4.4.2.10 MRAPI_RMEM_READ_I

NAME

mrapi_rmem_read_i

SYNOPSIS

```
void mrapi_rmem_read_i(
  MRAPI_IN mrapi_rmem_hndl_t rmem,
  MRAPI_IN mrapi_uint32_t rmem_offset,
  MRAPI_OUT void* local_buf,
  MRAPI_IN mrapi_uint32_t local_offset,
  MRAPI_IN mrapi_uint32_t bytes_per_access,
  MRAPI_IN mrapi_uint32_t num_strides,
  MRAPI_IN mrapi_uint32_t rmem_stride,
  MRAPI_IN mrapi_uint32_t local_stride,
  MRAPI_OUT mrapi_request_t* mrapi_request,
  MRAPI_OUT mrapi_status_t* status
);
```

DESCRIPTION

This (non-blocking) function performs `num_strides` memory reads, where each read is of size `bytes_per_access` bytes. The *i*-th read copies `bytes_per_access` bytes of data from `rmem` with offset `rmem_offset + i*rmem_stride` to `local_buf` with offset `local_offset + i*local_stride`, where 0 <= *i* < `num_strides`.

This supports scatter/gather type accesses. To perform a single read, without the need for scatter/gather, set the `num_strides` parameter to 1.

RETURN VALUE

On success, `*status` is set to `MRAPI_SUCCESS`. On error, `*status` is set to the appropriate error defined below. Use `mrapi_test()`, `mrapi_wait()` or `mrapi_wait_any()` to test for completion of the operation.

ERRORS

`MRAPI_ERR_RMEM_INVALID`	Argument is not a valid remote memory segment handle.
`MRAPI_ERR_RMEM_BUFF_OVERRUN`	`rmem_offset + (rmem_stride * num_strides )` would fall out of bounds of the remote memory buffer.
`MRAPI_ERR_RMEM_STRIDE`	`num_strides`>1 and `rmem_stride` and/or `local_stride` are less than `bytes_per_access`.
`MRAPI_ERR_REQUEST_LIMIT`	`No more request handles available.`
`MRAPI_ERR_RMEM_NOTATTACHED`	The caller is not attached to the remote memory.
`MRAPI_ERR_RMEM_BLOCKED`	We have hit a hardware limit of the number of asynchronous DMA/cache operations that can be pending ("in flight") simultaneously. Thus we now have to block because the resource is busy.
`MRAPI_ERR_PARAMETER`	Either the `local_buf` is invalid or the `buf_size` is zero or `bytes_per_access` is zero.
`MRAPI_ERR_NODE_NOTINIT`	The calling node is not intialized.