// 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\_MESSAGE\_CODEC\_H\_

#define FLUTTER\_SHELL\_PLATFORM\_COMMON\_CLIENT\_WRAPPER\_INCLUDE\_FLUTTER\_MESSAGE\_CODEC\_H\_

#include <memory>

#include <string>

#include <vector>

namespace flutter {

// Translates between a binary message and higher-level method call and

// response/error objects.

template <typename T>

class MessageCodec {

public:

MessageCodec() = default;

virtual ~MessageCodec() = default;

// Prevent copying.

MessageCodec(MessageCodec<T> const&) = delete;

MessageCodec& operator=(MessageCodec<T> const&) = delete;

// Returns the message encoded in |binary\_message|, or nullptr if it cannot be

// decoded by this codec.

std::unique\_ptr<T> DecodeMessage(const uint8\_t\* binary\_message,

const size\_t message\_size) const {

return std::move(DecodeMessageInternal(binary\_message, message\_size));

}

// Returns the message encoded in |binary\_message|, or nullptr if it cannot be

// decoded by this codec.

std::unique\_ptr<T> DecodeMessage(

const std::vector<uint8\_t>& binary\_message) const {

size\_t size = binary\_message.size();

const uint8\_t\* data = size > 0 ? &binary\_message[0] : nullptr;

return std::move(DecodeMessageInternal(data, size));

}

// Returns a binary encoding of the given |message|, or nullptr if the

// message cannot be serialized by this codec.

std::unique\_ptr<std::vector<uint8\_t>> EncodeMessage(const T& message) const {

return std::move(EncodeMessageInternal(message));

}

protected:

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

virtual std::unique\_ptr<T> DecodeMessageInternal(

const uint8\_t\* binary\_message,

const size\_t message\_size) const = 0;

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

virtual std::unique\_ptr<std::vector<uint8\_t>> EncodeMessageInternal(

const T& message) const = 0;

};

} // namespace flutter

#endif // FLUTTER\_SHELL\_PLATFORM\_COMMON\_CLIENT\_WRAPPER\_INCLUDE\_FLUTTER\_MESSAGE\_CODEC\_H\_