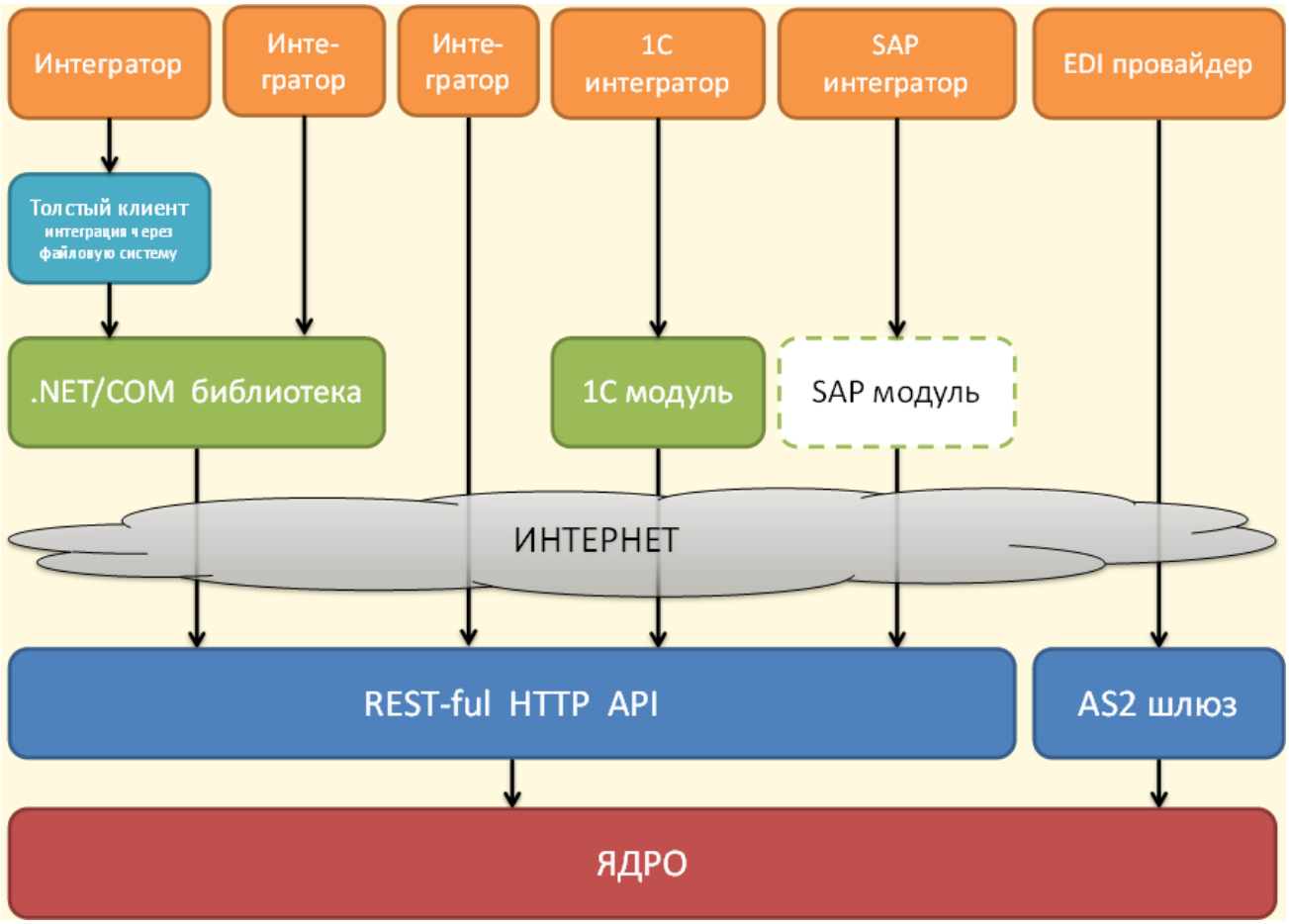
Возможности для интеграции

На схеме ниже представлен весь спектр возможностей для интеграции с Диадоком:



В зависимости от решаемых прикладных задач, а также в зависимости от требований к клиентской части и доступных средств разработки интегратор может выбрать наиболее подходящий ему уровень взаимодействия с API Диадока.

Базовым уровнем интеграции с Диадоком является его HTTP-based API. Этот уровень является наиболее общим, и на нем, в частности, обеспечивается платформо-независимость для интеграционных решений. Это значит, что с HTTP API могут работать как клиенты, написанные на языке C# под платформу .NET и запускающиеся на машинах с ОС Microsoft Windows, так и клиенты, написанные на Java или C++, запускающиеся на машинах под управлением ОС Linux.

Для интеграторов, ведущих разработку в стеке технологий Microsoft, доступна клиентская .NET/COM-библиотека DiadocApiDll, которая берет на себя работу по преобразованию локальных обращений к свойствам и методам некоторых объектов в соответствующие HTTP-команды API Диадока.

Кроме того, DiadocApi.dll инкапсулирует детали работы с криптографией, так что прикладному разработчику не нужно разбираться с тонкостями обращений к CryptoAPI. Актуальная версия библиотеки DiadocApi.dll вместе с исходниками входит в состав SDK, который доступен для скачивания по ссылке.

Для разработчиков, занимающихся интеграцией Диадока с различными программными продкутами, построенными на платформе 1С, доступен специальный внешний компонент, который позволяет максимально быстро решать типовые задачи, возникающие при стыковке 1С-решений с Диадоком. Этот компонент написан на языке C++ и не содержит лишних зависимостей, что позволяет использовать его на любой Windows-системе, не требуя установки каких-либо дополнительных модулей.

Кроме того, для EDI-провайдеров и организаций, имеющих системы, поддерживающие работу по **AS2-протоколу**, Диадок предоставляет возможность интеграции через AS2-шлюз. AS2-шлюз позволяет, например, конвертировать EDI-сообщения типа INVOIC в Диадок-сообщения, содержащие XML счета-фактуры, передаваемые в соответствие с порядком Минфина.

Наконец, программная реализация API Диадока доступна для следующих языков (входит в состав SDK): Java и C++, а из языков Visual Basic for Applications и JavaScript возможна работа с Диадоком посредством COM-сервисов (примеры также доступны в составе SDK).

.NET/COM-библиотека

При разработке интеграторских решений можно использовать клиентскую библиотеку DiadocApi.dll, которая берет на себя работу по преобразованию локальных вызовов методов определенных объектов в соответствующие HTTP-команды API Диадока. Кроме того, DiadocApi.dll инкапсулирует детали работы с криптографией, так что прикладному разработчику не нужно разбираться с тонкостями взаимодействия с CryptoAPI.

Библиотека написана на языке C# на платформе .NET Framework и распространяется бесплатно вместе с исходным кодом в составе Diadoc SDK.

Также в составе SDK есть примеры использования библиотеки DiadocApi.dll.

Точкой входа для .NET-разработчиков является объект DiadocApi, большая часть методов которого представляет собой обертки методов HTTP API Диадока.

Кроме того, у этого объекта есть методы, позволяющие при необходимости задать имя пользователя и пароль для работы через прокси-сервер.

Также библиотека DiadocApi.dll включает в себя объект ComDiadocApi, представляющий собой обертку объекта DiadocApi, доступную для обращений через инфраструктуру COM.

Данный COM-компонент может, например, использоваться в скриптах на языке JavaScript или VBScript.

Модель данных

Диадок предназначен для обеспечения юридически значимого документооборота между организациями. Соответственно, основными задачами Диадока являются:

- маршрутизация документов между организациями и/или подразделениями организаций;

- формирование и хранение цепочек документооборота (документ формируется, затем подписывается одной стороной, передается на подпись второй стороне, подписывается второй стороной, и т.п.);

Единицей маршрутизации в Диадоке являются Ящики (*Box*).

То есть когда один пользователь системы хочет начать документооборот со своим контрагентом, он формирует цепочку документооборота (сообщение из документа и подписи к нему) в своем ящике (ящик отправителя), а система выполняет доставку документа контрагенту путем формирования соответствующей цепочки документооборота в ящике контрагента (ящике получателя).

Этот процесс в некотором смысле аналогичен доставке писем через электронную почту с той разницей, что реально документы никуда не пересылаются, а в ящиках отправителя и получателя лишь формируются записи, позволяющие добраться до соответствующих документов.

Каждый ящик в Диадоке относится к некоторой организации (юридическое лицо или индивидуальный предприниматель), которая представляется в Диадоке в виде единственной сущности типа Организация (Organization).

Ящики в системе закрепляются за организациями таким образом, что каждый ящик принадлежит одной и только одной организации. За счет этого, например, обмен бухгалтерскими документами и обмен договорами с одной компанией могут замыкаться на различные отделы и различных людей внутри этой компании, так что соответствующие бизнес-процессы будут автоматизироваться независимо.

Пользователи Диадока могут аутентифицироваться в системе как по паре логин/пароль, так и при помощи X.509-сертификата.

Разграничение доступа аутентифицированных пользователей Диадока производится на уровне ящиков. То есть к одному ящику могут получить доступ несколько пользователей, и наоборот, один пользователь может получить доступ сразу к нескольким ящикам.

Доступ пользователя к ящику подразумевает возможность просматривать имеющиеся в нем документы и отправлять из него документы в ящики контрагентов.

Цепочки документооборота представляются в Диадоке Сообщениями (Message).

Таким образом, ящик представляет собой логический контейнер для хранения всех сообщений, как отправленных из этого ящика, так и полученных в этот ящик.

Содержимое ящика можно воспринимать как список, в котором исходящие сообщения перечислены вперемешку с входящими.

Соответственно, чтобы адресовать конкретное сообщение в Диадоке, нужны два уникальных идентификатора:

- идентификатор ящика;
- идентификатор сообщения внутри ящика.

Сообщение служит логической единицей группировки документов, относящихся к одной бизнес-транзакции (к одной цепочке документооборота).

Например, в одном сообщении может содержаться пакет документов, закрывающий сделку, – счет-фактуру и акт о выполнении работ (оказании услуг).

При этом набор документов, представляющих одно сообщение, со временем может меняться.

Например, порядок обмена счетами-фактурами подразумевает, что в ответ на полученный счет-фактуру покупатель должен отправить обратно продавцу специальный документ, подтверждающий факт получения счета-фактуры.

В этом случае и счет-фактура, и этот служебный документ окажутся в одном Диадок-сообщении.

Если продолжать аналогию с электронной почтой, то сообщение в Диадоке больше похоже не на отдельное e-mail сообщение, а на цепочку сообщений, возникающую в ходе переписки.

Чтобы обеспечить такой механизм эволюции сообщений во времени, вводится понятие дополнения к сообщению (патч)(MessagePatch).

Дополнения к сообщению описывают изменения, произошедшие с сообщением и могут содержать добавившиеся документы с ЭП.

Сообщение, таким образом, можно воспринимать как упорядоченный набор связанных дополнений (патчей).

Причем дополнение, с которого «начинается» сообщение, называется заглавным.

Дополнение к сообщению с точки зрения Диадока является неделимым — оно доставляется либо все целиком (то есть доставляются ВСЕ документы и ЭП из этого дополнения), либо, при возникновении каких-либо ошибок, не доставляется целиком (то есть не доставляются ни документы, ни ЭП из такого дополнения).

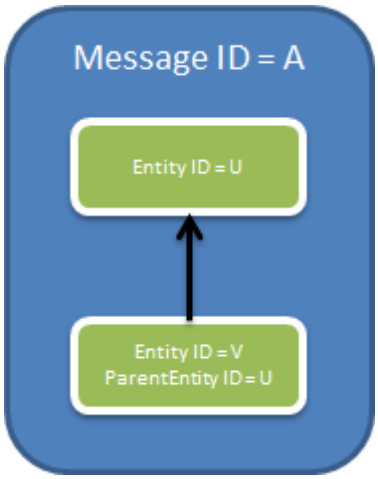
Отдельные документы, электронные подписи к ним, комментарии к документам и другие служебные данные представляются в Диадоке понятием Сущность (Entity).

Каждая сущность имеет идентификатор, тип и массив байтов, содержащий бинарное представление сущности (интерпретация этого представления зависит от типа сущности).

Например, бинарным представлением сущности, представляющей счет-фактуру, будет являться массив байтов XML-файла в формате и кодировке, определенных ФНС. А бинарным представлением сущности, представляющей ЭП, являться массив байтов отсоединенной подписи в соответствии со стандартом CMS (**RFC 5652**) в DER-кодировке.

Для однозначной идентификации сущностей нужно уже три идентификатора: - идентификатор ящика, - - идентификатор сообщения внутри ящика, - - идентификатор сущности внутри сообщения.

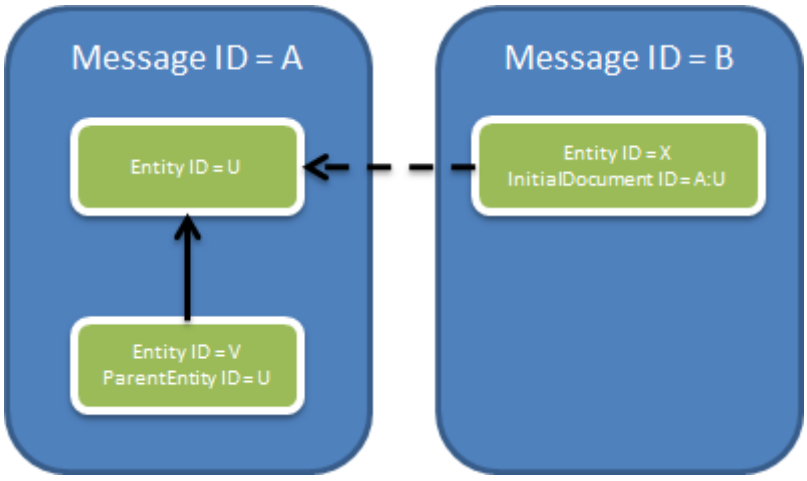
Между сущностями в Диадоке могут устанавливаться связи двух типов. Первый тип связей задается атрибутом *ParentEntityId* и служит для связывания сущностей внутри одного сообщения (при этом они могут находиться в разных дополнениях):



На рисунке сообщение A содержит две сущности – U и V. Сущность V является «дочерней» по отношению к сущности U, поскольку у нее поле *ParentEntityId* содержит значение U.

Таким способом связываются, например, документ и ЭП под ним (U – документ, V – подпись), или счет-фактура и подтверждение о его получении (U – счет-фактура, V – подтверждение).

Другой тип связей задается при помощи атрибута *InitialDocumentId* и служит для установки ссылок между документами, находящимися в разных сообщениях:



Здесь сущность X в сообщении B связана с сущностью U в сообщении A при помощи поля *InitialDocumentId*. В качестве значения этого поля используется пара идентификаторов – сообщения и сущности (A:U).

Этот механизм может использоваться, например, для связывания корректирующего счета-фактуры с исходным, или для связывания дополнительного соглашения с основным договором.

Порядок работы с API

В данном разделе описаны только базовые функции, предоставляемые посредством программного интеграторского веб-интерфейса Диадока для организации документооборота. Дополнительные возможности описаны в соответствующих разделах. Полный список команд API всегда можно посмотреть в технической документации.

Аутентификация

Любой сеанс работы клиента API начинается с прохождения процедуры аутентификации, в ходе которой программный клиент от имени пользователя получает авторизационный токен, однозначно идентифицирующий этого пользователя и дающий клиенту право читать и писать в ящики Диадока, к которым данный пользователь имеет доступ.

Пользователь может аутентифицироваться в системе либо по сертификату электронной подписи формата X.509, либо по паре логин/пароль.

В результате аутентификации формируется специальный авторизационный токен, дающий право доступа к Диадоку посредством интеграционного API в течение некоторого интервала времени (порядка нескольких часов).

Полученный авторизационный токен в дальнейшем должен передаваться в качестве параметра во все методы API, выполняющие какие-либо операции в контексте конкретного ящика, чтобы авторизовать доступ пользователя к этому ящику. Кроме того, во все методы API клиент должен передавать так называемый «ключ разработчика», однозначно идентифицирующий автора данного интеграционного решения.

См.также

Более подробно механизм аутентификации и авторизации в Диадоке описан здесь.

Передача документа

Каждый передаваемый через Диадок документ представляется в виде структуры Entity, которая может содержать как описание документа пользователя, так и описание служебного документа системы (комментарии к документу, подписи к документу и т.п.).

Соответственно, набор взаимосвязанных пользовательских и служебных документов (т.е. вся цепочка документооборота, связанная с конкретным набором пользовательских документов, передаваемых через Диадок) образует сообщение (объект Message).

Каждый документооборот может охватывать либо две организации (в этом случае естественным образом по направлению передачи первого документа в цепочке выделяются организация-отправитель и организация-получатель), либо одну организацию (в случае внутреннего документооборота внутри конкретной организации).

Соответственно, в рамках документооборота между двумя организациями формируются две цепочки: в ящике отправителя и в ящике получателя.

Документы, которые были сформированы и отправлены, отображаются в обеих цепочках. Документы, которые были только сформированы, но не отправлены (см. возможности метода PostMessage ниже), отображаются в соответствующей цепочке (либо только в ящике отправителя, либо только в ящике получателя).

Подготовка и отправка исходящих сообщений осуществляется при помощи метода PostMessage, которому на вход передается структура MessageToPost. Эта структура содержит идентификаторы ящиков участников документооборота (ящики отправителя и получателя сообщения) и собственно набор отправляемых документов.

В качестве ящика отправителя клиент может указывать только «свой» ящик, то есть ящик, к которому он может получить доступ при помощи имеющегося у него авторизационного токена.

В результате вызова метода PostMessage формируется новая цепочка документооборота, связывающая ящики отправителя и получателя.

В ящике отправителя информация о сформированной цепочке появляется в момент вызова метода PostMessage (соответственно, формируется событие о появлении документа, информацию о событиях см. ниже).

Информация о новой цепочке документооборота и связанных с ней документах в ящике получателя, вообще говоря, появится с некоторой задержкой, связанной с асинхронной передачей информации из ящика отправителя в ящик получателя.

То есть успешный вызов метода PostMessage гарантирует лишь появление исходящего сообщения в ящике отправителя; в ящике получателя сообщение и соответствующее событие могут появиться с некоторой задержкой.

Метод PostMessage можно также использовать для формирования на сервере сообщений, содержащих отправляемые документы без подписей к ним (см. флаг IsDraft структуры MessageToPost; если он выставлен в true, то сообщение будет загружено на сервер, но задание на доставку сообщения его получателю формироваться не будет). В этом случае для формирования подписей к документам и отправки сообщения следует использовать метод SendDraft.

Дополнение документа

Уже сформированные цепочки документооборота можно дополнять служебными документами при помощи метода PostMessagePatch, которому на вход передается структура MessagePatchToPost.

Эта структура содержит идентификатор цепочки документооборота, которую следует дополнить новым документом, и идентификатор ящика, с которым эта цепочка связана (если в документооборот вовлечено две организации, то в ящике второй стороны цепочка документооборота также будет обновлена; обновление производится асинхронно).

Клиент должен дополнять цепочку документооборота через «свой» ящик, то есть через тот ящик, к которому у него есть доступ.

Если загружаемый документ имеет большой размер (больше 100Кб), то для загрузки такого документа в Диадок следует пользоваться сервисом «полки документов».

В этом случае документ сначала загружается на сервер Диадока с помощью серии вызовов ShelfUpload, а затем в структурах MessageToPost и MessagePatchToPost можно ссылаться на уже загруженный документ. Такой подход позволяет повысить скорость и надежность загрузки.

Получение документа

Для получения текущего состояния конкретной цепочки документооборота можно использовать метод GetMessage, который возвращает все документы, составляющие данную цепочку, агрегированные в одну структуру Message.

Отметим, что структура Message может содержать документы, сформированные в разное время разными организациями (например, в одну такую структуру могут попасть исходящий документ одной организации и подпись к этому документу, поставленная представителем другой организации).

Для того, чтобы получить содержимое конкретного документа в цепочке документооборота, следует взять идентификаторы из полей boxId и messageId структуры Message и идентификатор документа entityId из соответствующей структуры Entity, а затем воспользоваться методом GetEntityContent.

Таким образом, каждый ящик в Диадоке может изменяться лишь одним из двух способов:

- в ящике формируется новая цепочка документооборота;
- дополняется уже существующая в ящике цепочка документооборота.

То есть вся уже существующая в ящике информация не может быть изменена, она может быть лишь дополнена. Соответственно, все модификации ящика естественным образом упорядочиваются хронологически, и можно говорить о «событиях», связанных с конкретным ящиком:

- событие о формировании новой цепочки документооборота;
- событие о добавлении документа к уже имеющейся цепочки документооборота.

Чтобы получить информацию о новых событиях следует использовать метод GetNewEvents. Этот метод предоставляет доступ к упорядоченному хронологически потоку всех Событий (BoxEvent), «происходящих» в заданном ящике.

Управление списком активных контрагентов

Для каждой организации в Диадоке ведется список ее активных контрагентов - это список тех организаций, с которыми данная организация обычно осуществляет документооборот.

Веб-интерфейс Диадока использует этот список для того, чтобы упростить выбор контрагента в тех диалогах, где это необходимо.

Если интеграционное решение хочет поддерживать аналогичную функциональность и при этом обеспечивать единство списка активных контрагентов с веб-версией Диадока, оно должно:

- получать текущее состояние списка посредством вызова метода GetCounteragents,
- при необходимости изменить состав списка - использовать методы AcquireCounteragent для добавления контрагента в список активных и BreakWithCounteragent для удаления контрагента из списка активных.

Присутствие контрагента в списке активных никак не влияет на возможность документооборота с контрагентом.

Зная идентификатор организации-контрагента, можно получить более подробную информацию о самом контрагенте и о документообороте с ним. Для этого следует использовать метод GetCounteragent.

Управлять списком активных контрагентов можно и через веб-интерфейс Диадока (https://diadoc.kontur.ru). Изменения, сделанные через веб-интерфейс, отражаются на списках контрагентов, получаемых через API, и наоборот.

Поэтому в каких-то случаях может оказаться проще не включать в интеграционное решение логику управления списком активных контрагентов, и выполнять такое управление через веб-интерфейс (нужный список может быть сформирован через веб-интерфейс перед вводом интеграционного решения в эксплуатацию).

Такой подход может быть оправдан, например, когда списки контрагентов организаций-потребителей интеграционного решения не очень большие и редко меняются с течением времени.

Дополнительные функции API

Ниже дан краткий обзор функций API, не упомянутых в предыдущих разделах. Более полную информацию по всем методам и структурам данных Diadoc API можно найти в технической документации.

Отправка заявления участника ЭДО

При помощи метода SendFnsRegistrationMessage можно отправить заявление участника ЭДО.

Поиск контрагентов

Метод GetOrganizationsByInnKpp позволяет искать в Диадоке ящики по ИНН и КПП организаций, которым они принадлежат.

Метод *GetOrganizationsByInnKpp* возвращает список организаций и ящиков, удовлетворяющих критерию поиска, в виде структуры данных OrganizationList.

При помощи метода GetBox можно по идентификатору ящика получить информацию об организации, которой он принадлежит.

При помощи метода GetOrganization можно по идентификатору организации узнать различные справочные данные, заведенные в Диадок, такие как ИНН, КПП, название, а также получить список ее ящиков.

Черновики

Диадок позволяет клиенту API помещать в «свой» ящик еще не подписанные документы без их немедленной отправки контрагентам.

Это может быть полезно, когда подготовкой документов занимается один человек, а правом подписи и отправки обладает другой. Такие документы называются черновиками. Для их создания через API используется метод PostMessage.

Подготовка печатных форм

Для документов, передаваемых через Диадок, в формализованном не человекочитаемом виде, предусмотрена возможность получения их печатных форм.

В первую очередь, этот функционал предназначен для формирования печатных форм счетов фактур и всех служебных документов, предусмотренных порядком Минфина в ходе документооборота счетов-фактур. Печатные формы интересующих пользователя документов можно получить через API при помощи метода GeneratePrintForm.

Фильтрация списка документов

При помощи метода GetDocuments, можно быстро получать информацию о документах (например, о счетах-фактурах) в своем ящике, задавая различные критерии фильтрации документов.

Например, можно запросить список всех входящих счетов-фактур от заданного контрагента за определенный период. В ряде сценариев этот метод может оказаться более удобным по сравнению с механизмом получения новостей методами GetNewEvents, GetEvent и GetMessage.

Кроме того, есть возможность получить всю метаинформацию об отдельном документе, зная его идентификатор. Для этого предназначен метод GetDocument.