Test Plan for LAG Functionality

The primary objective of this test plan is to validate the functionality of LAG (Link Aggregation Group) and LAG members, ensuring the following:

Initialization and Setup requirements:

- Up to 5 LAGs,
- Up to 16 LAG_MEMBERs,
- Up to 32 PORTs,
- Supported LAG attributes: PORT_LIST,
- Supported LAG_MEMBER attributes: PORT_ID, LAG_ID.

Key Aspects Tested:

- **LAG Creation**: Verify that LAGs can be created up to a maximum of 5.
- **LAG Member Creation**: Verify that LAG members can be created up to 16 for each LAG and correctly associate PORT_ID and LAG_ID attributes.
- **LAG and LAG Member Removal**: Ensure that LAG and LAG members can be removed, following proper rules for removal and checking for dependencies (i.e., no LAG members should remain when removing a LAG).
- **Attribute Handling**: Validate proper attribute processing, including PORT_LIST, LAG_ID, and PORT_ID for both LAG and LAG members.
- **Get Operations**: Ensure the get operation for both LAG and LAG members works as expected and returns the correct attributes.
- **Edge Case Testing**: Ensure the edge cases like exceeding the maximum LAG or LAG member count are handled appropriately.
- **Mandatory Attributes**: Verifying that all mandatory attributes are present for both LAGs and LAG_MEMBERs.
- **LAG-MEMBER Association**: Ensuring that LAG_MEMBERs are correctly linked to LAGs and that PORTs in PORT_LIST are valid.
- **LAG Removal**: Preventing LAG removal if there are LAG_MEMBERs associated with it.

- **Maximum Limits**: Ensuring the system handles the maximum number of LAGs, LAG_MEMBERs, and PORTs.

- **Error Handling**: Validating that improper inputs (like invalid PORT_IDs or LAG_IDs) are rejected.

Test Cases

1. LAG Creation Test

- **Objective**: Verify LAG creation up to a maximum of 5 LAGs.

- **Steps**:

    1. Create a LAG using the create_lag function.

    2. Ensure the LAG is created successfully by checking the return status.

    3. Verify that the LAG has no members initially (PORT_LIST should be empty).

    4. Repeat the process until 5 LAGs are created.

    5. Attempt to create the 6th LAG, and verify that the system returns a failure status due to the maximum LAG limit.

- **Expected Result**: LAGs 1 to 5 should be created successfully, but the 6th LAG creation should fail.

2. LAG Member Creation Test

- **Objective**: Verify LAG member creation for each LAG and ensure proper association of PORT_ID and LAG_ID.

Steps:

1. Create a LAG
2. Create 2 to 16 LAG members and associate each member with a valid PORT_ID and LAG_ID.

3. Verify that each member is successfully created and that the PORT_ID and LAG_ID are correctly assigned to each LAG member.

4. Check that each LAG member is listed in the LAG's PORT_LIST.

- **Expected Result**: LAG members should be created successfully, and each member should be correctly associated with the LAG.

3. LAG Member Removal Test

- **Objective**: Ensure LAG members can be removed correctly.
- **Steps**:
    1. Create a LAG and add LAG members.
    2. Remove a LAG member and verify that it is removed from the LAG's PORT_LIST.
    3. Verify that the LAG member is successfully removed by checking the status.
- **Expected Result**: LAG members should be removed successfully, and the LAG's PORT_LIST should reflect the changes.

4. LAG Removal with Members Test

- **Objective**: Ensure that a LAG can only be removed if it has no LAG members associated with it.
- **Steps**:
    1. Create a LAG with LAG members.
    2. Attempt to remove the LAG while it still has members.
    3. Ensure that the system returns a failure status for the removal attempt.
- **Expected Result**: The LAG should not be removed if it still has LAG members. The system should return an appropriate failure message.

5. LAG Member Removal and Check for Orphaned Members

- **Objective**: Ensure that removing a LAG member does not leave orphaned references to the member.
- **Steps**:
    1. Create a LAG and add several LAG members.
    2. Remove a LAG member.

3. Verify that the member is no longer present in the LAG's member list and that no orphaned data remains.

- **Expected Result**: The LAG member should be fully removed from both the member list and the LAG.

6. Get LAG Attributes Test

- **Objective**: Verify that the get operation correctly retrieves the PORT_LIST for a LAG.
- **Steps**:
    1. Create a LAG with multiple LAG members.
    2. Perform a get operation to retrieve the LAG's attributes.
    3. Check that the returned PORT_LIST contains the correct PORT_IDs for the members.
- **Expected Result**: The PORT_LIST returned by the get operation should match the actual PORT_IDs associated with the LAG members.

7. Get LAG Member Attributes Test

- **Objective**: Verify that the get operation correctly retrieves the LAG_ID and PORT_ID for a LAG member.
- **Steps**:
    1. Create a LAG and add LAG members.
    2. Perform a get operation to retrieve the LAG_ID and PORT_ID attributes for a specific LAG member.
    3. Verify that the retrieved LAG_ID and PORT_ID match the expected values.
- Expected Result: The LAG_ID and PORT_ID returned by the get operation should match the actual values assigned to the LAG member.

8. Edge Case: Maximum LAG and Member Count

- **Objective**: Verify the behavior when attempting to create more than the maximum allowed LAGs or LAG members.
- **Steps**:

1. Create 5 LAGs.

2. Attempt to create the 6th LAG and verify that the creation fails.

3. Create 16 LAG members for each LAG.

4. Attempt to create the 17th LAG member for any LAG and verify that the creation fails.

- **Expected Result**: Creating the 6th LAG or the 17th LAG member should fail, as they exceed the limits.

9. Invalid Attribute Test

- **Objective**: Ensure that the system rejects invalid or unsupported attributes.

- **Steps**:

  1. Attempt to create a LAG and LAG member with an invalid attribute (LAG ID, PORT ID).
  2. Creating LAGs without any attributes (should be allowed as PORT_LIST is not mandatory on creation).
  3. Attempt to create LAG members without mandatory attributes.
  4. Retrieving attributes of non-existent LAGs and LAG members

  5. Verify that the system returns an error status indicating the invalid attribute.

- **Expected Result**: The system should reject invalid attributes and return a failure status.

10. Edge Case: Removal error handling

Objective: Ensure that the system returns an appropriate failure message when trying to remove a non-existent LAG and LAG member

- **Steps**:

  1. Attempt to delete a non-existent LAG.
  2. Attempt to delete a non-existent LAG member.

- **Expected Result**: The LAG and LAG member should not be removed as they do not exist. The system should return an appropriate failure message.

Additional Considerations

- **Boundary Testing**: Test the limits of the number of LAGs and LAG members, especially at the edge of the allowed limits (5 LAGs and 16 members per LAG).

- **Error Handling**: Ensure that all error cases (such as invalid input or exceeding limits) are handled gracefully and return appropriate status codes.

To Run This Test

- **Save:** Save the code above as a .c file (e.g., test_lag.c).
- **Compilation**: The compilation command might look something like:
  Bash
  ```
  gcc -o unit_test1 -I /usr/include/sai -L /usr/lib test_lag.c -lsai
  ```
- **Execution**: Run the compiled executable:

Conclusion

This test plan covers a wide range of functionality and edge cases to ensure that the LAG and LAG member implementation meets the required specifications. Each test case targets a specific feature, ensuring that the system behaves as expected across various scenarios.