefixes all cartridge labels for this catalog.
- CatalogCurrentUserName — Name of the current user of the catalog.
- CatalogCurrentUserNameUtf8 — Name of the current user of the catalog in Utf8.
- CatalogDefaultUsrAccessRights — List of all the existing access rights in **Time Navigator** indicating whether each right is enabled or not for the user.
- CatalogDiskCacheVolumeFree — Free disk cache, in MB.
- CatalogDiskCacheVolumeTotal — Total disk cache, in MB.
- CatalogInstance — Number of instances.
- CatalogMaxOnDemandBackup — Maximum number of simultaneous on demand backups.
- CatalogMemoryCacheVolumeFree — Free volume of memory cache assigned to the catalog.
- CatalogMemoryCacheVolumeTotal — Total volume of memory cache assigned to the catalog.

■ `CatalogName`	Catalog name. The `tina_shell` language requires at least one argument on opening. Whatever the choice of `CatalogName`, it is always the catalog to be administered that is opened (interrogation of the `CatalogName` gives the name of this catalog).
■ `CatalogObject`	Number of objects.
■ `CatalogDiskCacheVolumeFree`	Free disk cache, in MB.
■ `CatalogVolumeTotal`	Catalog volume, in MB.

☞ For a script example concerning a `Catalog` object, refer to “Getting Catalog Information”, page 131.

Cartridge Object

The `Cartridge` object represents a labelled cartridge. A cartridge is associated with the owner user who is associated to an access group (`AccessGroup`). If the cartridge is mounted, it is associated only to one drive. The cartridge type corresponds to the type of drive used to write its label. It is also associated with alarms.

Note The creation of a cartridge object corresponds to **Cartridge-Label-Write** menu in **Administration Console**.

Attributes	Actions				Type
CartridgeAlarm	- -	- -	- -	get	(handle list)
CartridgeBarCode	- -	- -	- -	get	(string)
CartridgeCloseStatus	- -	- -	set	get	(int)
CartridgeComment	- -	create	set	get	(string)
CartridgeCommentUtf8	- -	create	set	get	(string)
CartridgeContainer	- -	- -	- -	get	(handle)
CartridgeDateCreate	- -	- -	- -	get	(int)
CartridgeDateLastBck	- -	- -	- -	get	(int)
CartridgeDateReused	- -	- -	- -	get	(int)
CartridgeDescription	- -	- -	- -	get	(string)
CartridgeDrive	- -	create	- -	get	(handle)
CartridgeFileNb	- -	- -	- -	get	(int)
CartridgeFillStatus	- -	- -	- -	get	(int)
CartridgeFormat	- -	- -	- -	get	(int)
CartridgeLocation	- -	- -	set	- -	(int)
CartridgeLocked	- -	- -	- -	get	(boolean)
CartridgeName	open	- -	- -	get	(string)
CartridgeNbLoad	- -	- -	- -	get	(int)
CartridgeNbRecycle	- -	- -	- -	get	(int)
CartridgeNbTapeFile	- -	- -	- -	get	(int)
CartridgeNumber	- -	create	- -	get	(int)
CartridgeOperationMask	- -	- -	- -	get	(int)
CartridgePathLocation	- -	- -	set	get	(string)
CartridgeRetUnit	- -	- -	- -	get	(int)
CartridgeRetValue	- -	- -	set	get	(int)
CartridgeStatus	- -	- -	set	get	(int)
CartridgeTimeUsed	- -	- -	- -	get	(int)
CartridgeType	- -	- -	- -	get	(int)
CartridgeUseNb	- -	- -	- -	get	(int)
CartridgeUseNbMax	- -	- -	- -	get	(int)
CartridgeUser	- -	create	- -	get	(handle)
CartridgeVolume	- -	- -	- -	get	(int)
CartridgeVolumeRead	- -	- -	- -	get	(string)

Attributes	Actions				Type
CartridgeVolumeWritten	- -	- -	- -	get	(string)

Attributes

- `CartridgeAlarm` — List of handles of alarms associated with cartridge.
- `CartridgeBarCode` — Bar code associated with cartridge or EmptyString if no associated bar code.
- `CartridgeCloseStatus` — Status of the cartridge (open, closed, reopened).
- `CartridgeComment` — ASCII comment stored in the catalog that allows to identify a cartridge. If no comment is specified, the cartridge pool comment is used.
- `CartridgeCommentUtf8` — UTF8 comment stored in the catalog that allows to identify a cartridge. If no comment is specified, the cartridge pool comment is used.
- `CartridgeContainer` — Handle of the object containing the cartridge.
- `CartridgeDateCreate` — Date of cartridge creation.
- `CartridgeDateLastBck` — Date of last backup.
- `CartridgeDateReused` — Date of last recycling of cartridge or date of its creation if no recycling .
- `CartridgeDescription` — Descriptive character string of a cartridge. By default, name of the current catalogue.
- `CartridgeDrive` — Handle of last drive used for cartridge or drive selected for labelling.
- `CartridgeFileNb` — Number of tape files on cartridge (including label).
- `CartridgeFillStatus` — Filling level of the cartridge.
- `CartridgeFormat` — Cartridge format:
  1 Tar format
  2 Cpio format
  3 TiNa format
  5 Fastrax format
  6 Sidf format
  7 Unknown format

- `CartridgeLocation` Putting a cartridge off-line
  1 Cartridge in mailbox or move cartridge to mailbox (set/get)
  2 Cartridge in the library (get)
  3 Cartridge in the drive (get)
  4 Cartridge outside the library/drive (get)

- `CartridgeLocked` Indicates whether the cartridge is locked by another process. A test is performed to ensure the validity of the lock.

- `CartridgeName` Cartridge name, syntax: <Label prefix><Name of owner user><Cartridge number>.

- `CartridgeNbLoad` Number of times the cartridge was mounted in a drive.

- `CartridgeNbRecycle` Number of times the cartridge has been recycled.

- `CartridgeNbTapeFile` Number of files on tape drives (acting as cartridges).

- `CartridgeNumber` Cartridge number.

- `CartridgeOperationMask` Indicates the operations that can be performed on the cartridge. The value is a bit mask that must be decoded.
  1 Duplicate
  2 Recycle
  4 Close
  8 Reopen
  16 Delete
  32 ToSpare
  64 SafeRecycle
  128 SafeToSpare
  256 SafeDelete

- `CartridgePathLocation` Cartridge location.

- `CartridgeRetUnit` Indicates the cartridge whose retention you are seeking. Retention is no longer based on the pool but on individual cartridges.
  If a security rule has been attached to a cartridge, the retention of this rule overrides the pool retention.

- `CartridgeRetValue` Indicates the value of the cartridge retention.

- `CartridgeStatus` Cartridge status:
  1 Cartridge empty
  2 Cartridge partly full
  3 Cartridge closed, full
  4 Cartridge closed on incident
  5 Cartridge closed at initialization
  6 Cartridge closed manually
  7 Cartridge to be reopened (temporary state)
  8 Cartridge cleared
- `CartridgeTimeUsed` Number of hours the cartridge has been used for.
- `CartridgeType` Type of cartridge, identical to the type of unit which created it.
- `CartridgeUseNb` Number of uses of cartridge.
- `CartridgeUseNbMax` Maximum number of uses of cartridge.
- `CartridgeUser` Handle of user owner. User or pool of cartridge type AccessGroup.
- `CartridgeVolume` Space occupied on cartridge (in MB).
- `CartridgeVolumeRead` Amount of data that has been read from the cartridge.
- `CartridgeVolumeWritten` Amount of data that has been written on the cartridge.

☞ For a script example concerning a `Cartridge` object, refer to “Getting a Cartridge List via a Cartridge Pool”, page 134.

Job Object

The `Job` object represents a data read and/or write operation. It can be active (observed in real time), or finished and sent to the history.

Note This object cannot be created since it only gathers information.

Attributes	Actions				Type
JobAlarms	- -	- -	- -	get	(handle list)
JobAlarmSeverity	- -	- -	- -	get	(int)
JobCacheTotal	- -	- -	- -	get	(int)
JobCacheUsed	- -	- -	- -	get	(int)
JobCurrentObject	- -	- -	- -	get	(string)
JobDateCreate	- -	- -	- -	get	(int)
JobDateEnd	- -	- -	- -	get	(int)
JobDateRun	- -	- -	- -	get	(int)
JobDateSubmit	- -	- -	- -	get	(int)
JobExecutions	- -	- -	- -	get	(int)
JobExecutionsInterval	- -	- -	- -	get	(int)
JobExecutionsMax	- -	- -	- -	get	(int)
JobExpectedVolume	- -	- -	- -	get	(string)
JobFolder	- -	- -	- -	get	(string)
JobFormatRead	- -	- -	- -	get	(int)
JobFormatWrite	- -	- -	- -	get	(int)
JobHost	- -	- -	- -	get	(handle)
JobId	open	- -	- -	get	(int)
JobMode	- -	- -	- -	get	(int)
JobOperationMask	- -	- -	- -	get	(int)
JobParallelismRead	- -	- -	- -	get	(int)
JobParallelismWrite	- -	- -	- -	get	(int)
JobPlatformClass	- -	- -	- -	get	(int)
JobPlatformHandle	- -	- -	- -	get	(handle)
JobPoolNameRead	- -	- -	- -	get	(string list)
JobPoolNameWrite	- -	- -	- -	get	(string list)
JobProcessedObjects	- -	- -	- -	get	(int)
JobProcessedVolume	- -	- -	- -	get	(string)
JobPropertyMask	- -	- -	- -	get	(int)
JobRank	- -	- -	set	- -	(int)
JobStatus	- -	- -	set	get	(int)
JobStrategyName	- -	- -	- -	get	(int)
JobType	- -	- -	- -	get	(int)
JobUser	- -	- -	- -	get	(handle)

Attributes

- `JobAlarmSeverity` Severity of the alarm associated to the job.
  0 No severity
  1 Minor severity
  2 Major severity
  3 Critical severity
- `JobCacheTotal` Total number of cache blocks.
- `JobCacheUsed` Number of cache blocks used.
- `JobCurrentObject` Name of object currently being processed.
- `JobDateCreate` Creation date
  - 3 Incorrect date (if the job was not created).
- `JobDateEnd` End date
  -3 Incorrect date (if job was not finished).
- `JobDateRun` Start date
  - 3 Incorrect date (if the job was not started).
- `JobDateSubmit` Submission date
  -3 Incorrect date (if the job was not submitted).
- `JobExecutions` Number of job executions.
- `JobExecutionsInterval` Interval between two executions.
- `JobExecutionsMax` Maximum number of executions.
- `JobExpectedVolume` Expected volume, in the form of string with thousands separator.
- `JobFolder` Name of source or destination folder (`Job` type backup, synthetic backup, archive, restore, duplication, export source and export target)
  -1 Invalid value.
- `JobFormatRead` Read format (job type duplication):
  1 TiNa format
  2 tar format
  3 Cpio format
  4 Fastrax format
  5 Sidf format
  6 Unknown
  -1 Invalid value

- `JobFormatWrite` — Write format (`Job` type backup, synthetic backup, archive, duplication, export source and export target):
  1 Format TiNa
  2 Format tar
  3 Format cpio
  4 Fastrax format
  5 Sidf format
  6 Unknown
  -1 Invalid value
- `JobHost` — Handle of host executing job or value returned: Invalid handle.
- `JobId` — Unique job identification number.
- `JobMode` — Backup mode (`Job` type backup, synthetic backup, export source and export target):
  1 Full mode for job
  2 Incremental mode for job
  -1 Invalid value
- `JobOperationMask` — Indicates the operations that can be performed on the job. The value is a bit mask that must be decoded.
  1 Abort
  2 Suspend
  4 Restart
  8 Rank
- `JobParallelismRead` — Degree of parallelism on read (`Job` type backup, synthetic backup, archive, restore, duplication, export source and export target):
  -5 Total parallelism
  The values 2 to 4 correspond to the degree of parallelism
  -1 Invalid value
- `JobParallelismWrite` — Degree of parallelism on write (`Job` type backup, synthetic backup, export source and export target):
  -5 Total parallelism
  The values from 2 to 4 correspondent to the degree of parallelism
  -1 Invalid value.
- `JobPlatformClass` — Class of backed up platform (`Job` type backup, synthetic backup, export source, export target, catalog maintenance):
  1 Host platform class
  2 Application platform class
- `JobPlatformHandle` — Handle of backed up platform (`Job` type backup, synthetic `backup`, export source and export target).
- `JobPoolNameRead` — Name of cartridge pools corresponding to drive sessions.

- `JobPoolNameWrite` Name of cartridge pools corresponding to write sessions.
- `JobProcessedObjects` Number of processed jobs.
- `JobProcessedVolume` Processed volume, in the form of string with thousands separator.
- `JobPropertyMask` Job property:
  PropertyMaskOnDemandBackup: On-demand backup job
  PropertyMaskReplication: Replication job
  PropertyMaskSnapshot: Snapshot backup
- `JobRank` Change job priority ranking:
  1 Go to higher ranking
  2 Go to lower ranking
  3 Go to first ranking
  4 Go to last ranking
- `JobStatus` Job status, retrievable by `Get` function:
  1 Sessions not created
  2 Sessions created and ready
  3 Suspended on request
  4 Suspended automatically
  5 Running
  6 Suspending on request
  7 Returning to Ready
  8 Suspending automatically
  9 Stopping after `tina_stop`
  10 Stopping on error
  11 Stopping on abort request
  12 Stopping on suspend request
  13 Terminated after tina_stop
  14 Terminated on error
  15 Terminated on abort
  16 Terminated on suspension
  17 Terminated normally
  18 Restart

  Job status that can be modified with the Set function:
  15 Terminated on abort
  16 Terminated on suspension
  18 Restart
  19 Scheduled
  20 Terminated but not started
  21 TerminatedNot Reachable

■ `JobStrategyName`	Strategy name (`Job` type backup, synthetic backup, export source and export target): 1 Strategy A 2 Strategy B 3 Strategy C 4 Strategy D otherwise, the returned value is -1.
■ `JobType`	Type of job: 1 Backup 2 Synthetic backup 3 Archiving 4 Restore 5 Duplication 6 Source export 7 Target export 8 Catalog maintenance
■ `JobUser`	Handle of user/job owner -6 Root user or value returned: "Invalid handle".

☞ For an example of a script implementing a `Job` object, refer to “Getting a Job List and its Characteristics”, page 130.

Alarm Object

The `Alarm` object represents an alarm triggered during an operation, linked to an object indicated by the `AlarmObjectHandle` attribute.

Note This object cannot be created since it only gathers information.

Attributes	Actions				Type
AlarmAcknowledged	- -	- -	set	get	(int)
AlarmCount	- -	- -	- -	get	(int)
AlarmDate	- -	- -	set	get	(int)
AlarmDateLast	- -	- -	- -	get	(int)
AlarmHelpId	- -	create	- -	get	(int)
AlarmHelpMessage	- -	- -	- -	get	(string)
AlarmHelpMessageUtf8	- -	- -	- -	get	(string)
AlarmId	open	- -	- -	get	(int)
AlarmMessage	- -	- -	- -	get	(string)
AlarmObjectClass	- -	- -	- -	get	(int)
AlarmObjectHandle	- -	- -	- -	get	(handle)
AlarmSeverity	- -	- -	- -	get	(int)

Attributes

- `AlarmAcknowledged`

  Acknowledgement of alarm:

0	Unacknowledged
1	Temporary acknowledgement
2	Permanent acknowledgement

- `AlarmCount`

  Number of identical alarms.

- `AlarmDate`

  Date of alarm.

- `AlarmDateLast`

  Date of last identical alarm.

- `AlarmHelp`

  Alarm help ID to associate with an alarm help message. The help ID must be between 0 and 10000.

- `AlarmHelpMessage`
  Alarm help message.

- `AlarmHelpMessageUtf8`
  Alarm help message in Utf8.

- `AlarmId`
  Alarm ID.

- `AlarmMessage`
  Text of alarm message.

- `AlarmObjectClass`
  Object class concerned by the alarm:

1	Host class
2	Drive class
3	Access group class
4	User class
5	Library class
6	Cartridge class
7	Application class

- `AlarmObjectHandle`
  Handle of object concerned by the alarm.

- `AlarmSeverity`
  Level of severity of alarm:

1	Minor severity
2	Major severity
3	Critical severity

Note To create a new alarm on an object, use the API. Refer to TNAlarm.

Backup Related Objects

Strategy Object

The `Strategy` object represents a backup strategy in the **Time Navigator** sense. A strategy is associated with a group of hosts or applications (`HostGroup`).

Attributes	Actions					Type
StrategyContErrMultipleWriting	-	-	create	set	get	(int)
StrategyEpilog	-	-	create	set	get	(string)
StrategyFioMode	-	-	create	set	get	(int)

Attributes	Actions				Type
StrategyFormat	- -	create	set	get	(int)
StrategyFtxAccessGroup	- -	- -	set	get	(handle list)
StrategyFtxMaxParallelIdxFull	- -	- -	set	get	(int)
StrategyFtxMaxParallelIdxIncr	- -	- -	set	get	(int)
StrategyFullAccessGroup	- -	create	set	get	(handle list)
StrategyFullEnable	- -	create	set	get	(int)
StrategyFullNext	- -	create	set	get	(int)
StrategyFullSchedule	- -	create	set	get	(handle)
StrategyFullSynthetic	- -	create	set	get	(int)
StrategyHostGroup	open	create	- -	get	(handle)
StrategyIncrAccessGroup	- -	create	set	get	(handle list)
StrategyIncrEnable	- -	create	set	set	(int)
StrategyIncrSchedule	- -	create	set	get	(handle)
StrategyLanFree	- -	create	set	get	(int)
StrategyManageACL	- -	create	set	get	(int)
StrategyMultiplexable	- -	create	set	get	(int)
StrategyName	open	create	- -	get	(int)
StrategyNFS	- -	create	set	get	(int)
StrategyOnDemandBackup	- -	create	set	get	(int)
StrategyProlog	- -	create	set	get	(string)
StrategyRelaunchable	- -	create	set	get	(int)
StrategyReplicaDestApp	- -	create	set	get	(handle)
StrategyReplicaKeepInst	- -	create	set	get	(int)
StrategyRetryInterval	- -	create	set	get	(int)
StrategyRetryNumber	- -	create	set	get	(int)
StrategySnapAccessGroup	- -	create	set	get	(handle)
StrategySnapCommand	- -	create	set	get	(string)
StrategySnapKeep	- -	create	set	get	(int)
StrategyStandBy	- -	- -	set	get	(int)
StrategySynchro	- -	create	set	get	(int)
StrategyType	- -	create	set	get	(int)

Attributes

- StrategyContErrMultipleWriting

  Specifies if a backup must continue if one of the multiple writing sessions fails.

  0 Backup does not continue.

  1 Backup continues.

- StrategyEpilog

  Post processing script.

- `StrategyFioMode`
  Specifies if the Fastrax serverless backup mode is enabled.
  0 Disabled
  1 Enabled

- `StrategyFormat`
  Data write format:
  1 tar format
  2 Cpio format
  3 TiNa format
  5 Fastrax format
  6 Sidf format
  7 Unknown format

- `StrategyFtxAccessGroup`
  List of handles of cartridge pool type access groups used for Fastrax. Can only be used if the Fastrax backup mode is enabled (`StrategyFioMode=1`).

- `StrategyFtxMaxParallelIdxFull`
  Maximum parallel index for full backups. Can only be used if the Fastrax backup mode is enabled (`StrategyFioMode=1`).

- `StrategyFtxMaxParallelIdxIncr`
  Maximum parallel index for incremental backups. Can only be used if the Fastrax backup mode is enabled (`StrategyFioMode=1`).

- `StrategyFullAccessGroup`
  List of handles of cartridge pool type access groups used for full backups.

- `StrategyFullEnable`
  0 Full backup deactivated
  1 Full backup activated.

- `StrategyFullNext` Date of the next full backup of the strategy.

- `StrategyFullSchedule` Schedule for full backup strategy.

- `StrategyFullSynthetic`
  0 Synthetic backup deactivated
  1 Synthetic backup activated

- `StrategyHostGroup`
  Handle of the group of hosts or applications associated with the strategy.

- `StrategyIncrAccessGroup`
  List of handles of cartridge pool type access groups used for incremental backups.

- `StrategyIncrEnable`
  0 Incremental backup deactivated
  1 Incremental backup activated.

- `StrategyIncrSchedule` Schedule for incremental backup strategy.

- `StrategyLanFree`
  Specifies if the LAN-free backup mode is enabled.
  0 Disabled.
  1 Enabled.
  Must not be used if the macromutiplexing is enabled (`StrategyMultiplexable`=1 or 2) or if multiple writing is performed.

- `StrategyManageACL`
  Backup of Access Control Lists (ACL).
  0 No
  1 Yes

- `StrategyMultiplexable`
  Backs up in macro-multiplexing mode:
  0 None.
  1 For full and incremental backups.
  2 For incremental backup only.

- `StrategyName`
  Strategy name:
  1 Strategy A
  2 Strategy B
  4 Strategy C
  8 Strategy D

- `StrategyNFS`
  Crossing NFS links:
  0 Do not cross NFS links
  1 Cross NFS links

- `StrategyOnDemandBackup`
  Indicates if the strategy is usable or not by on-demand backups:
  0 No
  1 Yes

- `StrategyProlog`
  Preprocessing script.

- `StrategyRelaunchable`
  Specifies if the backup session is relaunched until platform is reachable.
  0 The backup session is not relaunched
  1 The backup session is relaunched

- `StrategyReplicaDestApp` Application replication destination.

- `StrategyReplicaKeepInst` Keeps an instance of a replica.

- `StrategyRetryInterval` Interval between two retries on incident.

- `StrategyRetryNumber` Number of retries on incident.

- `StrategySnapAccessGroup` Gets the access group for a strategy snapshot.

- `StrategySnapCommand` Takes a snaphot of the strategy.

- `StrategySnapKeep` Keeps a strategy snaphot that has been taken.

- `StrategyStandBy`
  Specifies if the backup session is waiting for a platform to be brought online.
  0 Strategy is not on standby.
  1 Incremental strategy is on standby (get).
  2 Full strategy is on standby (get).
  4 Force to reset incremental strategy (set).
  8 Force to reset full strategy (set).

- `StrategySynchro`
  Synchronization of writing to cartridges:
  0 No
  1 Yes

- `StrategyType` Strategy types are: Backup (1), SnapShot (2), Replication (3).

Note The definition of a strategy is only part of the backup definition. The backup is only completely defined after positioning one or more of the backup classes.

NOTE The new attributes `StrategyFullSchedule` and `StrategyIncrSchedule` are **NOT** compatible with the deleted strategy attributes in version 4.1. You **MUST** modify and recompile any executable or tina_shell script using the old attributes. See the following section for details.

Backup ClassObject

The `BackupClass` object represents a backup class. A backup class is associated to a group of hosts or applications (`HostGroup`) and applies to all platforms in the group. It determines the directories to back up within the associated strategies, regardless of any filter currently set.

Attributes	Actions				Type
BackupClassDayNumber	- -	create	set	get	(int)
BackupClassFilterDateOn	- -	create	set	get	(int)
BackupClassFilterNameOn	- -	create	set	get	(int)
BackupClassFilterSizeOn	- -	create	set	get	(int)
BackupClassFormat	- -	create	set	get	(int)
BackupClassHostGroup	open	create	- -	get	(handle)
BackupClassMaxSize	- -	create	set	get	(int)
BackupClassPath	open	create	- -	get	(string)
BackupClassPhaseTime	- -	create	set	get	(int)
BackupClassReject	- -	create	set	get	(string)
BackupClassSelect	- -	create	set	get	(string)
BackupClassStrategyName	- -	create	set	get	(int)

Attributes

- `BackupClassDayNumber` — Filter selection of the last modified date. **Time Navigator** backs up only the files having been modified since the days that are more recent than the one specified. True only if `BackupClassFilterDateOn = 1`.

- `BackupClassFilterDateOn` — Date modification filter:
  0 The date modification filter is off
  1 The date modification filter is on

- `BackupClassFilterNameOn` — Selection filter or exclusion of the name:
  0 The selection filter or exclusion of the name is off
  1 The selection filter or exclusion of the name is on

- `BackupClassFilterSizeOn` — Size filter:
  0 The size filter is off
  1 The size filter is on

- `BackupClassFormat`

  Cartridge format:
  0 None
  1 Compressed
  2 Encoded
  These attributes can be combined.

- `BackupClassHostGroup`

  Handle of associated group of hosts or applications.

- `BackupClassMaxSize`

  Filter selection of the file size. **Time Navigator** backs up only the files whose size does not exceed the specified value. It is only true if `BackupClassFilterSizeOn=1`:
  0 Infinite
  1 100 bytes
  2 1 KB
  3 10 KB
  4 100 KB
  5 1 MB
  6 10 MB
  7 100 MB
  8 1 GB

- `BackupClassPath`

  Class name, such as the absolute access path in **Time Navigator** (format Unix = `/usr/people1/bjr/files`).

- `BackupClassPhaseTime`

  Phase time mask. The value of this attribute corresponds to the value of $2^{\texttt{activation hour}}$.
  If you want to activate the backup between 0:00 and 1:00, the attribute value is 2^0, therefore 1. Refer to the table below for the different attribute values.

Exponent of 2	Attribute Value	Activation Hour
2^0	1	0:00
2^1	2	1:00
2^2	4	2:00
2^3	8	3:00
2^4	16	4:00
2^5	32	5:00
2^6	64	6:00
2^7	128	7:00
…	…	…
2^{21}	2097152	21:00
2^{22}	4194304	22:00
2^{23}	8388608	23:00
2^{24}	16777215	All schedules
	0	0:00

- `BackupClassReject` — Exclusion mask. Rejection of character string, with UNIX syntax. The constraints to be verified are separated by a blank space. Example: `"*.o core"`

- `BackupClassSelect` — Selection filter. Selection character string, with UNIX syntax. The constraints (filter elements) to be verified are separated by a blank space.
  Example: `"/usr/* /bin/*"`
  This is true only if `BackupClassFilterNameOn=1`

- `BackupClassStrategyName` — Strategy mask:
  1 Strategy A
  2 Strategy B
  4 Strategy C
  8 Strategy D
  You can define several strategies by combining the values. (Ex: 15=1+2+4+8 equivalent to all strategies).

Backup Object

The `Backup` object represents a backup under **Time Navigator**. The `Backup` object can only be created and is used to launch backups.

This object does not exist as such in `tina_shell`. At its creation, it is a handle of the `Job` object that is returned. A `Folder` object is automatically created at the first host backup. It is possible to open the object to read its attributes.

Attributes	Actions				Type
BackupDate	- -	create	- -	- -	(int)
BackupFileList	- -	create	- -	- -	(string list)
BackupFormat	- -	create	- -	- -	(int)
BackupMode	- -	create	- -	- -	(int)
BackupNoErrOnBckp	- -	create	- -	- -	(boolean)
BackupNoRewind	- -	create	- -	- -	(int)
BackupPassword	- -	create	- -	- -	(string)
BackupPlatformClass	- -	create	- -	- -	(int)
BackupPlatformHandle	- -	create	- -	- -	(handle)
BackupStrategyName	- -	create	- -	- -	(int)
BackupSynchro	- -	create	- -	- -	(int)
BackupUser	- -	create	- -	- -	(string)

Attributes

- `BackupDate` — Official date of backup operation.

Warning This attribute makes it possible to antedate the result but not define a schedule for backups.

- `BackupFileList` — Path of a file containing the list of the files to be backed up. Allows backing up files without defining a class.

- `BackupFormat` — Indicates the backup data format.
  0 No format
  1 Compressed format
  2 Encoded format
  You can define several formats by combining the values in a mask. (Ex: 3=1+2 equivalent to compressed and encoded).

- `BackupMode` — Backup mode:
  1 Full mode (by default)
  2 Incremental mode
- `BackupNoErrOnBckp` — Determines if an error is returned or not upon an error in the operation.
- `BackupNoRewind` — No-rewind mode.
- `BackupPassword` — Password of the user.
- `BackupPlatformClass` — Class of backed up platform:
  1 Host
  2 Application
- `BackupPlatformHandle` — Handle of backed up platform.
- `BackupStrategyName` — Name of the strategy(s) to be taken into account:
  1 Strategy A
  2 Strategy B
  4 Strategy C
  8 Strategy D
- `BackupSynchro` — Wait for end or write on cartridges.
- `BackupUser` — User having the access rights to the machine to be backed up.

☞ For an example regarding the `Backup` object, refer to “Launching a Backup”, page 128.

Schedule Object

The `Schedule` object represents a schedule under **Time Navigator**.

Attributes	Actions				Type
ScheduleComment	- -	create	set	get	(string list)
ScheduleCommentUtf8	- -	create	set	get	(string list)
ScheduleName	open	create	set	get	(string list)
ScheduleNameUtf8	open	create	set	get	(string list)
ScheduleProperties	- -	create	set	get	(int)
ScheduleRules	- -	- -	- -	get	(handle)

Attributes

- `ScheduleComment` — Specifies the comment associated to the schedule.
- `ScheduleCommentUtf8` — Specifies the comment associated to the schedule in UTF-8.
- `ScheduleName` — Specifies the schedule name.
- `ScheduleNameUtf8` — Specifies the schedule name in UTF-8.
- `ScheduleProperties` — Specifies the schedule properties which can be:
  SchedulePropertiesNone: specifies no properties.
  SchedulePropertiesVerbose: spedifies properties in detail.
- `ScheduleRules` — List of schedule rules associated with a schedule.

Scheduler Object

The scheduler object represents a scheduler in **Time Navigator**.

You can open it without specifying an attribute.

Attributes	Actions				Type
SchedulerAlarmUnit	- -	- -	set	get	(int)
SchedulerAlarmValue	- -	- -	set	get	(int)
SchedulerHolidays	- -	- -	set	get	(string list)
SchedulerMaxNbJobs	- -	- -	set	get	(int)
SchedulerProperties	- -	- -	set	get	(int)
SchedulerTimeoutSchJObs	- -	- -	set	get	(int)
SchedulerTriggerAlarm	- -	- -	set	get	(int)
SchedulerWeekHoliday	- -	- -	set	get	(int)

Attributes

- SchedulerAlarmUnit
  0 none
  1 minute
  2 hour
  3 day
  4 week
- SchedulerAlarmValue
  Period of time after which the alarm is set off if the scheduler is disabled (this value is expressed in the unit chosen in SchedulerAlarmUnit).
- SchedulerHolidays
  Specifies the non-working days. Strings must follow the YYYY-MM-DD pattern (ie 2007-12-25).
- SchedulerMaxNbJobs
  The maximum number of parallel jobs.
- SchedulerProperties
  0 none
  1 disable activity
  2 verbose mode
  3 apply maximum simultaneous job limitation
  4 generates an alarm if the maximum number of jobs is reached
- SchedulerTimeoutSchJobs
  Specifies the duration (in seconds) of the timeout after which a scheduled job not started becomes aborted. The default value is 3600 seconds.
- SchedulerTriggerAlarm
  Sets off an alarm if the scheduler is disabled.
  1 the alarm is on
  0 the alarm is off

■ `SchedulerWeekHoliday`	Specifies the non-working days of the week. 0No Week Holiday 1 Sunday 2 Monday 4 Tuesday 8 Wednesday 16 Thursday 32 Friday 64 Saturday or any combination of these in a mask

ScheduleRule Object

The ScheduleRule object represents a scheduling rule in **Time Navigator**.

Attributes	Actions				Type
ScheduleRuleFreqMonths	- -	create	set	get	(int)
ScheduleRuleFreqMthsDaysVal	- -	create	set	get	(int)
ScheduleRuleFreqMthsDayType	- -	create	set	get	(int)
ScheduleRuleFreqMthsWeekDay	- -	create	set	get	(int)
ScheduleRuleFreqWeeksDays	- -	create	set	get	(int)
ScheduleRuleDaysOffset	- -	create	set	get	(int)
ScheduleRuleDescription	- -	create	set	get	(string)
ScheduleRuleDescriptionUtf8	- -	create	set	get	(string)
ScheduleRuleFreqDays	- -	create	set	get	(int)
ScheduleRuleFreqDaysNDays	- -	create	set	get	(int)
ScheduleRuleFreqDaysNDayVal	- -	create	set	get	(int)
ScheduleRuleFreqDaysOneDay	- -	create	set	get	(int)
ScheduleRuleFreqDaysOneDayVal	- -	create	set	get	(string)
ScheduleRuleFreqMthsNMthsVal	- -	create	set	get	(int)
ScheduleRuleFreqMthsNthDayVal	- -	create	set	get	(int)
ScheduleRuleFrequency	- -	create	set	get	(int)
ScheduleRuleFreqWeeksNWksVal	- -	create	set	get	(int)
ScheduleRuleFreqYear	- -	create	set	get	(int)
ScheduleRuleFreqYearDaysVal	- -	create	set	get	(int)
ScheduleRuleFreqYearDayType	- -	create	set	get	(int)
ScheduleRuleFreqYearMthsVal	- -	create	set	get	(int)
ScheduleRuleFreqYearNthDay	- -	create	set	get	(int)
ScheduleRuleFreqYearWeekDay	- -	create	set	get	(int)
ScheduleRuleFreqYearWeekVal	- -	create	set	get	(int)
ScheduleRuleId	open	- -	- -	get	(int)
ScheduleRuleName	- -	create	set	get	(string)
ScheduleRuleNameUtf8	- -	create	set	get	(string)
ScheduleRulePhaseJobAction	- -	create	set	get	(int)
ScheduleRulePhaseTimeEnd	- -	create	set	get	(string)
ScheduleRulePhaseTimeStart	- -	create	set	get	(string)
ScheduleRuleProperties	- -	create	set	get	(int)
ScheduleRuleSchedule	- -	create	- -	get	(handle)
ScheduleRuleTime	- -	create	set	get	(int list)
ScheduleRuleValidityEndDate	- -	create	set	get	(string)
ScheduleRuleValidityStartDate	- -	create	set	get	(string)

Attributes

- `ScheduleRuleDaysOffset` — Specifies an offset for a day.
  The value can be positive or negative (from -7 to +7).

- `ScheduleRuleDescription` — Specifies the description associated with the schedule rule.
  If nothing is set at creation, a description is automatically generated according to the content.

- `ScheduleRuleDescriptionUtf8` — Specifies the description associated with the schedule rule in UTF-8 format.
  If nothing is set at creation, a description is automatically generated according to the content.

- `ScheduleRuleFrequency` — Specifies the type of schedule.
  The following options (defines??) are available:
  1: day-based frequency
  2: week-based frequency
  3: month-based frequency
  4: year-based frequency

  Attributes to retrieve or set depend on the option. All these options are explained in the following sections.

- `ScheduleRuleId` — Specifies the Id that identifies the schedule rule.

- `ScheduleRuleName` — Specifies the name of the schedule rule.
  The name is mandatory in creation and unique in the schedule.

  Note: there can be several schedule rules with the same name in a specific catalog.

- `ScheduleRuleNameUtf8` — Specifies the name of the schedule rule in UTF8 format.
  The name is mandatory in creation and unique in the schedule.

- `ScheduleRulePhaseJobAction` — Specifies the rule phase job action.
  1: abort job if out of interval
  2: let the job continue if out of interval
  3: let the job continue if out of interval and sets off an alarm

- `ScheduleRulePhaseTimeEnd` — Specifies the rule phase end time.
- `ScheduleRulePhaseTimeStart` — Specifies the rule phase start time.
- `ScheduleRuleProperties` — Specifies the property of the schedule rule. Takes one of the following values:
  1: time slot-based schedule rule
  2: hour-based schedule rule
  4: exclusion-based schedule rule
- `ScheduleRuleSchedule` — Specifies the schedule handle to which the schedule rule is attached.
- `ScheduleRuleTime` — Specifies rule time selections.
  The following defines are available:

1	:00 minutes
2	:05 minutes
4	:10 minutes
8	:15 minutes
16	:20 minutes
32	:25 minutes
64	:30 minutes
128	:35 minutes
256	:40 minutes
512	:45 minutes
1024	:50 minutes
2048	:55 minutes

  To gather several time several time selections, assemble these defines into a mask (with a slash |).
- `ScheduleRuleValidityEndDate` — Specifies the end date of schedule rule validity.
- `ScheduleRuleValidityStartDate` — Specifies the start date of schedule rule validity.

Setting the rule frequency

To set the type of frequency of the scheduling rule, set the attribute `ScheduleRuleFrequency` to one of its four possible values:

- `1`: day-based frequency
- `2`: week-based frequency
- `3`: month-based frequency
  In this case, precise the scheduling rule by setting the attribute `SheduleRuleFreqMonths` to one of its three possible values:

- 1: month defined day by day
- 2: month defined week by week
- 3: month defined in a cycle

- 4: year-based frequency
  In this case, precise the scheduling rule by setting the attribute ScheduleRuleFreqYear to one of its three possible values:
  - 1: year defined day by day
  - 2: year defined week by week
  - 3: year defined in a cycle

Day-based Frequency

The following attributes are usable only if ScheduleRuleFrequency is set to 1:

Attribute	Comment	
ScheduleRuleFreqDays	Specifies the days the schedule rules apply to. The following options?? are available:	
	1	Specifies every N days. The value is set by the attribute: ScheduleRuleFreqDaysNDayVal
	2	Specifies all working days.
	3	Specifies all holidays.
	4	Specifies a specific day. The value is set by the attribute: ScheduleRuleFreqDaysOneDayVal The form must be yyyy-mm-dd.

Attribute	Comment
ScheduleRuleFreqDaysNDayVal	usable ony if ScheduleRuleFreqDays is set to 1
ScheduleRuleFreqDaysOneDayVal	usable ony if ScheduleRuleFreqDays is set to 4

Week-based Frequency

The following attributes are usable only if `ScheduleRuleFrequency` is set to `2`:

Attribute	Comment
`ScheduleRuleFreqWeeksDays`	Specifies one or several days of the week: 1: Sunday 2: Monday 4: Tuesday 8: Wednesday 16: Thursday 32: Friday 64: Saturday To gather several days, assemble them into a mask (with a slash |) . For example: (1|64)
`ScheduleRuleFreqWeeksNWksVal`	Specifies every N weeks.

Month-based Frequency

The following attributes are usable only if `ScheduleRuleFrequency` is set to `3`:

Attribute	Comment
`ScheduleRuleFreqMonths`	Specifies the frequency in months: 1: month defined day by day 2: month defined week by week 3: month defined in a cycle
`ScheduleRuleFreqMthsNMthsVal`	Specifies every N months.

Month defined day by day

The following attribute is usable only if `ScheduleRuleFreqMonths` is set to `1` and `ScheduleRuleFrequency` is set to `3`.

Attribute	Comment
ScheduleRuleFreqMthsDaysVal	Specifies one or several days in a month: 1 Day 01 2 Day 02 4 Day 03 8 Day 04 16 Day 05 32 Day 06 64 Day 07 128 Day 08 256 Day 09 512 Day 10 1024 Day 11 2048 Day 12 4096 Day 13 8192 Day 14 16384 Day 15 32768 Day 16 65536 Day 17 131072 Day 18 262144 Day 19 524288 Day 20 1048576 Day 21 2097152 Day 22 4194304 Day 23 8388608 Day 24 16777216 Day 25 33554432 Day 26 67108864 Day 27 134217728 Day 28 268435456 Day 29 536870912 Day 30 1073741824 Day 31 -2147483648 Last day To gather several days, assemble them into a mask (with a slash |) . For example: (1|16384)

Month Defined Week by Week

The following attributes are usable only if `ScheduleRuleFreq-Months` is set to `2` and `ScheduleRuleFrequency` is set to `3`:

Attribute	Comment
`ScheduleRuleFreqMthsWeekDay`	Specifies one or several days of the week: 1: Sunday 2: Monday 4: Tuesday 8: Wednesday 16: Thursday 32: Fridday 64: Saturday To gather several days, assemble them into a mask (with a slash |). For example: (1|64)
`ScheduleRuleFreqMthsWeekVal`	Specifies one or several weeks in the month: 1: first week 2: second week 4: third week 8: fourth week 16: last week To gather several weeks, assemble them into a mask (with a slash |) . For example: (1|2)

Month Defined in a Cycle

The following attributes are usable only if `ScheduleRuleFreq-Months` is set to `3` and `ScheduleRuleFrequency` is set to `3`:

Attribute	Comment
`ScheduleRuleFreqMthsDayType`	Specifies the type of day: 1: standard day 2: working day 3: holiday
`ScheduleRuleFreqMthsNthDayVal`	Specifies the Nth day of the month (1 to 31).

Year-based frequency

The following attributes are usable only if `ScheduleRuleFrequency` is set to `4`:

Attribute	Comment
`ScheduleRuleFreqYear`	Specifies the frequency in years: 1: year defined day by day 2: year defined week by week 3: year defined in a cycle
`ScheduleRuleFreqYearMthsVal`	Specifies one or several months: 1: January 2: February 4: March 8; April 16: May 32: June 64: August 128: September 256: October 512: November 1024: December To gather several months, assemble them into a mask (with a slash |) .

Year Defined Day by Day

The following attribute is usable only if `ScheduleRuleFreqYear` is set to `1`.

Attribute	Comment
ScheduleRuleFreqYearDaysVal	Specifies one or several days in a month: 1 Day 01 2 Day 02 4 Day 03 8 Day 04 16 Day 05 32 Day 06 64 Day 07 128 Day 08 256 Day 09 512 Day 10 1024 Day 11 2048 Day 12 4096 Day 13 8192 Day 14 16384 Day 15 32768 Day 16 65536 Day 17 131072 Day 18 262144 Day 19 524288 Day 20 1048576 Day 21 2097152 Day 22 4194304 Day 23 8388608 Day 24 16777216 Day 25 33554432 Day 26 67108864 Day 27 134217728 Day 28 268435456 Day 29 536870912 Day 30 1073741824 Day 31 -2147483648 Last day To gather several days, assemble them into a mask (with a slash |) . For example: (1|16384)

Year Defined Week by Week

The following attributes are usable only if `ScheduleRuleFreqYear` is set to `2`.

`ScheduleRuleFreqYearWeekDay` Specifies one or several days of the week:
1: Sunday
2: Monday
4: Tuesday
8: Wednesday
16: Thursday
32: Fridday
64: Saturday

To gather several days, assemble them into a mask (with a slash |).
For example: (1|64)

`ScheduleRuleFreqYearWeekVal` Specifies one or several weeks in the month:
1: first week
2: second week
4: third week
8: fourth week
16: last week

To gather several weeks, assemble them into a mask (with a slash |) .
For example: (1|2)

Year Defined in a Cycle

The following attributes are usable only if `ScheduleRuleFreqYear` is set to `3`.

Attribute	Comment
`ScheduleRuleFreqYearDayType`	Specifies the type of day: 1: standard day 2: working day 3: holiday
`SchedSchedScheduleRuleFreqYearNthDayVal`	Specifies the Nth day of the month (1 to 31).

Snapshot Object

The snapshot object describes a snapshot. Listing the snapshots returns their handles.

Attributes	Actions				Type
SnapShotFolderName	- -	- -	- -	get	(string)
SnapShotName	- -	- -	- -	get	(string)
SnapShotNameUtf8	- -	- -	- -	get	(string)
SnapShotSnapMountPoint	- -	- -	- -	get	(string)
SnapShotSnapMountPointUtf8	- -	- -	- -	get	(string)
SnapShotSnapName	- -	- -	- -	get	(string)
SnapShotSnapNameUtf8	- -	- -	- -	get	(string)
SnapShotStrategyName	- -	- -	- -	get	(int)
SnapShotType	- -	- -	- -	get	(int)
SnapShotVolumeMountPoint	- -	- -	- -	get	(string)
SnapShotVolumeMountPointUtf8	- -	- -	- -	get	(string)
SnapShotVolumeName	- -	- -	- -	get	(string)
SnapShotVolumeNameUtf8	- -	- -	- -	get	(string)

Attributes

- SnapShotFolderName — Handle of the backup folder containing the snapshot information.
- SnapShotName — Name of the snapshot.
- SnapShotNameUtf8 — Name of the snapshot in UTF-8 format.
- SnapShotSnapMountPoint — Mount point of the snapshot.
- SnapShotSnapMountPointUtf8 — Mount point of the snapshot in UTF-8 format.
- SnapShotSnapName — Physical name of the snapshot in UTF-8 format.
- SnapShotSnapNameUtf8 — Physical name of the snapshot.
- SnapShotStrategyName
  1 strategy A
  2 strategy B
  3 strategy C
  4 strategy D
- SnapShotType
  1 snapshots using VSS technology
  2 snapshots using NDMP technology
  3 snapshots using Snapvault technology
  4 snapshots using Engenio technology
- SnapShotVolumeMountPoint — Mount point of the snapshotted volume.
- SnapShotVolumeMountPointUtf8 — Mount point of the snapshotted volume in UTF-8 format.

- `SnapShotVolumeName` Mount point of the snapshotted volume in UTF-8 format.
- `SnapShotVolumeNameUtf8` Mount point of the snapshotted volume.

Archive Related Objects

Folder Object

The `Folder` object represents a folder, identified by the name and the type:

- Backup folder: a backup folder is associated to a host (`Host`) or an application (`Application`), themselves associated with a group of platforms (`HostGroup`). It is at the group of platforms level that the strategy and the associated cartridge pools are defined, as well as the backup classes. The backup file can be opened using its name or the associated object, `Host` or `Application`. It is automatically created with the creation of the `Host` object.
- Archive folder (central or local): an archive folder does not contain links to the archive contents. The only way to know which files are contained is to list all the files of the **Time Navigator** catalog. Then test the equality between the name of the selected archive folder and the one it seeks

The central archive has a direct link with the cartridge pools to be used.
The local archives have a link to cartridge pools via the user. It is identified by its name.

Attributes	Actions				Type
FolderAccessGroup	- -	create	set	get	(handle list)
FolderApplication	open	- -	- -	get	(handle)
FolderArchive	- -	- -	- -	get	(handle list)
FolderContErrMultWriting	- -	create	set	get	(int)
FolderCreateNewArchive	- -	create	set	get	(int)
FolderDateArchive	- -	- -		get	(int)
FolderDateCreate	- -	- -	- -	get	(int)
FolderDateExtract	- -	- -	- -	get	(int)
FolderDateModif	- -	- -	- -	get	(int)
FolderDeleteArchive	- -	create	set	get	(int)
FolderEpilog	- -	create	set	get	(string)
FolderFormatCart	- -	create	set	get	(int)
FolderFormatFile	- -	create	set	get	(int)
FolderHost	open	- -	- -	get	(handle)
FolderJobPriority	- -	create	set	get	(int)
FolderKeyWord	- -	create	set	get	(string list)
FolderKeyWordMandatory	- -	create	set	get	(int)
FolderKeyWordPropagate	- -	- -	set	- -	(int)
FolderLanFree	- -	create	set	get	(int)
FolderLibelle	- -	create	set	get	(string)
FolderManageACL	- -	create	set	get	(int)

Attributes	Actions				Type
FolderMultiplexable	- -	create	set	get	(int)
FolderName	open	create	set	get	(string)
FolderOSGroup	- -	create	set	get	(string)
FolderOSGroupUtf8	- -	create	set	get	(string)
FolderOSGroupId	- -	create	set	get	(int)
FolderOsType	- -	- -	- -	get	(int)
FolderOSUser	- -	create	set	get	(string)
FolderOSUserUtf8	- -	create	set	get	(string)
FolderOSUserId	- -	create	set	get	(int)
FolderPermission	- -	create	set	get	(int)
FolderPNbFiles	- -	create	set	get	(string)
FolderProlog	- -	create	set	get	(string)
FolderPSize	- -	create	set	get	(string)
FolderSynchro	- -	create	set	get	(int)
FolderThroughLink	- -	create	set	get	(int)
FolderType	- -	create	- -	get	(int)

Attributes

- `FolderAccessGroup` — List of handles of cartridge pool type access group associatedwith folder. Only valid for a central archiving type folder (`FolderType` = 3).
- `FolderApplication` — Handle associated application, for a backup folder (`FolderType` = 1).
- `FolderArchive` — List of handles of the archives contained in the archive folder.
  Only valid for a local or central archiving type folder (`FolderType` = 2 or 3) .
- `FolderContErrMultWriting` — Specifies if an archiving session must continue if one of the multiple writing sessions fails.
  0 Archiving does not continue.
  1 Archiving continues.
- `FolderCreateNewArchive` — Archiving in a empty archive:
  0 No
  1 Yes
- `FolderDateArchive` — Last archive date of folder.
- `FolderDateCreate` — Creation date of folder.
- `FolderDateExtract` — Last restore date of folder.

- `FolderDateModif` — Modification date of folder.
- `FolderDeleteArchive` — Deletion of archived files:
  0 No (by default)
  1 Yes
- `FolderEpilog` — Ends processing command.
- `FolderFormatCart` — Format of data writing:
  1 tar
  2 cpio
  4 TiNa
  5 fastrax
  6 sidf
  7 Unknown
- `FolderFormatFile` — File format, true only if the data writing format is TiNa (`FolderFormatCart` = 4):
  1 Encoded
  2 Compressed
  3 Encoded and compressed
- `FolderHost` — Handle of associated host, for a backup folder (`FolderType` = 1).
- `FolderJobPriority` — Specifies the priority level of jobs associated with the archive folder.
  1 Very low job priority
  2 Low job priority
  3 Medium job priority
  4 High job priority
  5 Very high job priority
- `FolderKeyWord` — List of keywords.
- `FolderKeyWordMandatory` — Mandatory keyword list.
- `FolderKeyWordPropagate` — Keyword list propagation.
- `FolderLanFree` — Specifies if the LAN-free archiving mode is enabled.
  0 Disabled (default).
  1 Enabled.
  Must not be used if multiple writing is performed.
- `FolderLibelle` — Folder description.

- `FolderManageACL` — Archives the Access Control Lists (ACL)
  0 No
  1 Yes
- `FolderMultiplexable` — Archives in multiplexing mode.
  Only valid for a local or central archiving type folder (`FolderType` = 2 or 3).
  0 No
  1 Yes
- `FolderName` — Unique name of folder, whatever the type.
- `FolderOSGroup` — Name of group to which the owner user belongs.
  Only valid for a local or central archiving type folder (`FolderType` = 2 or 3).
- `FolderOSGroupUtf8` — Name of group to which the owner user belongs in Utf8.
  Only valid for a local or central archiving type folder (`FolderType` = 2 or 3).
- `FolderOSGroupId` — ID of group to which the owner user belongs.
- `FolderOSType` — Returns an integer corresponding to the OS on which the folder was created:
  1 Netware
  2 Unix
  3 VMS
  4 Win3
  5 Win32
  6 MacOS
  7 OS2
- `FolderOSUser` — Name of owner user. Only valid for a local or central archiving type folder. (`FolderType` = 2 or 3).
- `FolderOSUserUtf8` — Name of owner user in Utf8. Only valid for a local or central archiving type folder. (`FolderType` = 2 or 3).
- `FolderOSUserId` — ID of owner user.
- `FolderPermission` — Access rights of folder:
  1 Owner read permission
  2 Owner write permission
  4 Group read permission
  8 Group write permission
  16 Everyone read permission
  32 Everyone write permission
  7 Values of the default access rights

Note The default values of the access rights correspond to the owner's read permission and the group, and for the write permission to the owner's permission only.

- `FolderPNbFiles` — Specifies the number of protected files in an archive folder .
- `FolderProlog` — Starts processing command.
- `FolderPSize` — Folder size.
- `FolderSynchro` — Synchronization of write cartridges:
  0 No (by default)
  1 Yes
- `FolderThroughLink` — Cross symbolic links:
  0 No (by default)
  1 Yes
- `FolderType` — Type of folder:
  1 Backup
  2 Local archive
  3 Central archive

Specific Cases

- `FolderHost` — Mandatory on opening for a backup folder type (`FolderType` = 1).
- `FolderName` — Mandatory on creation and opening for a local or central archiving type folder (`FolderType` = 2 or 3).
- `FolderAccessGroup` — Mandatory on creation for a central archiving type folder (`FolderType` = 3).

Archive Object

The `Archive` object represents an archive within an archive folder. An archive is identified by the folder name it belongs to, completed by its access path within this folder.

The access path is always defined in relation to the root of the folder. Its syntax is standard: `/Archive1/Arc1` represents sub-archive `Arc1` archive `Archive1` in the associated archive folder.

Attributes	Actions				Type
ArchiveDateArchive	- -	- -	- -	get	(int)
ArchiveDateCreate	- -	- -	- -	get	(int)
ArchiveDateExtract	- -	- -	- -	get	(int)
ArchiveDateModif	- -	- -	- -	get	(int)
ArchiveFolder	- -	create	- -	get	(handle)
ArchiveKeyWord	- -	create	set	get	(string list)
ArchiveLibelle	- -	create	set	get	(string)
ArchiveName	open	create	set	get	(string)
ArchiveOSGroup	- -	create	set	get	(string)
ArchiveOSUser	- -	create	set	get	(string)
ArchivePermission	- -	create	set	get	(int)

Attributes

- `ArchiveDateArchive` — Date of last archiving of the last archive.
- `ArchiveDateCreate` — Archive creation date.
- `ArchiveDateExtract` — Date of last restore of the archive.
- `ArchiveDateModif` — Archive modification date.
- `ArchiveFolder` — Handle of the folder to which the archive belongs.

Note This folder contains the archive that establishes the link with the cartridge pools, either directly in the case of central archiving, or with the user name in the case of a local archive.

- `ArchiveKeyWord` — List of keywords.
- `ArchiveLibelle` — Description.

- `ArchiveName` — Absolute access path of archive in folder.
- `ArchiveOSGroup` — Name of group to which the owner user belongs.
- `ArchiveOSUser` — Name of owner user.
- `ArchivePermission` — Access rights of the archive:
  - 1 Owner read permission
  - 2 Owner write permission
  - 4 Group read permission
  - 8 Group write permission
  - 16 Everyone read permission
  - 32 Everyone write permission
  - 7 Values of the default access rights

Note The default values of the access rights correspond to the owner read permission and the group, and for the write permission for the owner only.

DFM Archive Object

The `Disk File Management (DFM)` object is now available. This object has the following characteristics:

- no tar or cpio format available
- no manual archiving via **Time Navigator** is allowed
- the psize field cannot be filled

For more information on DFM, contact Atempo Professional Services and ask for the **Time Navigator Disk File Management** guide.

Appendix

Appendix 1: Conventions

The following conventions are designed to homogenize scripts and increase readability.

Extension of Script Files

By convention, the names of script files for `tina_shell` end with "`.tsh`".

Variables

By convention and to avoid any confusion with the attribute names, the names of `tina_shell` variables are written in capital letters, words being separated by the underscore character '_'.

This applies to all four variables present (by default) when starting `tina_shell`.

Appendix 2: List of Commands

Command	Description	Page
`add`	adds an element to the end of a list	18
`assign`	assigns a value to an attribute	33
`close`	closes an object	37
`concat`	concatenates the value of two variables	16
`create`	creates an object	36
`date`	converts the number of seconds since 01/01/1970 to a date	26
`decrement`	decrements a variable	15
`delete`	deletes a variable of type handle	36
`echo`	displays the value of a variable	14
`elif`	starts a test branch	21
`else`	ends a test branch	21
`endfor`	ends a repeated loop	22
`endif`	ends a test branch	21
`envget`	retrieves the value of an environment variable	24
`envput`	positions an environnement variable	24
`erase`	deletes variables as well as attribute assignments	37
`exist`	tests the existence of an object in the catalog	40
`exit`	quits `tina_shell`	9
`fileget`	retrieves the content of a text file	25
`fileput`	creates a text file containing the elements of a variable	25
`foreach`	starts a repeated loop	22
`get`	retrieves attribute information	38
`help`	displays on-line help	30
`if`	starts a test branch	21
`increment`	increments a variable	15
`input`	establishes a dialogue between the user and `tina_shell`	23

Command	Description	Page
item	counts the elements of a table	18
list	generates a list of handles of all the objects of a class	39
mask	generates a bit mask from all the integers in a list	19
multiply	multiplies the values of two variables	15
open	opens an object	37
percent	calculates the percentage of the values of two variables	
quit	quits tina_shell	9
reset	table attributes are returned to zero	35
set	modifies the information concerning attributes	38
show	displays all the variables	13
time	places the number of seconds that have passed since 01/01/1970	26
unmask	generates an integer list from an integer representing a bit mask	19
variable	creates a variable	12
wait	stops the program during a number of specified seconds	26

In the following examples, files are used with the command -file file.

- get_host.tsh

Recovers and displays the names of all the hosts of the application.

- enable.tsh

Enables all the hosts, applications and drives of the application.

- change_server_name.tsh

Modifies dynamically the name of the **Time Navigator** server.

- job.tsh

Controls the status (**abort**, **suspend**, **resume**, or **restart**) and the priority (up, down, top, bottom) of a job.

- config.tsh

Creates a test configuration.

- backup.tsh

Starts a backup.

Appendix 3: Relations between objects

This table summarizes the relations that links the objects.

- The **Administration Object** column lists the objects handled in **Administration Console**.
- The **tina_shell Object** column indicates, for each administration object, the corresponding `tina_shell` objects.
- The **Dependence** indicates the objects whose handle is necessary for creating/opening the `tina_shell` object.

☞ To make the distinction between the commands `create` and `open`, see "On-line Help", page 30 and "Objects and their Attributes", page 43.

- The **Child Objects** column lists the different objects of which this object `tina_shell` authorises access.
- The **Parent Objects** column lists the different objects that allow access to each `tina_shell` object.

Example The `tina_shell Application` object represents the **Application** object of the administration. It cannot be created nor opened without the handle of the `HostGroup` and `Host` objects. It allows access to the `Alarm` objects, `HostGroup` and `Host`. One can get this object via the `Folder`, `HostGroup`, `Backup` and `Job` objects.

Administration Object	tina_shell Object	Dependence	Child Objects	Parent Objects

Alarm	`Alarm`		`Host`
			`Application`
			`Drive`
			`User`
			`AccessGroup`
			`Library`
			`Cartridge`
				`Host`
				`Application`
				`Drive`
				`User`
				`AccessGroup`
				`Library`
				`Cartridge`

Application	`Application`	`HostGroup`
		`Host`
			`Alarm` (list)
			`HostGroup`
			`Host`
				`Folder`
				`HostGroup`
				`Backup`
				`Job`

Archive	`Archive`	`Folder` archive type	`Folder` archive type	
Catalog	`Catalog`			

Administration Object	tina_shell Object	Dependence	Child Objects	Parent Objects
Cartridge	Cartridge	Drive User associated to an Access-Group of cartridge pool type	Drive Alarm (list) User	User Drive Library
Backup class	BackupClass	HostGroup	HostGroup	HostGroup
Central archive folder	Folder		AccessGroup cartridge pool type	
Local archive folder	Folder			
Backup folder	Folder		Host or Application	
Platform group	HostGroup		Host (list) Application (list) BackupClass (list) Strategy (list)	BackupClass
Cartridge pool	User		AccessGroup cartridge pool type Cartridge Alarm (list)	
	AccessGroup cartridge pool type		User associated to AccessGroup Alarm (list) Drive (list)	User Folder central archive type Drive Strategy
Drive	Drive	Host	Cartridge AccessGroup user type (list) AccessGroup library type AccessGroup cartridge pool type (list) Alarm (list) Host Network DriveConnection	Cartridge AccessGroup library type AccessGroup user type (list) Host Network DriveConnection
Library	Library	Host	AccessGroup library type Cartridge (list) Alarm (list) Host	AccessGroup library type Host
	AccessGroup library type		Library Drive Alarm (list)	Drive Library

Administration Object	tina_shell Object	Dependence	Child Objects	Parent Objects
Backup	Backup	Host or Application	Host or Application	
Host	Host	HostGroup	Drive Library Alarm (list) HostGroup Host	Application Folder HostGroup Drive Library Backup Host Job
Strategy	Strategy	HostGroup	AccessGroup cartridge pool type (list) HostGroup	HostGroup
Job	Job		User Host or Application	
User	User		Cartridge (liste) for local archive Alarm (list) AccessGroup of user type	AccessGroup cartridge pool type (list) AccessGroup user type Job
User group	AccessGroup of user type		User associated to this AccessGroup Drive (list) Alarm (list)	Drive
Network	Network		Drive (list)	Drive
Drive/host Connection	DriveConnection		Host Drive	Drive

Appendix 4: Examples of tina_shell Scripts

This appendix provides several examples of commonly used scripts.

Each script is explained by the comment lines in the code (lines starting with #).

Getting and Displaying Host Names

get_host.tsh

```
#-------------------------------------------------------------------------#
# Title : get_host.tsh
# Description: Get and display the host name
# Use: tina_shell -file get_host.tsh
# The above line executes the script
# The variables are set and the host handle list
# is read and shown.
#-------------------------------------------------------------------------#
# variables
variable int INT_SUCCESS 0
variable int INT_ERROR 1
#Get host handle list
list Host in HDL_LIST_HOST
foreach HDL_HOST in HDL_LIST_HOST
  assign HostName &STR_HOST_NAME
  get HDL_HOST
  echo STR_HOST_NAME
endfor
exit SUCCESS
```

This script typically produces an output similar to the following:

```
BRAGON
elliot
kucek
```

Enabling Hosts, Applications and Drivers

enable.tsh

```
#-------------------------------------------------------------------------#
# Title: enable.tsh
# Description: enable the hosts, applications and drives.
# Use: tina_shell -file enable.tsh
# tina_shell
# Variables are defined by type with a value to indicate possible success or error
# The host handle list is retrieved through an iterative process (foreach...endfor)
# The application handle list is retrieved through an iterative process (foreach...endfor)
# The drive handle list is retrieved through an iterative process (foreach...endfor).
#-------------------------------------------------------------------------#
#variables
variable int INT_SUCCESS 0
variable int INT_ERROR 1
#Get host handle list
list Host in HDL_LIST_HOST
foreach HDL_HOST in HDL_LIST_HOST
assign HostEnable TRUE
  set HDL_HOST
endfor
# Get application handle list
list Application in HDL_LIST_APPLICATION
foreach HDL_APPLICATION in HDL_LIST_APPLICATION
  assign ApplicationEnable TRUE
  set HDL_APPLICATION
endfor
# Get drive handle list
list Drive in HDL_LIST_DRIVE
foreach HDL_DRIVE in HDL_LIST_DRIVE
  assign DriveEnable TRUE
  set HDL_DRIVE
endfor
exit SUCCESS
```

This script does not produce any output but enables hosts, applications, and drivers.

Launching a Backup

backup.tsh

```
#-------------------------------------------------------------------------------------------#
# Title: backup.tsh
# Description: Backup a host
# Use tina_shell -file backup.tsh -host host_name -strat A|B|C|D [-full|-incr]
#tina_shell
# Variables are defined by type with a value to indicate possible success or error.
# For each argument in ARGV, a test is made
# The syntax is verified, through an iterative process (foreach...endfor).
#
#-------------------------------------------------------------------------------------------#
# variables
variable int INT_SUCCESS 0
variable int INT_ERROR 1
# control ARGV
item ARGV ARGC
if ARGC < 7
  echo Usage : -host host_name -strat A|B|C|D [-full|-incr]
  exit ERROR
endif
# read ARGV
variable int INT_I 0
variable string STR_HOST_NAME EMPTY_STRING
variable string STR_STRATEGY_NAME EMPTY_STRING
variable int INT_MODE 1
foreach STR_ARG in ARGV
  # -host
  if STR_ARG == -host
    variable int INT_J INT_I
    increment INT_J 1
    variable string STR_HOST_NAME ARGV[INT_J
  endif
  # -strat
  if STR_ARG == -strat
    variable int INT_J INT_I
    increment INT_J 1
    variable string STR_STRATEGY_NAME ARGV[INT_J
  endif
  # -incr (-full by default)
  if STR_ARG == -incr
    variable int INT_MODE 2
  endif
  increment INT_I 1
endfor
# Verify the syntax
if STR_HOST_NAME == EMPTY_STRING
  echo Specify host_name
  echo Usage : -host host_name -strat A|B|C|D [-full|-incr]
  exit ERROR
endif
if STR_STRATEGY_NAME == A
  variable int INT_STRATEGY 1
elif STR_STRATEGY_NAME == B
  variable int INT_STRATEGY 2
elif STR_STRATEGY_NAME == C
  variable int INT_STRATEGY 4
elif STR_STRATEGY_NAME == D
  variable int INT_STRATEGY 8
else
  echo Specify strategy
  echo Usage : -host host_name -strat A|B|C|D [-full|-incr]
  exit ERROR
endif
# Open the host
assign HostName STR_HOST_NAME
exist Host INT_EXIST
```

```
if INT_EXIST == FALSE
  echo Host STR_HOST_NAME does not exist
exit ERROR
endif
open Host HDL_HOST
# Backup the host
assign BackupPlatformHandle HDL_HOST
assign BackupPlatformClass 1
assign BackupStrategyName INT_STRATEGY
assign BackupMode INT_MODE
create Backup HDL_JOB
close HDL_JOB
exit SUCCESS
```

This script does not produce any output but launches a backup.

Getting a Job List and its Characteristics

list_jobs

```
#-------------------------------------------------------------------------------------------#
# Title: list_jobs.tsh
# Description: job list
# Use: tina_shell -file list_jobs.tsh
# tina_shell
# The job is retrieved in the handle list through an iterative process (foreach...endfor)
# The various dates variables are set and the result is displayed.
#-------------------------------------------------------------------------------------------#
list Job in HDL_LIST_JOB
foreach HDL_JOB in HDL_LIST_JOB
  assign JobId &INT_JOB_ID
  assign JobType &INT_JOB_TYPE
  assign JobStatus &INT_JOB_STATUS
  assign JobDateSubmit &INT_DATE_SUBMIT
  assign JobDateCreate &INT_DATE_CREATE
  assign JobDateRun &INT_DATE_RUN
  assign JobDateEnd &INT_END_DATE
  assign JobStrategyName &INT_STRATEGY_NAME
  assign JobHost &HDL_HOST
  get HDL_JOB
  assign HostName &STR_HOST_NAME
  get HDL_HOST
  date INT_DATE_SUBMIT STR_DATE1
  date INT_DATE_CREATE STR_DATE2
  date INT_DATE_RUN STR_DATE3
  date INT_END_DATE STR_DATE4
  #Display the result
  echo id : INT_JOB_ID platform : STR_HOST_NAME strategy : INT_STRATEGY_NAME type : INT_JOB_TYPE
  status : INT_JOB_STATUS
  echo job : STR_DATE1 creation : STR_DATE2 run : STR_DATE3 end : STR_DATE4
endfor
```

This script typically produces an output similar to the following:

```
id : 147 platform : elliot strategy : -1 type : 8
job : Fri Sep 02 12:00:30 2005 creation : Fri Sep 02 12:00:30
2005 run : Fri Sep 02 12:00:32 2005 end : Fri Sep 02 12:00:33 2005
id : 146 platform : elliot strategy : -1 type : 3
job : Fri Sep 02 11:24:20 2005 creation : Fri Sep 02 11:24:20
2005 run : Fri Sep 02 11:24:22 2005 end : Fri Sep 02 11:28:03 2005
id : 145 platform : elliot strategy : -1 type : 4
job : Fri Sep 02 10:52:45 2005 creation : Fri Sep 02 10:52:45
2005 run : Fri Sep 02 10:52:49 2005 end : Fri Sep 02 10:53:16 2005
id : 144 platform : elliot strategy : -1 type : 4
job : Fri Sep 02 10:51:36 2005 creation : Fri Sep 02 10:51:36
2005 run : Fri Sep 02 10:51:39 2005 end : Fri Sep 02 10:51:58 2005
id : 143 platform : elliot strategy : -1 type : 4
job : Fri Sep 02 10:50:46 2005 creation : Fri Sep 02 10:50:46
2005 run : Fri Sep 02 10:50:49 2005 end : Fri Sep 02 10:51:15 2005
id : 142 platform : elliot strategy : -1 type : 4
job : Fri Sep 02 10:45:09 2005 creation : Fri Sep 02 10:45:09
2005 run : Fri Sep 02 10:45:13 2005 end : Fri Sep 02 10:45:35 2005
id : 141 platform : elliot strategy : -1 type : 4
job : Fri Sep 02 10:41:51 2005 creation : Fri Sep 02 10:41:52
2005 run : Fri Sep 02 10:41:55 2005 end : Fri Sep 02 10:42:17 2005
id : 140 platform : elliot strategy : -1 type : 4
job : Fri Sep 02 10:40:57 2005 creation : Fri Sep 02 10:40:57
2005 run : Fri Sep 02 10:41:01 2005 end : Fri Sep 02 10:41:30 2005
id : 139 platform : elliot strategy : -1 type : 4
job : Fri Sep 02 10:39:57 2005 creation : Fri Sep 02 10:39:57
2005 run : Fri Sep 02 10:40:00 2005 end : Fri Sep 02 10:40:07 2005
id : 138 platform : elliot strategy : -1 type : 4
```

Getting Catalog Information

Catalog_info.tsh

```
#-------------------------------------------------------------------------------------------
#Title:catalog_info.tsh -catalog catalog_name [-full] [-out file]
#Description: get catalog information and the cache.
#Use: Display a summary :no parameter (except -catalog)
# Displays all info :-full
# Backup results of the file :-out file
# Arguments are read.
# ------+-----------------------------------+-----------------------------------------------
#
# Variable initialization
#
variable string STR_USAGE_1 "Usage full information:-catalog catalog -full [-out file
(save result)]"
variable string STR_USAGE_2 "Usage summary : -catalog catalog [-out file (save result)]"
variable int INT_I 0
variable string STR_VERSION "1.1 19/04/2000"
variable int INT_TYPE FALSE
variable string STR_CATALOG EMPTY_STRING
variable string STR_ACTION summary
variable string STR_FILE_OUT EMPTY_STRING
variable stringlist STR_LIST_RESULT " "
#
# Read ARGV
#
# Variable initialization
# Read parameters
foreach STR_ARG in ARGV
  # -full
  if STR_ARG == -full
    variable string STR_ACTION full
  endif
  # -catalog
  if STR_ARG == -catalog
    variable int J INT_I
    increment J 1
    variable string STR_CATALOG ARGV[J
  endif
  # -out
  if STR_ARG == -out
    variable int J INT_I
    increment J 1
    variable string STR_FILE_OUT ARGV[J
  endif
  # -help
  if STR_ARG == -help
    echo USAGE_1
    echo USAGE_2
    exit ERROR
  endif
  increment INT_I 1
endfor
# End of drive parameters
# Verify the syntax
if STR_CATALOG == EMPTY_STRING
  echo You must specify a catalog
  echo STR_USAGE_1
  echo STR_USAGE_2
  exit ERROR
endif
#
# Execute the program
#
assign CatalogName STR_CATALOG
open Catalog HDL_CAT
```

```
assign CatalogVolumeTotal &INT_VOLUME
assign CatalogVolumeFree &INT_VOLUME_FREE
assign CatalogObject &INT_OBJ
assign CatalogInstance &INT_INST
assign CatalogDiskCacheVolumeTotal &INT_DSK_VOL
assign CatalogDiskCacheVolumeFree &INT_DSK_FREE
assign CatalogMemoryCacheVolumeTotal &INT_MEM_VOL
assign CatalogMemoryCacheVolumeFree &INT_MEM_FREE
assign CatalogCacheVolumeRead &INT_CACHE_RD
assign CatalogCacheVolumeWrite &INT_CACHE_WR
get HDL_CAT
########################
# get the information#
if STR_ACTION == full
  #
  # result complete
  #
  variable string STR_INFO "Catalog: "
  concat STR_CATALOG STR_INFO
  add STR_INFO in STR_LIST_RESULT
  variable string STR_INFO "OdbVolume: "
  variable string STR_INFO2 INT_VOLUME
  concat STR_INFO2 STR_INFO
  concat " MB" STR_INFO
  add STR_INFO in STR_LIST_RESULT
  variable string STR_INFO "OdbFree: "
  variable string STR_INFO2 INT_VOLUME_FREE
  concat STR_INFO2 STR_INFO
  concat " MB" STR_INFO
  add STR_INFO in STR_LIST_RESULT
  variable string STR_INFO "Object: "
  variable string STR_INFO2 INT_OBJ
  concat STR_INFO2 STR_INFO
  add STR_INFO in STR_LIST_RESULT
  variable string STR_INFO "Instance: "
  variable string STR_INFO2 INT_INST
  concat STR_INFO2 STR_INFO
  add STR_INFO in STR_LIST_RESULT
  variable string STR_INFO "DiskCache: "
  variable string STR_INFO2 INT_DSK_VOL
  concat STR_INFO2 STR_INFO
  concat " MB" STR_INFO
  add STR_INFO in STR_LIST_RESULT
  variable string STR_INFO "DiskCacheFree: "
  variable string STR_INFO2 INT_DSK_FREE
  concat STR_INFO2 STR_INFO
  concat " MB" STR_INFO
  add STR_INFO in STR_LIST_RESULT
  variable string STR_INFO "CacheMemory: "
  variable string STR_INFO2 INT_MEM_VOL
  concat STR_INFO2 STR_INFO
  concat " MB" STR_INFO
  add STR_INFO in STR_LIST_RESULT
  variable string STR_INFO "MemoryFree: "
  variable string STR_INFO2 INT_MEM_FREE
  concat STR_INFO2 STR_INFO
  concat " MB" STR_INFO
  add STR_INFO in STR_LIST_RESULT
  variable string STR_INFO "CacheRead: "
  variable string STR_INFO2 INT_CACHE_RD
  concat STR_INFO2 STR_INFO
  concat " MB" STR_INFO
  add STR_INFO in STR_LIST_RESULT
  variable string STR_INFO "CacheWrite: "
  variable string STR_INFO2 INT_CACHE_WR
  concat STR_INFO2 STR_INFO
  concat " MB" STR_INFO
  add STR_INFO in STR_LIST_RESULT
else
  #
```

```
  # Summary
  #
  variable string STR_INFO "Catalog: "
  concat STR_CATALOG STR_INFO
  add STR_INFO in STR_LIST_RESULT
  variable string STR_INFO "OdbVolume: "
  variable string STR_INFO2 INT_VOLUME
  concat STR_INFO2 STR_INFO
  concat " MB" STR_INFO
  add STR_INFO in STR_LIST_RESULT
  variable string STR_INFO "OdbFree: "
  variable string STR_INFO2 INT_VOLUME_FREE
  concat STR_INFO2 STR_INFO
  concat " MB" STR_INFO
  add STR_INFO in STR_LIST_RESULT
  variable string STR_INFO "CacheDisk: "
  variable string STR_INFO2 INT_DSK_VOL
  concat STR_INFO2 STR_INFO
  concat " MB" STR_INFO
  add STR_INFO in STR_LIST_RESULT
  variable string STR_INFO "CacheMemory: "
  variable string STR_INFO2 INT_MEM_VOL
  concat STR_INFO2 STR_INFO
  concat " MB" STR_INFO
  add STR_INFO in STR_LIST_RESULT
endif
#################################
# File result or "stdout"
#

if STR_FILE_OUT != EMPTY_STRING
  fileput STR_FILE_OUT STR_LIST_RESULT
else
  foreach STR_INFO in STR_LIST_RESULT
    if STR_INFO != " "
      echo STR_INFO
    endif
  endfor
endif
```

This script typically produces an output similar to the following:

```
Catalog: test
OdbVolume: 64 MB
OdbFree: 38 MB
CacheDisk: 0 MB
CacheMemory: 64 MB
```

Getting a Cartridge List via a Cartridge Pool

```
##
## Name:TinaPool [-h|help] [-v|verbose] [-out file]
## Description:References the list of cartridge pools.
## Use:
##
## Arguments:
##-v|verboseMode verbose
##
##-h|help Display the on-line help.
##
##-out file Record the result in the file
## A file is read.
# variables
variable int INT_CT 0
variable int INT_FLAG_HELP 0
variable int INT_VERBOSE 0
variable string STR_FILE_OUT EMPTY_STRING
variable string STR_FILE_SHELL EMPTY_STRING
variable stringlist STR_LIST_RESULTAT " "
##
## Definitions of the generic functions
##
#################################################
# Read a file with beginning and end of line
#
function_begin CatFile INT_DEBUT INT_FIN STR_FILE
  # Reread the file
  fileget STR_FILE STR_LIST_LINES INT_DEBUT INT_FIN
  foreach STR_LINE in STR_LIST_LINES
    echo STR_LINE
  endfor
  if INT_VERBOSE == 1
    echo "Fin affichage fichier"
  endif
function_end
############################
# Argument loop
#
foreach STR_ARGV in ARGV
  # ---------------------------------
  # Results of file name
  if STR_ARGV == "-out"
    variable int INT_CT2 INT_CT
    increment INT_CT2 1
    # Get STR_INFO of STR_FILE_OUT
    variable string STR_FILE_OUT ARGV[INT_CT2
    if INT_VERBOSE == 1
      echo "File Out :" STR_FILE_OUT
    endif
  endif
  # ---------------------------------
  # tina shell file name
  if STR_ARGV == "-file"
    variable int INT_CT2 INT_CT
    increment INT_CT2 1
    # Get STR_INFO of STR_FILE_OUT
    variable string STR_FILE_SHELL ARGV[INT_CT2
    if INT_VERBOSE == 1
      echo "File Shell :" STR_FILE_SHELL
    endif
  endif
  # ---------------------------------
  # Display on-line help
  if STR_ARGV == "-?"
    variable int INT_FLAG_HELP 1
  endif
  # ---------------------------------
  # Display processing of STR_INFO
```

```
  if STR_ARGV == "-v"
    variable int INT_VERBOSE 1
  elif STR_ARGV == "-verbose"
    variable int INT_VERBOSE 1
  endif
  increment INT_CT 1
endfor
# -- ON-LINE HELP---------------------------------------------------------
## Display on-line help if there is an error
if INT_FLAG_HELP == 1
  if INT_VERBOSE == 1
    echo "Display on-line help:" STR_FILE_SHELL
  endif
  function_execute CatFile 0 14 STR_FILE_SHELL
  if INT_VERBOSE == 1
    echo "Exit script :" INT_VERBOSE
  endif
  exit INT_VERBOSE
endif
############################################################
# Main program
# Loop of AccessGroup list: point of entry of processing
list AccessGroup in HDL_LIST_ACC_GR
foreach HDL_ACC_GR in HDL_LIST_ACC_GR
  assign AccessGroupType &INT_ACC_GR_TYPE
  assign AccessGroupName &STR_ACC_GR_NAME
  assign AccessGroupDrive &HDL_LISTACC_GR_DRIVES
  get HDL_ACC_GR
  if INT_ACC_GR_TYPE == 3
    if INT_VERBOSE == 1
      echo EMPTY_STRING
    endif
    assign AccessGroupUser &HDL_LIST_ACC_GR_USERS
    assign AccessGroupPolicy &INT_ACC_GR_POLICY
    get HDL_ACC_GR
    item HDL_LIST_ACC_GR_USERS INT_NB_USERS
    if INT_NB_USERS == 1
      if INT_VERBOSE == 1
        echo "get pool and cartridge label"
      endif
      # Get description of pool and cartridge pool
      assign UserName &STR_USER_NAME
      assign UserCartridge &HDL_LIST_USER_CARTRIDGES
      get HDL_LIST_ACC_GR_USERS[0
      variable string STR_INFO "_____________________________________________________"
      add STR_INFO in STR_LIST_RESULTAT
      variable string STR_INFO "Pool Name"
      variable string STR_INFO2 " : "
      concat STR_INFO2 STR_INFO
      concat STR_USER_NAME STR_INFO
      add STR_INFO in STR_LIST_RESULTAT
      # ---------------------------------------------
      # Loop on drive list associated to the Pool
      if INT_VERBOSE == 1
        echo "get the list of linked tape drive"
      endif
      item HDL_LISTACC_GR_DRIVES nbDrive# Nombre de lecteurs
      variable int INT_CT 0
      if nbDrive == 0
        variable string STR_INFO ">> no linked tape drive"
        add STR_INFO in STR_LIST_RESULTAT
      endif
      foreach leDrive in HDL_LISTACC_GR_DRIVES
        increment INT_CT 1
        assign DriveName &STR_DRIVE_NAME
        get leDrive
        variable string STR_INFO INT_CT
        variable string STR_INFO2 "> "
        concat STR_INFO2 STR_INFO
        concat STR_DRIVE_NAME STR_INFO
```

```
        add STR_INFO in STR_LIST_RESULTAT
      endfor
      # ----------------------------
      # Get the retention
      if INT_ACC_GR_POLICY == 2
        if INT_VERBOSE == 1
          echo "get the pool retention"
        endif
        assign AccessGroupRetUnit &INT_RET_UNIT
        assign AccessGroupRetValue &INT_RET_VALUE
        get HDL_ACC_GR
        if INT_RET_UNIT == 1
          variable int INT_VALUE_FACT 86400
          variable string STR_INFO " Jour(s)"
        elif INT_RET_UNIT == 2
          variable int INT_VALUE_FACT 604800
          variable string STR_INFO " Semaine(s)"
        elif INT_RET_UNIT == 3
          variable int INT_VALUE_FACT 2419200
          variable string STR_INFO " Mois"
        elif INT_RET_UNIT == 4
          variable int INT_VALUE_FACT 125798400
          variable string STR_INFO " Annee(s)"
        else
          variable int INT_VALUE_FACT 0
          variable string STR_INFO EMPTY_STRING
        endif
        variable string info3 " Retention = "
        variable string STR_INFO2 INT_RET_VALUE
        concat STR_INFO2 info3
        concat STR_INFO info3
        add info3 in STR_LIST_RESULTAT
        # Calculate retention in seconds
        variable int RetEnSeconde INT_RET_VALUE
        multiply INT_RET_VALUE INT_VALUE_FACT INT_RESULT
      endif
      # ------------------------------------------
      # Loop the list of Cartridge Pools
      if INT_VERBOSE == 1
        echo "get the catridge pool list"
      endif
      foreach HDL_CART in HDL_LIST_USER_CARTRIDGES
        assign CartridgeDateLastBck &INT_LAST_BACKUP
        assign CartridgeName &STR_CART_NAME
        get HDL_CART
        # Get the current date in seconds
        time INT_CURRENT_DATE
        decrement INT_CURRENT_DATE INT_LAST_BACKUP
        # if INT_CURRENT_DATE < INT_RESULT
        # variable string STR_INFO2 " Recyclable in "
        # else
        # variable string STR_INFO2 " Recyclable since "
        # endif
        variable string STR_INFO " "
        concat STR_CART_NAME STR_INFO
        concat STR_INFO2 STR_INFO
        add STR_INFO in STR_LIST_RESULTAT
      endfor
    endif
  endif
endfor
if INT_VERBOSE == 1
  echo "end of search"
endif
######################
# Outputs result
#
if STR_FILE_OUT != EMPTY_STRING
  # Display not necessary in Delete mode
  fileput STR_FILE_OUT STR_LIST_RESULTAT
```

```
  if INT_VERBOSE == 1
    # Reread the created file
    fileget STR_FILE_OUT STR_LIST_INFOS
    foreach STR_INFO in STR_LIST_INFOS
      echo STR_INFO
    endfor
  endif
else
  foreach STR_INFO in STR_LIST_RESULTAT
    if STR_INFO != " "
      echo STR_INFO
    endif
  endfor
endif
```

This script typically produces an output similar to the following:

```
Pool Name : lost+found
1> diskf1
2> diskf2
_____________________________________________
Pool Name : lab1
1> diskf1
 lab100001>
 lab100002>
 lab100003>
 lab100004>
 lab100005>
_____________________________________________
Pool Name : lab2
1> diskf2
_____________________________________________
Pool Name : spare
1> diskf1
2> diskf2
```

Retrieving DumpCartridgeInformation

```
#-------------------------------------------------------------------------------------------#
# Title: .DumpCartridgeInformation.tsh
# Description: DumpCartridgeInformation
# Use: tina_shell -file DumpCartridgeInformation.tsh
# tina_shell
#-------------------------------------------------------------------------------------------#
variable int ct 0
variable int flagHelp 0
variable string fileShellEMPTY_STRING
variable string PoolNameToUse EMPTY_STRING
variable int OnlyRecyclable 0
variable int IncludePartFilledRecyclable 0
variable int NoOtherInformation 0
variable int MinimiseInformation 0
variable int OnlyOnline 0
variable int Debug 0
variable int IncludeVolume 1 # 1 mean yes, 2 mean yes, in GB
variable int IncludeNbFile 1
variable int IncludeCartridgeType 1
variable int IncludeCartridgeFormat 1
variable int MultiPlyBy100 100
variable int OneGB 10240000
var intlist SecondIn 1 86400 604800 2592000 31536000
var stringlist CartridgeLocationArray "Error" "inside mail box" "in library" "in drive" "outside"
var intlist    CartridgeInLineArray   -1     0                    1             1            0
var string TheHeader1 " Cartridge Name - Bar Code - recyclable or Not - Status - Retention status
- "
concat TheHeader1 TheHeader
var string TheHeader2 " VolumeMB - "
concat TheHeader2 TheHeader
var string TheHeader3 " NbTapeFile - "
concat TheHeader3 TheHeader
var string TheHeader4 " LastBackup - "
concat TheHeader4 TheHeader
var string TheHeader5 " CartType - "
concat TheHeader5 TheHeader
var string TheHeader6 " CartFormat - "
concat TheHeader6 TheHeader
var string TheHeader7 " Location"
concat TheHeader7 TheHeader

echo TheHeader
# Loop on all AccessGroup
list AccessGroup in TheAccessGroupList
foreach OneAccessGroup in TheAccessGroupList
  assign AccessGroupType &leType
  assign AccessGroupName &ThePoolName
  get OneAccessGroup
if leType == 3
    echo "----------------------"
echo " Pool :" ThePoolName
assign AccessGroupUser &lesUser
assign AccessGroupPolicy &TheRecyclingPolicy
get OneAccessGroup

# Retrieve pool label
assign UserName &TheLabel
assign UserCartridge &allCartridge
get lesUser[0
close lesUser[0
# ----------------------------
# Get retention

if TheRecyclingPolicy == 2
```

```
assign AccessGroupRetUnit &RetUnit
assign AccessGroupRetValue &RetValue
get OneAccessGroup
variable int MultiplyFactor SecondIn[RetUnit

# Calculate retention in seconds
multiply RetValue MultiplyFactor RetentionInSeconds
  echo   ==> Pool retention in second :  RetentionInSeconds
# ----------------------------------------
# Loop on cartridge in pool

foreach theCartridge in allCartridge
assign CartridgeDateLastBck &lastBackup
assign CartridgeName &nameCart
assign CartridgeBarCode &TheBARCODE
assign CartridgeLocation &TheCartridgeLocation
assign CartridgeStatus &STATUS
assign CartridgeFileNb &TheFileNb
assign CartridgeVolume &TheVolume
assign CartridgeType &TheCartType
assign CartridgeFormat &TheCartFormat
get theCartridge

variable string infoCartrideFileNb  TheFileNb
variable string infoCartrideVolume  TheVolume
variable int TheVolumeGB TheVolume
multiply TheVolume MultiPlyBy100  TheVolume100
percent TheVolume100 OneGB TheVolumeGB
variable string infoCartrideVolumeGB  TheVolumeGB
variable string infoCartrideName  nameCart
variable string infoCartrideType  TheCartType
variable string infoCartrideFormat  TheCartFormat
date lastBackup lastBackupAsString
# Get the curent date in seconds
time CurrentDate

if lastBackup == 0
  if TheVolume == 0
#if The volume = 0 and The LastBck = 0, we take the LastRecycling
  assign CartridgeDateReused &DateReused
  get theCartridge
  var int lastBackup DateReused
  else
#if The volume != 0 and The LastBck = 0, we are in a strange situation !
# to avoid errors, we consider the LastBackupDate = Now !
  assign CartridgeDateReused &DateReused
  get theCartridge
  var int lastBackup CurrentDate
  endif

endif

                        variable int UseSince CurrentDate
decrement UseSince lastBackup

variable int ReclyclingDate lastBackup
increment ReclyclingDate  RetentionInSeconds

variable int RecyclableIn ReclyclingDate
decrement RecyclableIn CurrentDate

variable int RecyclableSince CurrentDate
decrement RecyclableSince ReclyclingDate

variable int IsRecyclable 2

if STATUS == 1
    variable string infoReclycling " - No  - Filling empty          - not concern by reclycling"
          variable int IsRecyclable 0
elif STATUS == 2
```

```
   variable string infoReclycling " - No  - Partialy filled         - not concern by reclycling"

    if IncludePartFilledRecyclable == 0
          variable int IsRecyclable 0
    else
 if UseSince > RetentionInSeconds
    variable string infoReclycling " - Yes - Partialy filled          - Recyclable since "
    variable string infoReclycling2  RecyclableSince
    variable string infoReclycling3  " seconds"
    concat infoReclycling2 infoReclycling
    concat infoReclycling3 infoReclycling
          variable int IsRecyclable 1
      else
    variable string infoReclycling " - No  - Partialy filled          - recyclable in    "
    variable string infoReclycling2  RecyclableIn
    variable string infoReclycling3  " seconds"
    concat infoReclycling2 infoReclycling
    concat infoReclycling3 infoReclycling
          variable int IsRecyclable 0
      endif

          endif
 elif STATUS == 3
 if UseSince > RetentionInSeconds
    variable string infoReclycling " - Yes - Filling full          - recyclable since "
    variable string infoReclycling2  RecyclableSince
    variable string infoReclycling3  " seconds"
    concat infoReclycling2 infoReclycling
    concat infoReclycling3 infoReclycling
          variable int IsRecyclableRecyclable 1
      else
    variable string infoReclycling " - No  - Filling full             - Recyclable in    "
    variable string infoReclycling2  RecyclableIn
    variable string infoReclycling3  " seconds"
    concat infoReclycling2 infoReclycling
    concat infoReclycling3 infoReclycling
          variable int IsRecyclable 0
      endif
elif STATUS == 4
 if UseSince > RetentionInSeconds
    variable string infoReclycling " - Yes - Closed on error    - Recyclable since "
    variable string infoReclycling2  RecyclableSince
    variable string infoReclycling3  " seconds"
    concat infoReclycling2 infoReclycling
    concat infoReclycling3 infoReclycling
          variable int IsRecyclable 1
      else
    variable string infoReclycling " - No  - Closed on error    - Recyclable in    "
    variable string infoReclycling2  RecyclableIn
    variable string infoReclycling3  " seconds"
    concat infoReclycling2 infoReclycling
    concat infoReclycling3 infoReclycling
     variable int IsRecyclable 0
      endif
 elif STATUS == 5
if UseSince > RetentionInSeconds
    variable string infoReclycling " - Yes - Closed at initialization - Recyclable since "
    variable string infoReclycling2  RecyclableSince
    variable string infoReclycling3  " seconds"
    concat infoReclycling2 infoReclycling
    concat infoReclycling3 infoReclycling
     variable int IsRecyclable 1
      else
    variable string infoReclycling " - No  - Closed at initialization - Recyclable in    "
    variable string infoReclycling2  RecyclableIn
    variable string infoReclycling3  " seconds"
    concat infoReclycling2 infoReclycling
    concat infoReclycling3 infoReclycling
     variable int IsRecyclable 0
      endif
```

```
  elif STATUS == 6
if UseSince > RetentionInSeconds
    variable string infoReclycling " - Yes - Closed by user            - Recyclable since "
    variable string infoReclycling2  RecyclableSince
    variable string infoReclycling3  " seconds"
    concat infoReclycling2 infoReclycling
    concat infoReclycling3 infoReclycling
     variable int IsRecyclable 1
      else
    variable string infoReclycling " - No  - Closed by user           - Recyclable in    "
    variable string infoReclycling2  RecyclableIn
    variable string infoReclycling3  " seconds"
    concat infoReclycling2 infoReclycling
    concat infoReclycling3 infoReclycling
     variable int IsRecyclable 0
      endif
 elif STATUS == 7
    variable string infoReclycling " - No  - Unclose              - not concerned by reclycling"
     variable int IsRecyclable 0
 elif STATUS == 8
    variable string infoReclycling " - No  - emptied              - not concerned by reclycling"
     variable int IsRecyclable 0
 endif

variable string infoLocation CartridgeLocationArray[TheCartridgeLocation

variable int IsOnline CartridgeInLineArray[TheCartridgeLocation
variable string BareCodeInfo " - "
variable string BareCodeInfo2 TheBARCODE
variable string BareCodeInfo3 " "
concat BareCodeInfo2 BareCodeInfo
concat BareCodeInfo3 BareCodeInfo

variable string Message " "
concat infoCartrideName  Message
concat BareCodeInfo Message
concat infoReclycling Message
variable string aSpace " - "

concat aSpace Message
concat infoCartrideVolume  Message
concat aSpace Message
concat infoCartrideFileNb  Message
concat aSpace Message
concat lastBackupAsString  Message
concat aSpace Message
concat infoCartrideType  Message
concat aSpace Message
concat infoCartrideFormat  Message
concat aSpace Message
concat infoLocation Message
        echo Message
    close theCartridge
endfor
else

echo ==> Pool retention is infinite
# here we have identify a infinite pool
# loop on cartridge in pool

foreach TheCartridge in allCartridge
assign CartridgeDateLastBck &lastBackup
assign CartridgeName &nameCart
assign CartridgeBarCode &TheBARCODE
assign CartridgeLocation &TheCartridgeLocation
assign CartridgeStatus &STATUS
assign CartridgeFileNb &TheFileNb
assign CartridgeVolume &TheVolume
assign CartridgeType &TheCartType
assign CartridgeFormat &TheCartFormat
```

```
get TheCartridge

variable string infoCartrideFileNb  TheFileNb
variable string infoCartrideVolume  TheVolume
variable int TheVolumeGB TheVolume
multiply TheVolume MultiPlyBy100  TheVolume100
percent TheVolume100 OneGB TheVolumeGB
variable string infoCartrideVolumeGB  TheVolumeGB
variable string infoCartrideName  nameCart
variable string infoCartrideType  TheCartType
variable string infoCartrideFormat  TheCartFormat
date lastBackup lastBackupAsString

# get the curent date in seconds
time CurrentDate
                        var int UseSince CurrentDate
decrement UseSince lastBackup

if STATUS == 1
    variable string infoReclycling " - No  - Filling empty                - "
elif STATUS == 2
    variable string infoReclycling " - No  - Partialy filled              - "
 elif STATUS == 3
 if UseSince > RetentionInSeconds
    variable string infoReclycling " - No  - Filling full                 - "
      else
    variable string infoReclycling " - No  - Filling full                 - "
      endif
elif STATUS == 4
 if UseSince > RetentionInSeconds
    variable string infoReclycling " - No  - Closed on error              - "
      else
    variable string infoReclycling " - No  - Closed on error              - "
      endif
 elif STATUS == 5
if UseSince > RetentionInSeconds
    variable string infoReclycling " - No - Closed at initialization     - "
      else
    variable string infoReclycling " - No  - Closed at initialization     - "
      endif
  elif STATUS == 6
if UseSince > RetentionInSeconds
    variable string infoReclycling " - No  - Closed by user               - "
      else
    variable string infoReclycling " - No  - Closed by user               - "
      endif
 elif STATUS == 7
     variable string infoReclycling " - No  - Unclose                      - "
 elif STATUS == 8
     variable string infoReclycling " - No  - emptied                      - "
 endif

variable string infoReclycling2  " UseSince "
variable string infoReclycling3  UseSince
variable string infoReclycling4  " seconds"
concat infoReclycling2 infoReclycling
concat infoReclycling3 infoReclycling
concat infoReclycling4 infoReclycling

variable string infoLocation CartridgeLocationArray[TheCartridgeLocation
variable int IsOnline CartridgeInLineArray[TheCartridgeLocation

variable string BareCodeInfo " - "
variable string BareCodeInfo2 TheBARCODE
variable string BareCodeInfo3 " "
concat BareCodeInfo2 BareCodeInfo
concat BareCodeInfo3 BareCodeInfo

variable string Message " "
```

```
concat infoCartrideName  Message
concat BareCodeInfo Message
concat infoReclycling Message

variable string aSpace " - "
concat aSpace Message
concat infoCartrideVolume  Message
concat aSpace Message
concat infoCartrideFileNb  Message

concat aSpace Message
concat lastBackupAsString  Message
concat aSpace Message
concat infoCartrideType  Message
concat aSpace Message
concat infoCartrideFormat  Message
concat aSpace Message
concat infoLocation Message

echo Message
    close TheCartridge
endfor
endif
endif
close OneAccessGroup
endfor
echo "======================"
echo CartridgeFormat
help CartridgeFormat
echo "----------------------"
echo CartridgeType ( in fact DriveType )
help DriveType
echo "----------------------"
```

This script typically produces an output similar to the following:

```
Cartridge Name - Bar Code - recyclable or Not - Status - Retention

status -  VolumeMB -  NbTapeFile -  LastBackup -  CartType -  C

on
----------------------
 Pool : lost+found

==> Pool retention is infinite
----------------------
 Pool : pool1

==> Pool retention is infinite

 lab100001 -   - No - Filling full -  UseSince 501004 seconds - 9 - 1 -

Tue Aug 30 14:41:50 2005 - 3 - 3 - outside

 lab100002 -   - No - Filling full -  UseSince 501004 seconds - 9 - 1 -

Tue Aug 30 14:41:50 2005 - 3 - 3 - outside

 lab100003 -   - No - Filling full -  UseSince 501004 seconds - 9 - 1 -

Tue Aug 30 14:41:50 2005 - 3 - 3 - outside

 lab100004 -   - No - Filling full -  UseSince 501004 seconds - 9 - 1 -

Tue Aug 30 14:41:50 2005 - 3 - 3 - outside
```

```
 lab100005 -   - No - Filling full -  UseSince 496341 seconds - 9 - 2 -
Tue Aug 30 15:59:33 2005 - 3 - 3 - outside
 lab100006 -   - No - Filling full -  UseSince 496342 seconds - 9 - 1 -
Tue Aug 30 15:59:33 2005 - 3 - 3 - outside
 lab100007 -   - No - Filling full -  UseSince 496342 seconds - 9 - 1 -
Tue Aug 30 15:59:33 2005 - 3 - 3 - outside
 lab100008 -   - No - Filling full -  UseSince 496342 seconds - 9 - 1 -
Tue Aug 30 15:59:33 2005 - 3 - 3 - outside
 lab100009 -   - No - Filling full -  UseSince 496342 seconds - 9 - 1 -
Tue Aug 30 15:59:33 2005 - 3 - 3 - outside
 lab100010 -   - No - Filling full -  UseSince 253658 seconds - 9 - 3 -
Fri Sep 02 11:24:17 2005 - 3 - 3 - outside
 lab100011 -   - No - Filling full -  UseSince 253658 seconds - 9 - 1 -
Fri Sep 02 11:24:17 2005 - 3 - 3 - outside
 lab100012 -   - No - Filling full -  UseSince 253658 seconds - 9 - 1 -
Fri Sep 02 11:24:17 2005 - 3 - 3 - outside
 lab100013 -   - No - Filling full -  UseSince 253658 seconds - 9 - 1 -
Fri Sep 02 11:24:17 2005 - 3 - 3 - outside
 lab100014 -   - No - Filling full -  UseSince 253658 seconds - 9 - 1 -
Fri Sep 02 11:24:17 2005 - 3 - 3 - outside
 lab100015 -   - No - Filling full -  UseSince 253658 seconds - 9 - 1 -
Fri Sep 02 11:24:17 2005 - 3 - 3 - outside
 lab100016 -   - No - Filling full -  UseSince 253658 seconds - 9 - 1 -
Fri Sep 02 11:24:17 2005 - 3 - 3 - outside
 lab100017 -   - No - Filling full -  UseSince 253658 seconds - 9 - 1 -
Fri Sep 02 11:24:17 2005 - 3 - 3 - outside
 lab100018 -   - No - Filling full -  UseSince 253658 seconds - 9 - 1 -
Fri Sep 02 11:24:17 2005 - 3 - 3 - outside
 lab100019 -   - No - Filling full -  UseSince 253658 seconds - 9 - 1 -
Fri Sep 02 11:24:17 2005 - 3 - 3 - outside
 lab100020 -   - No - Filling full -  UseSince 253658 seconds - 9 - 1 -
Fri Sep 02 11:24:17 2005 - 3 - 3 - outside
 lab100021 -   - No - Filling full -  UseSince 253658 seconds - 9 - 1 -
Fri Sep 02 11:24:17 2005 - 3 - 3 - outside
 lab100022 -   - No - Filling full -  UseSince 253658 seconds - 9 - 1 -
```

```
Fri Sep 02 11:24:17 2005 - 3 - 3 - outside
 lab100023 -   - No - Filling full -  UseSince 253658 seconds - 9 - 1 -
Fri Sep 02 11:24:17 2005 - 3 - 3 - outside
 lab100024 -   - No - Filling full -  UseSince 253658 seconds - 9 - 1 -
Fri Sep 02 11:24:17 2005 - 3 - 3 - outside
 lab100025 -   - No - Filling full -  UseSince 253658 seconds - 9 - 1 -
Fri Sep 02 11:24:17 2005 - 3 - 3 - outside
 lab100026 -   - No - Filling full -  UseSince 253658 seconds - 9 - 1 -
Fri Sep 02 11:24:17 2005 - 3 - 3 - outside
 lab100027 -   - No - Filling full -  UseSince 253658 seconds - 9 - 1 -
Fri Sep 02 11:24:17 2005 - 3 - 3 - outside
 lab100028 -   - No - Filling full -  UseSince 253658 seconds - 9 - 1 -
Fri Sep 02 11:24:17 2005 - 3 - 3 - outside
 lab100029 -   - No - Filling full -  UseSince 253658 seconds - 9 - 1 -
Fri Sep 02 11:24:17 2005 - 3 - 3 - outside
 lab100030 -   - No - Filling full -  UseSince 253658 seconds - 9 - 1 -
Fri Sep 02 11:24:17 2005 - 3 - 3 - outside
 lab100031 -   - No - Filling full -  UseSince 253658 seconds - 9 - 1 -
Fri Sep 02 11:24:17 2005 - 3 - 3 - outside
 lab100032 -   - No - Filling full -  UseSince 253658 seconds - 9 - 1 -
Fri Sep 02 11:24:17 2005 - 3 - 3 - outside
 lab100033 -   - No - Filling full -  UseSince 253658 seconds - 9 - 1 -
Fri Sep 02 11:24:17 2005 - 3 - 3 - outside
 lab100034 -   - No - Filling full -  UseSince 253659 seconds - 9 - 1 -
Fri Sep 02 11:24:17 2005 - 3 - 3 - outside
 lab100035 -   - No - Filling full -  UseSince 253659 seconds - 9 - 1 -
Fri Sep 02 11:24:17 2005 - 3 - 3 - outside
 lab100036 -   - No - Filling full -  UseSince 253659 seconds - 9 - 1 -
Fri Sep 02 11:24:17 2005 - 3 - 3 - outside
 lab100037 -   - No - Filling full -  UseSince 253659 seconds - 9 - 1 -
Fri Sep 02 11:24:17 2005 - 3 - 3 - outside
 lab100038 -   - No - Filling full -  UseSince 253659 seconds - 9 - 1 -
Fri Sep 02 11:24:17 2005 - 3 - 3 - outside
 lab100039 -   - No - Partialy filled -  UseSince 253659 seconds - 6 - 1
- Fri Sep 02 11:24:17 2005 - 3 - 3 - in drive
```

```
----------------------
 Pool : pool2
==> Pool retention is infinite
----------------------
 Pool : spare
==> Pool retention is infinite
======================
CartridgeFormat
         1 : Format tar
         2 : Format cpio
         3 : Format TiNa
         4 : Format None
         5 : Format Fastrax
         6 : Format Sidf
         7 : Format Unknown
----------------------
CartridgeType ( in fact DriveType )
         1 : Type DAT                        (DAT)
         2 : Type DAT-C                      (DAT-C)
         3 : Type Disk Drive                 (Disk Drive)
         4 : Type DLT 2000                   (DLT 2000)
         5 : Type DLT 4000                   (DLT 4000)
         6 : Type Exabyte 2GB                (EXB 2GB)
         7 : Type Exabyte 2GB-C              (EXB 2GB-C)
         8 : Type Exabyte 5GB                (EXB 5GB)
         9 : Type Exabyte 5GB-C              (EXB 5GB-C)
        10 : Type 3480                       (3480)
        11 : Type Magneto-Optical            (M-Optical)
        12 : Type Mag Tape 6250              (MagTape6250)
        13 : Type Streamer QIC24             (QIC24)
        14 : Type Streamer QIC150            (QIC150)
        15 : Type Streamer QIC525            (QIC525)
        16 : Type CompactTape TK/TZ          (TK/TZ)
        18 : Type Overland 3480              (3480)
        19 : Type Exabyte Mammoth            (Mammoth)
```

```
20 : Type DLT 7000                  (DLT 7000)
21 : Type Tandberg SLR32            (SLR32)
22 : Type Tandberg SLR50            (SLR50)
23 : Type IBM Magstar MP            (Magstar MP)
24 : Type T9840                     (T9840)
25 : Type IBM 3590                  (IBM 3590)
26 : Type Exabyte Mammoth 2         (Mammoth 2)
27 : Type AIT  (35GB)               (AIT)
28 : Type Exabyte VXA-1 (ECRIX VXA) (VXA-1)
29 : Type DLT 8000                  (DLT 8000)
30 : Type Tandberg SLR100           (SLR100)
31 : Type AIT2 (50GB)               (AIT-2)
32 : Type HP Ultrium                (HP Ultrium)
33 : Type IBM Ultrium               (IBM Ultrium)
34 : Type Seagate Ultrium           (Seag Ultrium)
35 : Type T9940                     (T9940)
36 : Type SuperDLT 220              (S-DLT 220)
37 : Type EMC Tape Emulator         (EMC DLU)
38 : Type DLT1                      (DLT1)
39 : Type Sony DTF2                 (Sony DTF2)
40 : Type Quantum DX30              (Quantum DX30)
41 : Type AIT3 (100GB)              (AIT-3)
42 : Type Exabyte VXA-2             (VXA-2)
43 : Type SuperDLT 320              (S-DLT 320)
44 : Type HP Ultrium 2              (HP LTO2)
45 : Type IBM Ultrium 2             (IBM LTO2)
47 : Type SAIT (500GB)              (SAIT (500GB))
48 : Type SuperDLT 600              (S-DLT 600)
49 : Type Tandberg SLR140           (SLR140)
50 : Type UDO                       (UDO)
51 : Type DLT VS160                 (DLT VS160)
52 : Type Centera                   (Centera)
53 : Type HP Ultrium 3              (HP LTO3)
54 : Type IBM Ultrium 3             (IBM LTO3)
55 : Type AIT4 (200GB)              (AIT-4)
```