RMR Route Tables

###DEPRECATED###

This page has been deprecated in favour of the documentation maintained within the RMR repository and published on the RTD site:

https://docs.o-ran-sc.org/projects/o-ran-sc-ric-plt-lib-rmr/en/latest/index.html

2020 May 12

~~RMR expects to receive routing information in the form of a *route table* from a manager process. The route information can be delivered as a complete table, or as updates to the table. For testing, a static route table can be supplied to an RMR based application, and as a developer knowing how to create such a static table might be useful. The following paragraphs describe the RMR route table.~~

~~Table Syntax~~

~~The table consists of a start record, one or more table entry records, and an end record. Each entry record defines one message type, with an optional sender application, and the endpoint(s) which accept the indicated message type. All table records contain fields separated with vertical bars (|), and allow for trailing comments with the standard shell comment symbol (hash, #) provided that the start of the comment is separated from the last token on the record by one or more spaces. Leading, and trailing, white space in each field is ignored.~~

~~Table Entry Syntax~~

~~Two types of table entries are supported for compatibility with the original RMR implementation, but only the MSE entry type is needed and that will be the only entry described here. (Existing test tables with rte entry types should be converted as support for that type might be deprecated soon.) The following shows the MSE (message type, sub-id, entry) format:~~

```
~~mse | <msg type>[,<sender endpoint>] | <sub id> | <endpoint group>[;<endpoint group>;...]~~
```

~~Where:~~

~~**mse** - is the table entry type~~

~~**<msg type>** - is the integer message type~~

~~**<sender endpoint>** - is the endpoint description of the message sender; only that sender will read the entry from the table, so a single table may be used for all senders when a common message type is delivered to varying endpoints based on senders. If the sender endpoint is omitted from the entry, then the entry will be used by all applications which read the table.~~

~~**<sub id>** - is the subscription id (integer) for subscription based messages, or -1 if the message type is not subscription based~~

~~**<endpoint group>** - is one or more, comma separated, endpoint descriptions. When an application sends a message with the indicated type, the message will be sent to one endpoint in the group in a round-robin ordering. If multiple endpoint groups are given, then the message is sent to a member selected from each group; 3 groups, then three messages will be sent. The first group is required.~~

~~Endpoint Description~~

~~The endpoint description is either the hostname or IP address followed by a port number; the two are separated by a single colon. Whether names or IP addresses are used in static tables during testing depends on the test environment. The illustration below assumes that host names (e.g. forwarder and app1) are defined; they also make the tables easier to read, but might not always be available depending on the environment used to test the applications. The port number given is the port number that the user application provides to RMR when the RMR initialisation function is invoked (and thus is the port that RMR is listening on).~~

~~A Complete Table~~

~~The following is a complete table to illustrate the previously discussed syntax.~~

```
newrt|start | id000345
mse| 1000 | 10 | forwarder:43086
mse| 1000,forwarder:43086 | 10 | app2:43086
mse| 1000 | -1 | app0:43086,app1:43086;logger:20311  # rr to app0/app1 and always send to logger
newrt|end
```

The order of message types in the table is not important. However, for a single message type, the order of entries in the route table is important; more specific entries (e.g. ones with both type and sender) should be after the related more generic (e.g. sender omitted) entries. To that end, the first entry will be recognised and added by all applications. When the `forwarder` application reads the second entry it will replace the first (also for message type 1000) as it sees itself as the sender. All other applications will ignore the second entry as their endpoint information does not match.

For the first two entries, a subscription ID of 10 is defined, and thus messages which set both a message type 1000 and the subscription ID of 10 will select an endpoint from the entry retained by the application. When a message is sent with the message type of 1000, but without setting the subscription ID, the last entry in the table will be used. This entry will also cause two messages to be sent: one will be sent to either app0 or app1 from the first round-robin group, and the second will always be sent to the logger.

Table ID on Start Record

The final field on the newrt|start record (id000345 in the example above) is used when RMR sends acknowledges to the Route Manager. Each ack is sent with the table ID provided in order for the Route Manager to track state of each update delivered. If omitted, RMR will return a default string similar to "<id-missing>".

Providing A Static Table

Normally, when the RMR initialisation function is invoked, a listener is started to receive route table information from a route manager process. During testing it is often useful to be able to supply a static table which is available should no route management process exist, or to provide a seed table to use before the first table is delivered. The environment variable `RMR_SEED_RT` can be set to provide the RMR initialisation function with the name of the static table to use. If a static table is provided, it will be loaded only once, and if a route manager subsequently connects and delivers a complete or updated table, it will override the static table.

Routing Using MEID

Starting with version 1.13.0, RMR provides the ability to select the endpoint for a message based on the MEID (managed entity ID) in the message, rather than selecting the endpoint from the round robin list for the matching route table entry. To make use of this, there must be one or more route table entries which list the special endpoint name **%meid** instead of providing a round robin list. As an example, consider the following route table entry:

```
mse| 1000,forwarder:43086 | 10 | %meid
```

The final field of the entry doesn't specify a round-robin group which means that when an application attempts to send a message with type 1000, and the subscription ID of 10, the MEID in the message will be used to select the endpoint.

MEID endpoint selection

To select an endpoint for the message based on the MEID in a message, RMR must know which endpoint *owns* the MEID. This information, known as an MEID map, is provided by the Routing Manager over the same communication path as the route table is supplied. The following is the syntax for an MEID map.

```
meid_map | start | <table-id>
mme_ar | <owner-endpoint> | <meid> [<meid>...]
mme_del | <meid> [<meid>...]
meid_map | end | <count> [| <md5sum> ]
```

The mme_ar records are add/update records and allow for the list of MEIDs to be associated with (owned by) the indicated endpoint. The <owner-endpoint> is the hostname:port, or IP address and port, of the application which *owns* the MEID and thus should receive any messages which are routed based on a route table entry with *%meid* as the round-robin group. The mme_del records allow for MEIDs to be deleted from RMR's view. Finally, the <count> is the number of add/replace and delete records which were sent; if RMR does not match the <count> value to the number of records, then it will not add the data to the table. Updates only need to list the ownership changes that are necessary; in other words, the Route Manager does **not** need to supply all of the MEID relationships with each update.

The optional <md5sum> field on the end record should be the MD5 hash of all of the records between the start and end records. This allows for a more precise verification that the transmitted data was correctly received.

If a static seed file is being used for the route table, a second section can be given which supplies the MEID map. The following is a small example of a seed file:

```
newrt|start | id-64306
        mse|0|-1| %meid
        mse|1|-1|172.19.0.2:4560
        mse|2|-1|172.19.0.2:4560
        mse|3|-1|172.19.0.2:4560
        mse|4|-1|172.19.0.2:4560
        mse|5|-1|172.19.0.2:4560
newrt|end

meid_map | start | id-028919
        mme_ar| 172.19.0.2:4560 | meid000 meid001 meid002 meid003 meid004 meid005
        mme_ar| 172.19.0.42:4560 | meid100 meid101 meid102 meid103
        mme_del | meid1000
meid_map | end | 1
```

The tables above will route all messages with a message type of 0 based on the MEID. There are 10 meids which are owned by two different endpoints. The table also deletes the MEID meid1000 from RMR's view.

Reserved Message Types

RMR is currently reserving message types in the range of 0 through 99 (inclusive) for its own use. Please do not use these types in any production or test environment as the results may be undesired.