// 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\_BINARY\_MESSENGER\_H\_

#define FLUTTER\_SHELL\_PLATFORM\_COMMON\_CLIENT\_WRAPPER\_INCLUDE\_FLUTTER\_BINARY\_MESSENGER\_H\_

#include <functional>

#include <string>

namespace flutter {

// A binary message reply callback.

//

// Used for submitting a binary reply back to a Flutter message sender.

typedef std::function<void(const uint8\_t\* reply, size\_t reply\_size)>

BinaryReply;

// A message handler callback.

//

// Used for receiving messages from Flutter and providing an asynchronous reply.

typedef std::function<

void(const uint8\_t\* message, size\_t message\_size, BinaryReply reply)>

BinaryMessageHandler;

// A protocol for a class that handles communication of binary data on named

// channels to and from the Flutter engine.

class BinaryMessenger {

public:

virtual ~BinaryMessenger() = default;

// Sends a binary message to the Flutter engine on the specified channel.

//

// If |reply| is provided, it will be called back with the response from the

// engine.

virtual void Send(const std::string& channel,

const uint8\_t\* message,

size\_t message\_size,

BinaryReply reply = nullptr) const = 0;

// Registers a message handler for incoming binary messages from the Flutter

// side on the specified channel.

//

// Replaces any existing handler. Provide a null handler to unregister the

// existing handler.

virtual void SetMessageHandler(const std::string& channel,

BinaryMessageHandler handler) = 0;

};

} // namespace flutter

#endif // FLUTTER\_SHELL\_PLATFORM\_COMMON\_CLIENT\_WRAPPER\_INCLUDE\_FLUTTER\_BINARY\_MESSENGER\_H\_