MPI_REDUCE_SCATTER(sendbuf, recvbuf, recvcounts, datatype, op, comm)

IN	sendbuf	starting address of send buffer (choice)
OUT	recvbuf	starting address of receive buffer (choice)
IN	recvcounts	non-negative integer array (of length group size) specifying the number of elements [in]of the result distributed to each process. [Array must be identical on all calling processes.]
IN	datatype	data type of elements of [input buffer]send and receive buffers (handle)
IN	op	operation (handle)
IN	comm	communicator (handle)

ticket93.
ticket124.
ticket124.
ticket124.

```
int MPI_Reduce_scatter(void* sendbuf, void* recvbuf, int *recvcounts,
             MPI_Datatype datatype, MPI_Op op, MPI_Comm comm)

MPI_REDUCE_SCATTER(SENDBUF, RECVBUF, RECVCOUNTS, DATATYPE, OP, COMM,
             IERROR)
    <type> SENDBUF(*), RECVBUF(*)
    INTEGER RECVCOUNTS(*), DATATYPE, OP, COMM, IERROR
```

ticket150.

```
{void MPI::Comm::Reduce_scatter(const void* sendbuf, void* recvbuf,
             int recvcounts[], const MPI::Datatype& datatype,
             const MPI::Op& op) const = 0 (binding deprecated, see Section ??) }
```

ticket150.

ticket124. If comm is an intracommunicator, MPI_REDUCE_SCATTER first [does an element-wise reduction on vector of count $= \sum_i$ recvcounts[i] elements in the send buffer defined by sendbuf, count and datatype. Next, the resulting vector of results is split into n disjoint segments, where n is the number of members in the group. Segment i contains recvcounts[i] elements. The i-th segment]performs a global, element-wise reduction on vectors of count $= \sum_{i=0}^{n-1}$ recvcounts[i] elements in the send buffers defined by sendbuf, count and datatype, using the operation op, where n is the number of processes in the group of comm. The routine is called by all group members using the same arguments for recvcounts, datatype, op and comm. The resulting vector is treated as n consecutive blocks where the number of elements of the i-th block is recvcounts[i]. The blocks are scattered to the processes of the group. The i-th block is sent to process i and stored in the receive buffer defined by recvbuf, recvcounts[i] and datatype.

Advice to implementors. The MPI_REDUCE_SCATTER routine is functionally equivalent to: an MPI_REDUCE collective operation with count equal to the sum of recvcounts[i] followed by MPI_SCATTERV with sendcounts equal to recvcounts. However, a direct implementation may run faster. (*End of advice to implementors.*)

The "in place" option for intracommunicators is specified by passing MPI_IN_PLACE in the sendbuf argument. In this case, the input data is taken from the [top of the receive]receive buffer. It is not required to specify the "in place" option on all processes, since the processes for which recvcounts[i]==0 may not have allocated a receive buffer.

ticket91.
ticket124.

If comm is an intercommunicator, then the result of the reduction of the data provided by processes in [group A]one group (group A) is scattered among processes in [group B]the other group (group B), and vice versa. Within each group, all processes provide the same recvcounts argument, and [the sum of the recvcounts entries should]provide input vectors of count $= \sum_{i=0}^{n-1}$ recvcounts[i] elements stored in the send buffers, where n is the size of the group. The resulting vector from the other group is scattered in blocks of recvcounts[i] elements among the processes in the group. The number of elements count must be the same for the two groups. ticket124. ticket124. ticket124.

Rationale. The last restriction is needed so that the length of the send buffer can be determined by the sum of the local recvcounts entries. Otherwise, a communication is needed to figure out how many elements are reduced. (*End of rationale.*)

5.11 Scan

5.11.1 Inclusive Scan

MPI_SCAN(sendbuf, recvbuf, count, datatype, op, comm)

IN	sendbuf	starting address of send buffer (choice)
OUT	recvbuf	starting address of receive buffer (choice)
IN	count	number of elements in input buffer (non-negative integer)
IN	datatype	data type of elements of input buffer (handle)
IN	op	operation (handle)
IN	comm	communicator (handle)

```
int MPI_Scan(void* sendbuf, void* recvbuf, int count,
             MPI_Datatype datatype, MPI_Op op, MPI_Comm comm )
```

```
MPI_SCAN(SENDBUF, RECVBUF, COUNT, DATATYPE, OP, COMM, IERROR)
    <type> SENDBUF(*), RECVBUF(*)
    INTEGER COUNT, DATATYPE, OP, COMM, IERROR
```

ticket150.

```
{void MPI::Intracomm::Scan(const void* sendbuf, void* recvbuf, int count,
             const MPI::Datatype& datatype, const MPI::Op& op) const
             (binding deprecated, see Section ??) }
```

ticket150.

If comm is an intracommunicator, MPI_SCAN is used to perform a prefix reduction on data distributed across the group. The operation returns, in the receive buffer of the process with rank i, the reduction of the values in the send buffers of processes with ranks 0,...,i (inclusive). The type of operations supported, their semantics, and the constraints on send and receive buffers are as for MPI_REDUCE.

The "in place" option for intracommunicators is specified by passing MPI_IN_PLACE in the sendbuf argument. In this case, the input data is taken from the receive buffer, and replaced by the output data.

This operation is invalid for intercommunicators.