// Copyright 2013 The Flutter Authors. All rights reserved.

// Use of this source code is governed by a BSD-style license that can be

// found in the LICENSE file.

#ifndef FLUTTER\_SHELL\_PLATFORM\_COMMON\_CLIENT\_WRAPPER\_INCLUDE\_FLUTTER\_EVENT\_SINK\_H\_

#define FLUTTER\_SHELL\_PLATFORM\_COMMON\_CLIENT\_WRAPPER\_INCLUDE\_FLUTTER\_EVENT\_SINK\_H\_

namespace flutter {

class EncodableValue;

// Event callback. Events to be sent to Flutter application

// act as clients of this interface for sending events.

template <typename T = EncodableValue>

class EventSink {

public:

EventSink() = default;

virtual ~EventSink() = default;

// Prevent copying.

EventSink(EventSink const&) = delete;

EventSink& operator=(EventSink const&) = delete;

// Consumes a successful event

void Success(const T& event) { SuccessInternal(&event); }

// Consumes a successful event.

void Success() { SuccessInternal(nullptr); }

// Consumes an error event.

void Error(const std::string& error\_code,

const std::string& error\_message,

const T& error\_details) {

ErrorInternal(error\_code, error\_message, &error\_details);

}

// Consumes an error event.

void Error(const std::string& error\_code,

const std::string& error\_message = "") {

ErrorInternal(error\_code, error\_message, nullptr);

}

// Consumes end of stream. Ensuing calls to Success() or

// Error(), if any, are ignored.

void EndOfStream() { EndOfStreamInternal(); }

protected:

// Implementation of the public interface, to be provided by subclasses.

virtual void SuccessInternal(const T\* event = nullptr) = 0;

// Implementation of the public interface, to be provided by subclasses.

virtual void ErrorInternal(const std::string& error\_code,

const std::string& error\_message,

const T\* error\_details) = 0;

// Implementation of the public interface, to be provided by subclasses.

virtual void EndOfStreamInternal() = 0;

};

} // namespace flutter

#endif // FLUTTER\_SHELL\_PLATFORM\_COMMON\_CLIENT\_WRAPPER\_INCLUDE\_FLUTTER\_EVENT\_SINK\_H\_