# SUATMM. Эмулятор процессинга банка и демо-приложение, демонстрирующее работу API процессинга.

Решение включает в себя три части:

1. SUATMM\_Server.exe. Эмулятор процессинга (серверная часть)
2. SuatmmApi.dll. Библиотека, содержащая API для вызова сервисов процессинга, а также разделяемые серверной и клиентской частью форматы сообщений, сериализаторы и общие проверки.
3. SUATMM\_Demo.exe. Демо-приложение, демонстрирующее вызовы API из библиотеки

Кроме того, в решение включены юнит-тесты на основе xUnit

Целевой версией .NET выбрана версия 4.0, как наиболее распространенная (за исключением проекта с тестами, версия которого 4.6).

## SUATMM\_Server. Эмулятор процессинга (серверная часть)

Серверная часть выполнена как самостоятельное приложение, не требующее сторонних веб-серверов для работы (для простоты демонстрации).

Включает в себя:

1. SuatmmServer - непосредственно http сервер, прослушивающий соединение
2. Processing - обработчики сообщений для различных версий форматов сообщений (о версиях см. ниже)
3. SuatmmDatabase - эмулятор базы данных, содержит сведения о тестовых картах, а также об обработанных заказах.

Поступающие запросы HTTP десериализуются в зависимости от ContentType в пакеты сообщений. Пакет сообщения имеет атрибут Version указывающий на версию формата сообщений в пакете. В соответствии с версией подбирается обработчик Processing (в самом решении представлена только одна версия v1). Таким образом достигается обратная совместимость - можно обновлять серверную часть добавляя новые форматы не опасаясь, что клиенты, не обновившие свое ПО не смогут работать.

В обработчике в зависимости от вида сообщения (платеж, запрос статуса или запрос возврата средств) вызывается соответствующая функция для обработки. Перед выполнением каких-либо действий выполняется ряд проверок. Проверки представляют из себя объекты классов-наследников от абстрактного класса Checkup<TIn, TOut>, где

TIn - класс проверяемого объекта (в данном случае сообщения)

TOut - класс объекта, в который записываются результаты проверки (ответное сообщение)

Класс Checkup содержит делегаты OnSuccess и OnFail, которые вызываются соответственно если объект прошел проверку или не прошел. Сама логика проверки задается в переопределяемом методе Invoke.

Объекты классов-наследников Checkup указываются в списке и при обработке сообщений последовательно вызываются для проверки корректности сообщения.

Такой подход дает следующие преимущества:

1. Декларативный характер указания проверок. Мы не пишем непосредственно в коде обработчика что с чем сверять, а указываем какая проверка должна выполняться, и какой код ошибки она устанавливает если сообщение не проходит проверки.
2. Повторное использование кода. Одна и так же проверка Checkup включается в списки для проверки различных видов сообщений. В данном примере — это проверка номера заказа, так как этот параметр имеется во всех трех формах сообщений.
3. Возможность управления проверками во время выполнения. Например, в случае нештатной ситуации часть проверок может быть отключена или наоборот включена без остановки сервера.
4. Готовность к распараллеливанию. Проверки в списке могут быть разбиты на несколько меньших списков и выполняться параллельно.

В зависимости от успешности проверок в ответ возвращается сообщение OperationStatus, содержащее либо код подтверждения успешности операции, либо код запрашиваемого статуса, либо код ошибки.

## SUATMM\_Lib. Библиотека, содержащая API для вызова сервисов процессинга

Библиотека включает в себя:

1. Форматы сообщений (Formats). В примере реализована только версия форматов v1
2. Сериализаторы (Serialize). В примере реализован только XML ввиду доступности включенных в поставку .NET средств для работы с xml.
3. Проверки (Checkups), разделяемые между серверной и клиентской частью
4. Транспорты (Transport) - средства для доставки сообщений после сериализации. В примере реализован только транспорт для доставки сообщений по протоколу HTTP.
5. Непосредственно API для вызова из прикладного кода (SuatmmExchange)

При вызове API выполняются действия:

1. Проверка входящих данных
2. Формируется пакет с сообщением соответствующей формы (платеж, запрос статуса или запрос возврата) и формата (по умолчанию версия v1)
3. Пакет сериализуется в зависимости от ContentType (по умолчанию application/xml)
4. Сериализованный пакет отправляется посредством транспорта
5. Полученный в ответ пакет десериализуется, в пакете ожидается сообщение OperationStatus
6. Из сообщения OperationStatus извлекается код результата, который и возвращается в качестве ответа вызова API

## SUATMM\_Demo. Демо-приложение, демонстрирующее вызовы API из библиотеки

Просто консольное приложение, вызывающее API из библиотеки и отображающее результат вызова. Для удобства ввода уже включает в себя данные тестовых карт и карт с неверными параметрами.