RPMNative Tool, Version 2.0 Q&A
==========

WARNING: Answers are to the best of our current knowledge; please report any problems and/or errors that you find.

_____

Q1. How does the RPMnativeTool allow access to the Native Mode Power Off info produced by the radio ?

A. [BACKGROUND:
The Native Mode 1.2 Manual (970.0777R1) describes the “Power Off” event on page 4-61, Section 4.9.6 Hardware (HW_EVENT). The event in question is documented in 4.9.6.2:

HW_OFF “RPM is shutting down as a result of the user turning the unit off. Device shutdown is imminent.” The previous quoted defintion is taken directly from the Native Mode manual.

Contact Motorola directly to obtain the Native Mode manual; however we do not believe that it is necessary to have this manual to make effective use of the RPMnativeTool.]

This information is available thru your EventHandler slot of your NS endpoint; see RPMnativeTool doc. for details.

_____

Q2. In the newer 2.0 development version of the tool,if the radio is off and then turn it back on WITHOUT cycling the Newton’s PCMCIA door switch, I never see a NS “on” event ?

A. The current version “RPMnativeTool.pkg1.2d4”, which is a development version for 2.0 OS, does not yet support this operation; the older formally released 1.x version had to do some really wierd things to make this work which were OS version dependent. Please stand by; in the mean time, just cycle the PCMCIA door switch.

_____

Q3.I really need compatability with protoBasicEndpoint. The new functionality solves lots of other problems. Does the RPMnativeTool work with this new NS proto ? [The docs. show proto’ing from the older protoEndpoint ?!?]

A. [including background info]

The tool was developed on 1.x initially. It was ported as closely as possible .

Several developers had successfully used the new 2.0 protoBasicEndpoint; we await the results of their testing. At the CommTool interface level, there is no significant difference in the API; all the NS-visible changes in 2.0, e.g. protoBasicEndpoint, are implemented above the CommTool API. Thus, the tool remains as before, and may be accessed via either proto, “protoEndpoint” or “basicProtoEndpoint”.

_____

Q4.The documentation I received with the 2.0 development version of the tool seems to be just like the 1.x docs ?

A. As you might have guessed the documentation was written for the 1.x version and has not been updated. The truth is nothing has changed from the developer's perspective.

_____

Q5. I have exisiting NS code that uses the RPMnativeTool via the 1.x protoEndpoint, does OS 2.0 version support ?

A. Yes, 2.0 still preserves backward compatibility with protoEndPoint. If you have s/w already works with the old endpoint (and thus represents a great model of how to use the n/s endpoint and the nativetool) , using this as a starting point might be a good way to minimize risk.

_____

Q6. Does theTPMnativeTool work with an external DataTAC radio.

A. No support for this was removed prior to the official release of the 1.x-timeframe tool.

_____

Q7. How feasible is it to move some of the code (eg the stuff which talks to the card handler) in your open/close/closeComplete methods into bind/unbind? - so we wouldn't receive events until after the bind call.

A. It is a feature that you get events after instantiate. This allows the newton script programs to be event driven.

_____

Q8. How do I use the RPMnativeTool under the old 1.x protoEndpoint, in the most controlled fashion ?

A. TO OPEN TOOL (so that you can get events, esp. the radio on event when the PCMCIA door switch is closed)

```
ep:Instantiate()          //note you do not have to do a ep:justOpen here;
                    // this is automatically done by the TCommToolService:Start method
```

TO CONNECT TOOL (do this after you get radio on event):

```
ep:SetSync(true)
ep:justBind()
ep:JustConnect()
ep:SetSync(nil)
```

TO DISCONNECT TOOL

ep:FlushOutput(); //i think this is still necessary in 2.0
<adddeferred action>(....ep:Disconnect() .....)

TO CLOSE TOOL

<adddeferred action>(....ep:Dispose() .....)

If you get a radio off event you must disconnect, then close, and then reopen the tool to monitor further events; note under1.x you were able to skip the close-reopen step because the ROM endpoint code didn't enforce states as well as it does in 2.0's protoEndpoint

———————

Q9. We believe that RPMnative endpoint is to be used on Newton 2.0 system. In that case, why does the RPMnative endpoint proto from protoEndPoint which is essentially a Newton 1.X feature?

A. This is almost as above in Q5; the tool doesn't know about or implement the protoEndpoint or the protoBasicEndpoint behavior.

———————

Q10. We have received the RPMnativeTool.pkg1.2d4 package but do not know what exactly to do with it. It appears to be an AutoPart type of package and thus tapping on its icon does nothing. How do we use its services? We could not find any document that explains this.

A. Write a NS program using the Comm endpoint facility.

———————

Q11. I tried to use the tool and got a -24006. Help !

A. The package is a card handler/driver. It lays claim to the PM100D when it is installed. It controlls the very eccentric power modes of the devcice, and facilitates data movement to/from the device.

I suspect the cause of the -24006 to be that you did not have a PM100D installed, with the latch closed. The -24006 is essentially that the tool got no response.

———————

Q12. Using NTK ,we made Project settings to various values like (Newton 2.0, use 1.0 build rules, setting platform to 2.0 as well as message Pad, etc.) but to no avail. What Project settings are to be set specifically for this?

A. Nothing special is required; make sure you have the latest NTK for 2.0, not a Beta version.

———————

Q13. What is -54014 error ?

A. It means NS endpoint not available.

———————

Q14. Is the "MG" peer-to-peer messaging record format for the driver the same as the ARDIS spec. "MG" & "SSSSSSSS/" & "DDDDDDDD/" & unicodeCR etc? If it is not the same then what is required? The docs you sent show an example that doesn't match the spec without any explanation of the fields.

A. The driver is not involved with the MG format at all; however the 1st byte of your message, i.e. before any Ardis/MG stuff must be the RPM Native Command Lanaguage's "send options byte"[See Moto RPM NCL doc; if you don't have here's a summary: bit 0 (oxo1) is the "resend" flag; bits 2-1 (mask 0x06) are the "priority" and bit 6 (0x40) must always be set and is the "send options byte tag".]

Also the "example" is poor and was written before we knew anything about the ARDIS spec — please ignore the format of the example; it is meant to show how to do multiple NewtonScript "Output"s to form a frame . That's all.

———————

Q15. What is exception |evt.ex.comm| (10) weird ranged Error Code that I get from the driver when I try to send something with the Standard MG Record Format for Peer-to-Peer Messaging. Is the a list of these low number exceptions with the RPM driver? I've also gotten (11) too; what do they mean ?

A . Sorry; the 1.x driver was released without the error base being assigned and this changed from 0; we are consulting with the 1.x users who are moving to 2.0 re changing the base to some nice big negative number.

from the C++: (for tool versions prior to 6Feb96)

```
enum eRPM_RESULT
{
keRPM_OK=0,
keRPM_TOO_MANY_OUTPUT,
keRPM_FRAMED_OUTPUT_TOO_BIG,
keRPM_FRAMED_INPUT_TOO_BIG,
keRPM_FRAMED_INPUT_BADLY_FORMED_SDU,
keRPM_OUT_OF_MEM,
keRPM_UNEXPECTED_EVENT_SDU_RCVD,
keRPM_ORPHAN_RESP_SDU_RCVD,
keRPM_UNRETRYABLE_FAILURE,
keRPM_LOW_BAT,
keRPM_OUT_OF_RANGE,
keRPM_FAILURE_AFTER_RETRY,
keRPM_RADIO_DETECTED_SYNTAX_ERROR,
keRPM_UNKNOWN_RADIO_RESULT,
keRPM_INTERNAL_TOOL_ERR,
keRPM_RADIO_OFF
};
```

ATTENTION : Expanded error codes for versions of tool > 1.2d10 ^ Feb 96); in particular, the list of errors was expanded starting at 16. Errors that were previously 8, or "keRPM_UNRETRYABLE_FAILURE", will now be mapped to the new numbers, to provide more details.

from the C++ :

```
enum eRPM_RESULT
{       keRPM_OK=0,
keRPM_TOO_MANY_OUTPUT,
keRPM_FRAMED_OUTPUT_TOO_BIG,
keRPM_FRAMED_INPUT_TOO_BIG,
keRPM_FRAMED_INPUT_BADLY_FORMED_SDU,
keRPM_OUT_OF_MEM,
//5
keRPM_UNEXPECTED_EVENT_SDU_RCVD,
keRPM_ORPHAN_RESP_SDU_RCVD,
keRPM_UNRETRYABLE_FAILURE,
keRPM_LOW_BAT,
keRPM_OUT_OF_RANGE,
//10
keRPM_FAILURE_AFTER_RETRY,
keRPM_RADIO_DETECTED_SYNTAX_ERROR,
keRPM_UNKNOWN_RADIO_RESULT,
keRPM_INTERNAL_TOOL_ERR,
keRPM_RADIO_OFF

//15
,
keRPM_XFAIL_NO_RSP,
//16
keRPM_XFAIL_NO_ACK,
keRPM_XFAIL_HOST_DOWN,
keRPM_XFAIL_NOT_REG,
keRPM_XFAIL_IBQ_FULL,
//20
keRPM_XFAIL_TX_DISABLED,
keRPM_XFAIL_BUSY,
keRPM_XFAIL_NOT_AVAIL,
keRPM_XFAIL_HW_ERR,
keRPM_XFAIL_INVALID_MODE,
//25
keRPM_XFAIL_SW_ERR,
```

These codes are direct reflections of the actual NativeMode XFAIL result values which caused the operation to fail.

Q16. I haven’t been able to get either the battery level or radio signal strength events to work. I do get other events from the driver with my eventHandler() and when I do an :Option() method call for the current event parameters I get back the booleans that say that I’ve turned on the correct parameters. I was wondering if these events are implemented? I set the options from the :instantiate() method if that makes any difference.

A. Moto may have changed the radio signal level location in their NCL proprietary status page in the US ARDIS version of the radio; in the RDLAP Eagle card this was the only “good” signal strength.

[UPDATE 23apr96 Signal strength now relialbly functions in current version of tool.]

Additionally, concerning battery levels, the “Eagle” and the ARDIS PCMCIA radios simply have a bat. good and bat. very low indication that we can get in driver software; we map “good” to 90% and “very bad” to 10% of the [min/max] range the driver reports in. The old Infotac “brick” radios had a good “analog” battery indication information.

Also, it doesn’t matter how/when your set the options.

———————

Q17. I want to use NewtonScript Virtual Binary Objects (VBOs) in my NS program, does the RPMnativeTool work with them ? I understand that I can probably use driver as it exists, true?

A. VBOs are handled in the OS ROM’s endpoint/NS mechanisms. The RPMnativeTool, or any CommTool for that matter, is not “influenced” by your choice of NS data representations.

———————

Q18. A developer states/asks:

“We have to handle all error conditions.
We need to know what kind of error codes come when we try to do various calls under various circumstances. .....[in] like instantiate, connect, setInputSpec, and output, etc. we have seen how they [these calls] behave when everything is OK. But how do they behave when say the card is :

(1) unlocked
(2) outOfRange
(3) locked but powered off
(4) Any other conditions you can think of

They should either return an error synchronously in rc which we can then check OR they should throw up an exception which we can then catch OR they should invoke the exceptionHandler method that we have coded within the endPoint definition.”

A. Well this is a two-part question; let’s answer the second part 1st: Regarding the presentation of errors to your NS application, the RPMnativeTool has no control over how these error data are actually returned to NS apps. A good app should expect any of the above 3 methods (i.e. “synchronously in rc” , “throw up an exception....catch”, and “invoke the exceptionHandler”) based on particular Comm Scripting usages and/or timing and loading of the Newton.

The actual error values received via these methods may include RPMnative error values as outlined in Q15, and also Comm Scripting and Endpoiint error codes. A NS app should assume that any non-0 error code is bad.

Highly probable error codes(depends on
Comm Scripting, e.g. async vs. sync, Newton load, etc.) when the card is :

(1) unlocked
keRPM_RADIO_OFF
(2) outOfRange
keRPM_OUT_OF_RANGE
(3) locked but powered off keRPM_RADIO_OFF