Name: H5Pset_nbit

Signature:

herr_t `H5Pset_nbit`(*hid_t* `plist_id`)

Purpose:

Sets up use of the N-Bit filter.

Description:

`H5Pset_nbit` sets the N-Bit filter, `H5Z_FILTER_NBIT`, in the dataset creation property list `plist_id`.

The N-Bit filter is used effectively for compressing data of N-Bit datatype as well as compound and array datatype with N-Bit fields. It also supports complex situations where a compound datatype contains member(s) of compound datatype or an array datatype has compound datatype as the base type.

The user of HDF5 library can create an N-Bit datatype by writing codes like:

```
hid_t nbit_datatype = H5Tcopy(H5T_STD_I32LE);
H5Tset_precision(nbit_datatype, 16);
H5Tset_offset(nbit_datatype, 4);
```

In memory, one value of this example N-Bit datatype will be stored on a little-endian machine like this:

byte 3	byte 2	byte 1	byte 0
????????	????SPPP	PPPPPPPP	PPPP????

Note: S - sign bit, P - significant bit, ? - padding bit,
For signed integer, the sign bit is included in the precision

When data of above datatype are stored on disk using N-bit filter, all padding bits are chopped off and only significant bits are stored. So, values on disk will something like:

1st value	2nd value	
SPPPPPPPPPPPPPPP	SPPPPPPPPPPPPPPP	...

The N-Bit filter currently supports N-Bit datatype or N-Bit datatype field with its datatype class being integer or floating-point. It is expected to support other datatype fields such as string intervened between N-Bit fields of a compound datatype in the near future.

Like other I/O filters supported by the HDF5 library, application using the N-Bit filter must store data with chuncked storage.

It makes no sense to use N-Bit filter together with other I/O filters.

Parameters:

hid_t `plist_id`

IN: Dataset creation property list identifier.

Returns:

Returns a non-negative value if successful; otherwise returns a negative value.