|  |
| --- |
|  |
| Библиотека базового класса WCF-клиента ALX.Common.WCF |
| Документация к сборке |

|  |
| --- |
| Сорочан Алексей Александрович  11.1.2023 |

Оглавление

[Общие сведения 3](#_Toc124415120)

[Структура сборки 3](#_Toc124415121)

[Enums 4](#_Toc124415122)

[Extensions 4](#_Toc124415123)

[ServiceClientBase 4](#_Toc124415124)

[ServiceClientWrapper 5](#_Toc124415125)

[SimpleOutMessageHeader 5](#_Toc124415126)

[Приложение №1 0](#_Toc124415127)

[Клиент WCF-службы 0](#_Toc124415128)

[Вызов клиента WCF-службы 4](#_Toc124415129)

# Общие сведения

Сборка представляет собой набор классов, которые позволяют создать клиент WCF-службы и упрощают взаимодействие со службой. Позволяет обращаться к методам службы посредством лямбда-выражения из клиентского приложения, в котором добавлена ссылка на WCF-службу. Для использования возможностей сборки необходимо добавить в проект клиентского приложения промежуточный класс, который не потребует дальнейшего сопровождения при изменении WCF-контракта.

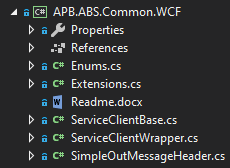
При использовании функционала, который предоставляет данная сборка, следует учитывать особенности, связанные с вызовом WCF-службы. К примеру, тот факт, что нет возможности сконфигурировать подключение к службе из запущенного экземпляра клиентского приложения. То есть настройки подключения и ссылку на WCF-службу возможно указать только в файле конфигурации (\*.config), настройки которого применяются при запуске приложения.

Некоторая часть документации к сборке отражена в комментариях к исходным кодам.

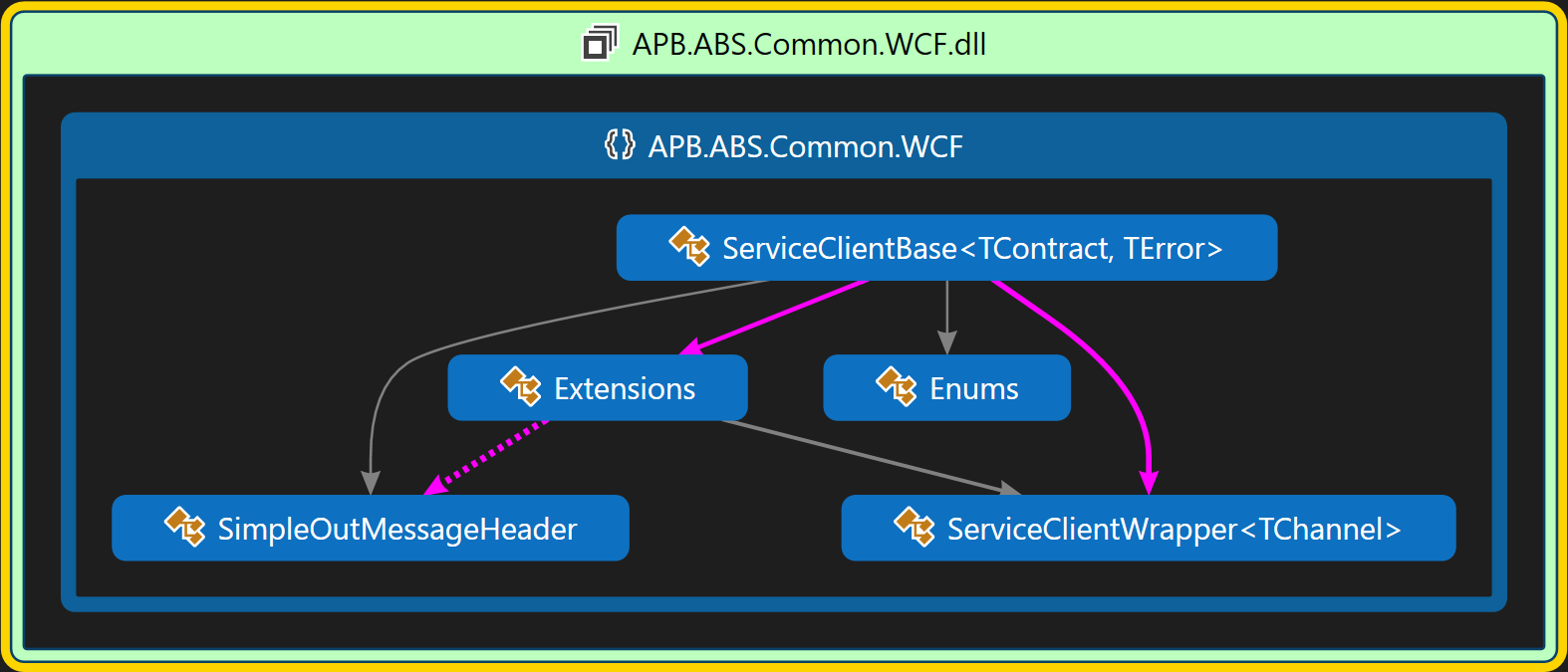
Пример создания клиента WCF-службы с использованием сборки ALX.Common.WCF, приведен в [приложении №1](#_Приложение_№1).

# Структура сборки

В состав сборки входит ряд классов, часть из которых являются вспомогательными и предназначены только для внутреннего использования в рамках сборки.



На рисунке ниже представлена карта исходного кода библиотеки классов ALX.Common.WCF.



Каждый из перечисленных классов будет более подробно рассмотрен ниже.

## Enums

Класс, в котором представлены перечисления, использующиеся с сборке, предназначенных для внутреннего использования. В частности, перечисление «LoginMethod», в нем представлены доступные способы аутентификации для взаимодействия с WCF-службой:

1. NoAuth – вызов WCF без аутентификации;
2. Certificate – аутентификация по сертификату клиента;
3. UserName – аутентификация по имени пользователя и паролю.

## Extensions

Набор методов-расширений, предназначенных для внутреннего использования. Представлен следующими методами:

1. SetServiceCertificate – позволяет указать сертификат сервера для экземпляра WCF-службы.
2. SetOutGoingMessageHeaders – позволяет указать заголовки для исходящих сообщений.

## ServiceClientBase

Базовый класс клиента WCF-службы, является родительским классом, для которого в обязательном порядке, при наследовании, необходимо указать обобщенные типы:

* TContract – контракт WCF-сервиса;
* TError – класс ошибки WCF-сервиса (наследник System.Exception), к примеру BusinessLogicException, который генерируется исключительно на стороне конкретной WCF-службы. Если WCF не генерирует такие исключения, то для обобщенного типа TError можно указать любого другого наследника класса System.Exception.

В базовом классе приведен полный набор свойств для всех способов аутентификации, таких как: сертификат сервера и клиента, имя пользователя и пароль, имя привязки (BindingName) для каждого из доступных [способов аутентификации](#_Enums).

Отдельно следует отметить некоторые методы класса:

* CreateServiceClient – позволяет создать клиент WCF-службы, по указанным контракту, способу аутентификации и привязке.
* ServiceExceptionFilter – метод, который выполняет фильтрацию (парсинг) исключений, сгенерированных клиентом WCF-службы. К примеру, метод возвращает более понятное толкование ошибки тайм-аута подключения или ошибки безопасности при проверке сообщения.

Также, из особенностей этого метода следует отметить то, что он рекурсивно разбирает сообщение об ошибке, до последнего InnerException. Таким образом в клиентское приложение будет возвращаться полный текст сообщения.

* InvokeWcf – метод-обёртка, для вызова функции WCF-службы. Метод принимает лямбда-выражение вызова WCF-службы и список заголовков исходящего сообщения. Возвращает объект (или перечисление объектов) результирующего типа данных, для функции указанной в лямбда-выражении вызова.

Заголовки исходящих сообщения указываются в рамках контекста текущего вызова, при помощи метода-расширения [рассмотренного выше](#_Extensions).

* CallService – делегат вызова WCF-службы, который необходимо реализовать при наследовании, на стороне клиентского приложения. Наиболее важная часть базового класса, реализация которого позволяет вызывать WCF-службу при помощи лямбда-выражения на стороне клиентского приложения. Метод принимает лямбда-выражение, которое в последствии будет вызвано в методе InvokeWcf. Пример реализации будет рассмотрен ниже, [в соответствующем разделе документа](#_Пример_использования).

## ServiceClientWrapper

Класс-обертка над базовым клиентом WCF-службы, наследуется от класса [System.ServiceModel.ClientBase](https://learn.microsoft.com/en-us/dotnet/api/system.servicemodel.clientbase-1?view=dotnet-plat-ext-7.0). Предназначен только для внутреннего использования. Для класса в обязательном порядке необходимо указать обобщенный тип контракта WCF-службы (TChannel).

Из особенностей использования этого класса, следует отметить тот факт, что в обязательном порядке необходимо указать привязку (BindingName), который описан в config-файле клиентского приложения и будет использоваться в контексте вызова WCF-службы.

## SimpleOutMessageHeader

Класс представляет набор полей для заголовка исходящего сообщения вызова WCF-службы. Создан исключительно для использования в рамках базового класса [ServiceClientBase](#_ServiceClientBase). При необходимости, на стороне клиентского приложения можно указать коллекцию заголовков, содержащих дополнительные параметры контекста вызова.

Поля класса представлены следующим набором параметров:

1. NameSpace – Пространство имен заголовка;
2. Name – Имя заголовка;
3. Value – Значение заголовка.

# Приложение №1

## Клиент WCF-службы

Ниже представлен исходный код, класса клиента WCF-службы.

|  |
| --- |
| namespace ClientApplication.ServiceHelper  {      /// <summary>      /// Клиент WCF-службы (singleton)      /// </summary>      public class ServiceClient : ServiceClientBase<IServiceContract, BusinessLogicException>      {          protected override string NonAuthBindingName { get; }          protected override string AuthByUserNameBindingName { get; }          protected override string AuthByCertificateBindingName { get; }          internal List<SimpleOutMessageHeader> ServiceHeaders = new List<SimpleOutMessageHeader>();          private static ServiceClient \_serviceClient;          /// <summary>          /// Клиент WCF-службы          /// </summary>          /// <param name="args">Аргументы клиента WCF-службы</param>          protected ServiceClient(ServiceClientArgs args)          {              NonAuthBindingName = string.Empty;              AuthByUserNameBindingName = "WSHttpBindingUserName\_IServiceContract";              AuthByCertificateBindingName = "WSHttpBindingCertificate\_IServiceContract";              LoginMethod = args.LoginMethod;              ServiceCertificate = args.ServiceCertificate;              switch (LoginMethod)              {                  case LoginMethods.Certificate: ClientCertificate = args.ClientCertificate ??                      throw new WarningException("Сертификат клиента не указан"); break;                  case LoginMethods.UserName:                      UserName = string.IsNullOrEmpty(args.UserName)  ? throw new WarningException("Имя пользователя не указано")  : args.UserName;                      Password = string.IsNullOrEmpty(args.Password)  ? throw new WarningException("Пароль не указан")  : args.Password;                      break;                  default:                      throw new ArgumentOutOfRangeException(                          paramName:nameof(LoginMethod),                          actualValue:args.LoginMethod,                          message:@"Способ аутентификации указан некорректно.");              }          }          /// <summary>          /// Инициализация          /// </summary>          /// <param name="args">Аргументы клиента WCF-служба</param>          public static ServiceClient Initialize(ServiceClientArgs args)          {              return \_serviceClient ?? (\_serviceClient = new ServiceClient(args));          }          /// <summary>          /// Выполнить вызов WCF сервиса          /// </summary>          /// <typeparam name="TResult">Тип возвращаемого значения</typeparam>          /// <param name="invocation">Лямбда выражения вызова</param>          public override TResult CallService<TResult>(Func<IServiceContract, TResult> invocation)          {              return InvokeWcf(invocation, ServiceHeaders);          }          /// <summary>          /// Добавить заголовок вызова службы          /// </summary>          /// <param name="name">Имя</param>          /// <param name="value">Значение</param>          public void ServiceMessageHeaderAdd(string name, string value)          {              ServiceHeaders.Add(new SimpleOutMessageHeader(nameSpace:"MyService", name:name, value:value));          }          /// <summary>          /// Чтение параметров типизированого исключения WCF-службы          /// </summary>          /// <param name="error">Типизированое исключение</param>          protected override string ParsServiceFaultException(BusinessLogicException error)          {              return error.DisplayText;          }      }      /// <summary>      /// Аргументы клиента WCF-службы      /// </summary>      public class ServiceClientArgs      {          /// <summary>          /// Способ аутентификации          /// </summary>          public LoginMethods LoginMethod { get; }          /// <summary>          /// Имя пользователя          /// </summary>          public string UserName { get; }          /// <summary>          /// Пароль          /// </summary>          public string Password { get; }          /// <summary>          /// Сертификат пользователя          /// </summary>          public X509Certificate2 ClientCertificate { get; }          /// <summary>          /// Сертификат сервиса          /// </summary>          public X509Certificate2 ServiceCertificate { get; }          /// <summary>          /// Аргументы клиента WCF-службы          /// </summary>          /// <param name=”serviceCertificate”>Сертификат сервиса</param>          /// <param name=”loginMethod”>Способ аутентификации</param>          /// <param name=”sername”>Имя пользователя</param>          /// <param name=”password”>Пароль</param>          /// <param name=”clientCertificate”>Сертификат пользователя</param>          public ServiceClientArgs(X509Certificate2 serviceCertificate, LoginMethods loginMethod,  string sername = “”, string password = “”, X509Certificate2 clientCertificate = null)          {              ServiceCertificate = serviceCertificate;              LoginMethod = loginMethod;              UserName = sername;              Password = password;              ClientCertificate = clientCertificate;          }      }  } |

## Вызов клиента WCF-службы

Ниже представлен исходный код вызова метода WCF-службы, через класс наследник базового класса ServiceClientBase.

|  |
| --- |
| namespace ClientApplication.MainForm  {      // Клиент WCF-службы      private readonly ServiceClient \_serviceClient;      public MainForm()      {          X509Certificate2              serviceCertificate = new X509Certificate2(),              clientCertificate = new X509Certificate2();          // Инициализация клиента WCF-службы          \_serviceClient = ServiceClient.Initialize(new ServiceClientArgs(                      serviceCertificate: serviceCertificate,                      loginMethod: LoginMethods.Certificate,                      userName: string.Empty, // необходимо указать только в случае LoginMethods.UserName                      password: string.Empty, // необходимо указать только в случае LoginMethods.UserName                      clientCertificate: clientCertificate));      }      private void buttonTest\_Click(object sender, EventArgs e)      {          // Вызов метода WCF-службы          Client[] clients = \_serviceClient.CallService(c => c.GetClients());          gridCtrlClients.DataSource = clients;      }  } |