```swift
import Foundation

/// Handle to the state buffers.
///
/// A stateful model maintains a state from one prediction to another by storing the information in the state buffers. To use such a model, the client must request the model to create state buffers and get `MLState` object, which is the handle to those buffers. Then, at the prediction time, pass the `MLState` object in one of the stateful prediction functions.
///
/// ```swift
/// // Load a stateful model
/// let modelAsset = try MLModelAsset(url: modelURL)
/// let model = try await MLModel.load(asset: modelAsset, configuration: MLModelConfiguration())
///
/// // Request a state
/// let state = model.newState()
///
/// // Run predictions
/// for _ in 0 ..< 42 {
///   _ = try await model.prediction(from: inputFeatures, using: state)
/// }
///
/// // Access the state buffer.
/// state.withMultiArray(for: "accumulator") { stateMultiArray in
///   ...
/// }
/// ```
///
/// The object is a handle to the state buffers. The client shall not read or write the buffers while a prediction is in-flight.
///
/// Each stateful prediction that uses the same `MLState` must be serialized. Otherwise, if two such predictions run concurrently, the behavior is undefined.
///
@available macOS 15.0
public class MLState  NSObject  @unchecked Sendable


@available macOS 15.0  iOS 18.0  watchOS 11.0  tvOS 18.0
extension MLState

    @available                          "Use withMultiArray(for:) instead."
    public func withMultiArray R  _       MLMultiArray    R throws   R
```

```
@available macOS 15.0  iOS 18.0  watchOS 11.0  tvOS 18.0

public func withMultiArray R                String  _
   MLMultiArray  throws    R  rethrows    R
```