nirva

How-to: Init and exit of an application

Document version: 1.01

Every Nirva application Nirva contains two files - init.nvp and exit.nvp – that are triggered when an application respectively starts and stops. These two Nirva files can be used when specific initialisation or termination operations are needed, such as:

During the init

- Creation of storage volumes and physical mount points (if they do not exist)
- Retrieval of configuration data from a Nirva registry, a database or a configuration file
- Initialisation of the containers used by the applications
- Creation of named sessions and their context definition.

The init and exit procedures are necessarily in the Nirva native format (nvp extension) but they can obviously call any other procedure written in perl or java:

```
; Do all init in a perl procedure
NV_PROC=|perl:init.pl|
```

These procedures are executed in a special security context giving them all available rights. However, starting and ending Nirva applications is subject to admin security rights so that only authorised users can perform these tasks. The same rights will therefore be automatically carried over to the init and exit procedures.

During the exit

- Housekeeping and cleaning code.

Usage examples

Named sessions

In init.nvp

The creation of a named session and the definition of its context are usually controlled in the init file of the application since the session has a life span equal to the life of the application itself. Here is an example which creates a pool of names sessions for a connection to databases:

init.nvp:

```
; Just call the perl init procedure
NV_PROC=|perl:init|
```

init.pl:

```
# Get number of connections from registry
NV::Command("NV_CMD=|REGISTRY:GET| KEY=|parameters.database| ENTRIES=|database|");
NV::Command("NV_CMD=|OBJECT:INDSTRINGLIST_GET_VALUE| NAME=|database|
KEY=|num_connections|");
$DATA_NUM_CONNECTIONS=$NV::RESULT;

# Open the named sessions
for($i=1;$i<=$DATA_NUM_CONNECTIONS;$i=$i+1)
{
  NV::Command("NV_MD=|SESSION:CREATE| NAME=|$DBSESSION| OPEN=|database/opendb|
CLOSE=|database/closedb|");
}
```

The actual init of the named session itself is performed in the "database/opendb" procedure and the termination code is executed in the "database/closedb" procedure. The two parameters OPEN and CLOSE for the SESSION:CREATE command allow the definition of these procedures. The default values of these parameters are respectively "session_open" and "session_close".

In exit.nvp

In the case of a named session, it is not necessary to execute any termination code in the application init.nvp as this is generally done in the termination procedure of the named session ("database/closedb" in the previous example).

Standard initialisation of an application

In init.nvp

During the initialisation of the application, it is possible to create storage volumes, to create tables and more generally to create objects that will be needed by the application. For instance, the following Nirva commands could be needed:

init.nvp:

```
; Just call the perl init procedure
NV_PROC=|perl:init|
```

init.pl:

```
# Create some directories if they don't exist
NV::Command("NV_CMD=|APPLICATION:CREATE_DIR| DIR=|Input|");
NV::Command("NV_CMD=|APPLICATION:CREATE_DIR| DIR=|Output|");
NV::Command("NV_CMD=|APPLICATION:CREATE_DIR| DIR=|MyWork|");

# Get application directory
NV::Command("NV_CMD=|APPLICATION:GET_DIR|");
$APP_DIR=$NV::RESULT;

# Create storage volume if it doesn’t exist
NV_CMD=|STORAGE:VOLUME:CREATE| NAME=|volume1| NV_NO_ERROR=|YES|

; Create storage level if it doesn’t exist
$DIR=$APP_DIR."Volume/RAMSAY/MAIN";
NV_CMD=|STORAGE:LEVEL:CREATE| NAME=|volume1| LNAME=|level1| ERASABLE=|YES| PATH=|$DIR|
NV_NO_ERROR=|YES|

# Create scheduled tasks if they are not existing
NV::Command("NV_CMD=|SCHEDULER:TASK_LIST|");
NV::Command("NV_CMD=|OBJECT:TABLE_SET_PRIMARY_COLUMN| NAME=|TASK_LIST| COLNAME=|NAME|");

# check mytask task
NV::Command("NV_CMD=|OBJECT:TABLE_GET_CELL_LINE| NAME=|TASK_LIST| PRIMARY=|mytask|
COLNAME=|NAME| NV_NO_ERROR=|YES|");
if($NV::RESULT eq "")
{
  # The task doesn't exist
  # We create it
  NV::Command("NV_CMD=|SCHEDULER:CREATE_TASK| NAME=|mytask| DESCRIPTION=|test|
NV_NO_ERROR=|YES|");
  # First disable it because by default it runs
   NV::Command("NV_CMD=|SCHEDULER:ENABLE_TASK| NAME=|mytask| ENABLE=|NO|
NV_NO_ERROR=|YES|");

  # Set the default parameters
```

```
  NV::Command("NV_CMD=|SCHEDULER:SET_TASK_PARAM| NAME=|mytask| USER=||
PROCT=|perl:input/mytask| PROCI=|session_open| PROCE=|session_close| FREQUENCY=|DAILY|
TIME_SPAN=|86399| REPEAT=|YES| DELAY=|1| OCCURENCES=|0| NV_NO_ERROR=|YES|");
}
```

The NV_NO_ERROR=|YES| parameter is often used, in this particular case to instruct Nirva not to take into account the possible error the command could return. For instance, when creating a storage volume, the command will return an error if the volume already exists. Setting NV_NO_ERROR to "YES" allows the creation of the volume if it does not exist. Obviously, this error control is very simple and may be sometimes sufficient. In other cases, one would have to develop a more thorough error control mechanism..

In exit.nvp

```
exit.nvp:

; Remove all the files and subdirectories of the application MyWork dir

NV_CMD=|APPLICATION:REMOVE_DIR| DIR=|MyWork| SELF=|NO|
```