Глава 12

Службы и инфраструктура интеграции приложений

В этой главе

- Введение
- Типы служб Microsoft Dynamics AX
- Использование служб Microsoft Dynamics AX
- Инфраструктура отправки односторонних запросов
- Использование внешних веб-служб
- Аспекты производительности

Введение

Сразу после внедрения в вашей компании Microsoft Dynamics AX 2012, она начинает приносить пользу от автоматизации ваших бизнес-процессов. Однако для полного раскрытия потенциала Microsoft Dynamics AX 2012, получения максимального уровня отдачи и повышения коэффициента возврата инвестиций (ROI) от развернутого приложения следует подумать над автоматизацией взаимодействия между Microsoft Dynamics AX 2012 и другим программным обеспечением в вашей компании и компаниях-партнерах. Во многих бизнес-процессах требуется доступ со стороны внешних приложений к информации, хранимой в Microsoft Dynamics AX 2012. На рис. 12-1 изображены несколько сценариев, в которых различные пользователи обращаются к информации Microsoft Dynamics AX 2012 для выполнения определенных деловых задач, а также показаны сценарии, в которых сама Microsoft Dynamics AX обращается за информацией к сторонним приложениям. Направление доступа к информации указано стрелками.

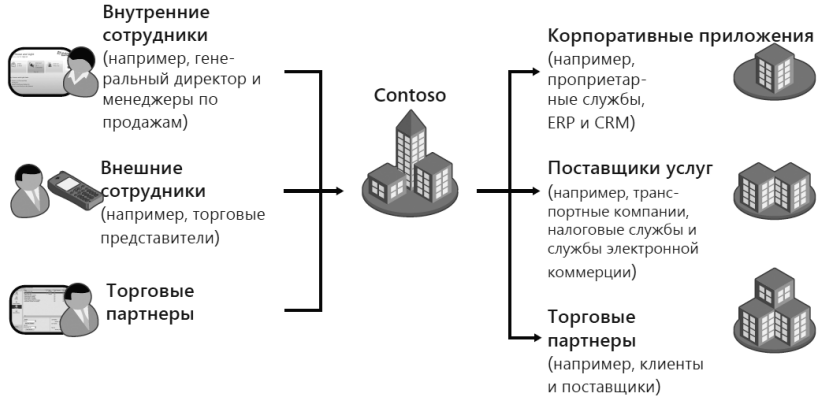

Рис. 12–1. Распространенные сценарии интеграции приложений

Если вы посмотрите на рис. 12-1, то увидите, что по левую сторону расположены приложения, взаимодействующие с хранилищем данных Microsoft Dynamics AX. Эти приложения отправляют запросы (например, для чтения информации о заказе на продажу) к Microsoft Dynamics AX и в большинстве случаев ожидают от нее ответа (к примеру, экземпляр запрашиваемого заказа на продажу).

Во всех этих сценариях сторонние приложения обмениваются информацией с Microsoft Dynamics AX для решения определенной задачи.

- Генеральный директор компании анализирует данные о продажах в Microsoft Dynamics AX в приложении Office.
- Менеджер по продажам, встречаясь с потенциальным клиентом, использует веб-страницу или мобильное приложение для создания нового клиента, а также первого заказа на продажу.
- Торговый представитель вводит заказ на продажу и использует информацию о клиенте, хранимую в приложении CRM, для заполнения данных о клиенте в заказе Microsoft Dynamics AX.
- Торговый партнер размещает заказы на продажу в виде электронных документов, которые должны периодически импортироваться в Microsoft Dynamics AX.
- Бухгалтер отправляет электронные платежи или накладные торговым партнерам.

Выполнение этих действий вручную, без интеграции Microsoft Dynamics AX со сторонними приложениями и бизнес-процессами плохо масштабируется и подвержено ошибкам. С использованием инфраструктуры служб Microsoft Dynamics AX вы можете поместить бизнес-логику (например, создание заказов на продажу) в службы Dynamics AX. Опубликовав их с помощью инфраструктуры интеграции приложений (AIF), вы можете сделать их участниками сервисно-ориентированной архитектуры (SOA).

Примечание. SOA является важной областью разработки программного обеспечения. Полное ее описание выходит за рамки этой книги. Отличное изложение содержания SOA включено в спецификацию «Reference Model for Service Oriented Architecture 1.0», изданную *Organization for the Advancement of Structured Information Standards* (OASIS), и книгу «Service-Oriented Architecture: Concepts, Technology, and Design», написанную Thomas Erl.

Инфраструктура служб Microsoft Dynamics AX предлагает набор инструментов для создания, управления, настройки и публикации служб Microsoft Dynamics AX, позволяя легко предоставить внешним приложениям бизнес-логику, реализуемую службами через их интерфейсы. Все интерфейсы служб, опубликованные с помощью инфраструктуры служб Dynamics AX, совместимы с промышленными стандартами и основываются на ключевых технологиях Microsoft, включая пакет разработчика программного обеспечения (SDK) для Microsoft Windows Server, Microsoft .NET Framework, Windows Communication Foundation (WCF) и Message Queuing (также известную как MSMQ).

Наряду с моделью программирования и средствами реализации служб, инфраструктура служб Microsoft Dynamics AX включает:

- набор системных служб и служб документов, которые поставляются с Microsoft Dynamics AX 2012 и уже готовы к использованию;
- набор возможностей обработки входящих и исходящих сообщений, такие как поддержка преобразований, подстановки значений и т.д.
- расширяемая инфраструктура интеграции, поддерживающая создание новых сервисов Microsoft Dynamics AX и их публикацию с использованием множества транспортных протоколов, таких как Message Queuing, файл HTTP или Net.tcp.

Примечание. Концепция ссылок на службы была убрана из Microsoft Dynamics AX 2012.

Публикация сервисов Microsoft Dynamics AX является простой задачей, которую администратор системы может осуществить прямо в процессе работы системы. После опубликования службы Microsoft Dynamics AX приложения-потребители данной службы могут использовать ее для интеграции с Microsoft Dynamics AX.

Примечание. В этой главе обсуждаются вопросы настройки и администрирования в объеме, минимально необходимом для понимания сценариев разработки. Для получения подробных указаний и примеров кода обратитесь к документации по администрированию Microsoft Dynamics AX 2012 на TechNet (*http://technet.microsoft.com/en-us/library/gg731797.aspx*) или в Microsoft Dynamics AX 2012 Developer Center в MSDN (*http://msdn.microsoft.com/en-us/dynamics/ax/gg712261*).

Типы служб Microsoft Dynamics AX

В Microsoft Dynamics AX 2012 различают три типа служб: системные, произвольные и службы документов. Каждый из них имеет свою модель программирования. Microsoft Dynamics AX публикует метаданные этих служб и их возможностей в форме файла Web Services Description Language (WSDL), который может быть использован для автоматического создания для них вспомогательного класса (далее – прокси).

В последующих разделах мы рассмотрим каждый тип служб подробнее.

Системные службы

Системные службы Microsoft Dynamics AX – это основные инфраструктурные службы, не относящиеся к конкретной бизнес-логике. Системные службы, включенные в Microsoft Dynamics AX 2012, развертываются автоматически, так что компоненты Microsoft Dynamics AX и внешние компоненты всегда могут быть уверены в их наличии.

Функциональность, доступная с помощью системных служб, часто используется интерактивными клиентами, запрашивающими возможно-

сти или настройки конкретного приложения в процессе работы системы. Системные службы и их интерфейсы не предусматривают их изменение и настройку; они могут размещаться только на сервере приложений (AOS) и не могут быть вызваны через асинхронные транспортные механизмы, такие как Message Queuing.

Microsoft Dynamics AX включает следующие системные службы.

- **Служба запросов.** Предоставляет доступ к службе, позволяющей выполнять существующий (статический) или произвольный запросы от клиентов службы, возвращая результаты в форме набора данных .NET.
- **Служба метаданных.** Может быть использована для получения информации от Microsoft Dynamics AX о ее метаданных, таких как таблицы, запросы, формы и так далее, то есть о ее конфигурации.
- **Служба информации о сессии.** Может быть использована для получения настроек окружения, в котором выполняются запросы текущего пользователя; например, клиентское приложение может использовать эту службу для получения информации о текущей валюте, компании, часовом поясе пользователя и другой информации.

Произвольные службы

Произвольные службы Microsoft Dynamics AX используются для публикации необходимых методов X++, таких как действия служб посредством портов интеграции для дальнейшего их использования внешними клиентскими приложениями. Для реализации этого вам потребуется использовать модель разработки произвольных служб, чтобы определить метаданные, задающие вид публикуемых действий служб и контрактов данных. Произвольные службы необязательно должны сопоставляться с какими-либо запросами или таблицами Microsoft Dynamics AX. Например, вы можете создать произвольную службу для публикации функциональности одобрения накладной или остановки платежа.

Примечание. Вообще говоря, для публикации стандартных действий, манипулирующих запросами или таблицами, таких как создание, чтение, обновление и удаление (часто называемых операциями CRUD), лучше использовать службы документов Microsoft Dynamics AX.

После определения действий служб и контрактов данных вы можете опубликовать свою произвольную службу. Ее внешние интерфейсы можно настроить, используя соответствующие формы администрирования системы.

Артефакты произвольных служб.

Чтобы представить метод X++ в виде произвольного сервиса, вам нужно создать следующие артефакты.

- **Класс, реализующий службу.** Класс, реализующий бизнес-логику, доступную через методы X++.
- **Контракт службы.** Относящиеся к службе метаданные (не код). Наиболее важные метаданные службы состоят из действий службы, публикуемой для использования внешними приложениями, и ссылки на класс X++ реализации службы, в котором и реализованы эти действия службы.
- **Один или несколько контрактов данных.** Классы X++, представляющие сложные типы параметров действий службы. Контракты данных для простых типов данных не требуются.

Класс реализации службы

В классе реализации службы содержится код, выполняющий публикуемую бизнес-логику.

Таким классом может быть любой X++-класс. Он не обязан реализовывать какие-либо интерфейсы или наследовать какие-либо родительские классы. Определение класса для класса реализации службы *MyService* может выглядеть так:

```
public class MyService
{
}
```

Но существует ограничение, которому должны подчиняться методы класса реализации службы, которые будут опубликованы как действия службы. Такие методы должны быть public-методами, использующими параметры только тех типов, которые могут быть сериализованы и десериализованы; сюда входит большая часть простых типов данных, наряду с корректными контрактами данных Microsoft Dynamics AX. Кроме того, такие методы должны быть объявлены, как действия служб в контракте службы в AOT.

Примечание. Каждый метод, который будет опубликован как действие службы, должен быть отмечен атрибутом *SysEntryPointAttribute,* который указывает, что сервером приложений должны быть проведены проверки контроля доступа.

Нижеприведенный код представляет собой пример метода, который может быть объявлен в AOT как действие службы. Предполагается, что тип X++ *MyParam – это корректный контракт данных. (Подробнее об этом будет рассказано в разделе «Контракты данных» ниже в этой главе.)*

```
[SysEntryPointAttribute(true)]
public MyParam HelloWorld(MyParam in)
{
    MyParam out = new MyParam();
    out.intParm(in.intParm() + 1);
    out.strParm("Hello world.\n");
    return out;
}
```

Контракты служб

Контракты служб определяют, какие методы класса реализации службы могут быть опубликованы как действия службы и содержат дополнительные метаданные, указывающие, как эти методы должны быть опубликованы.

Примечание. Объявление метода как действия службы отнюдь не означает автоматической публикации метода как действия службы.

Чтобы создать новый контракт службы, вам нужно создать новый дочерний узел в AOT под узлом *Services*, например, *MyService.*

Новый узел **AOT имеет несколько свойств, которые нужно задать** перед публикацией любых методов как действий службы.

- **Service implementation class.** Это обязательное свойство связывает интерфейс службы с классом ее реализации. В этом примере его значение будет *MyService.*
- **Namespace. Необязательное. Вы можете указать пространство имен XML, которое будет использовано в WSDL. Если оно не указано,** то **будет использовано значение по умолчанию** *http://tempuri.org.* Этот пример использует пространство имен *http://schemas.contoso.com/axbook/2012/services*.

- **External name.** Необязательное. Вы можете задать внешнее имя каждой службе. В этом примере оно оставлено пустым.

Наконец, вам нужно добавить действия службы в контракт. Раскройте созданный узел AOT, щелкните на нем правой кнопкой мыши и выберите Operations > Добавить Operation.

Заметим, что вы можете опубликовать как действие службы только те методы, которые были явно добавлены в контракт службы в AOT.

Контракты данных

Контракт данных – это сложный тип данных X++, используемый как входной и выходной параметр в действиях служб. Еще важнее то, что он должен быть сериализуемым. Атрибутами X++ *DataContractAttribute* и *DataMemberAttribute* можно управлять тем, как он будет упакован и распакован инфраструктурой служб Microsoft Dynamics AX.

- *DataContractAttribute объявляет класс* X++ как контракт данных.
- *DataMemberAttribute объявляет* атрибут класса свойством контракта данных.

Этот код демонстрирует определение контракта данных *MyParam* из предыдущего примера:

```
[DataContractAttribute]
public class MyParam
{
    int intParm;
    str strParm;
}
```

А следующий код показывает объявление свойства, включаемого в контракт данных:

```
[DataMemberAttribute]
public int intParm(int _intParm = intParm)
{
    intParm = _intParm;
    return intParm;
}
```

Коллекции X++ как контракты данных

Если вы хотите использовать в роли контракта данных типы коллекций X++, вам нужно обеспечить то, что все включенные в коллекцию элемен-

ты должны иметь тип, поддерживающий контракты данных. Кроме того, необходимо в процессе разработки задать дополнительные метаданные на определении метода службы, использующего параметр, которые укажут точный тип данных значений в коллекции. Это можно сделать с помощью X++ атрибута *AifCollectionTypeAttribute, как* показано для приведенного ниже метода *UseIntList()*:

```
[SysEntryPointAttribute(true),
    AifCollectionTypeAttribute('inParm', Types::Integer)]
public void UseIntList(List inParm)
{
...
}
```

Два параметра, которые нужно передать конструктору атрибутов – это имя параметра, которому будут присвоены метаданные (в примере – *inParm*), и тип элементов в коллекции (в примере – *Types::Integer*).

Если же вы собираетесь использовать в своей коллекции типы классов X++, то должны также указать имя класса, как показано ниже:

```
[SysEntryPointAttribute(true),
    AifCollectionTypeAttribute('return', Types::Class, classStr(MyParam))]
public List ReturnMyParamList(int i)
{
...
}
```

Три параметра, переданные в конструктор *AifCollectionAttribute:* имя параметра (*return*), тип элементов в коллекции (*Types::Class*) и указанный тип класса (*MyParam*).

Примечание. Зарезервированное название параметра *return* закреплено за возвращаемым значением метода.

Регистрация произвольной службы

После создания всех нужных для произвольной службы артефактов следующим шагом будет регистрация новой службы в инфраструктуре служб Microsoft Dynamics AX. Для ее регистрации (в примере – службы *MyService*) в AIF, откройте в AOT узел Services, щелкните правой кнопкой мыши на созданном вами ранее узле и выберите Надстройки > Регистрация услуг.

После регистрации у вас появится возможность опубликования всех объявленных действий службы. Дополнительную информацию см. в разделе «Публикация служб Microsoft Dynamics AX» ниже в этой главе.

Службы документов

Термин *служба документов* порожден реальной необходимостью обмена деловыми документами, такими как накладные и заказы на продажу, между предприятием и его деловыми партнерами. Службы документов работают с электронными версиями таких бизнес-документов.

Реализацию этих документов в Microsoft Dynamics AX еще называют *Axd-документами*. Службы документов строятся на базе запросов Microsoft Dynamics AX. Набор мастеров автоматизирует и убыстряет процесс создания и поддержки всех необходимых артефактов для этих сервисов на базе запросов, позволяя реализовать настраиваемый набор известных действий.

Естественно, службы документов предоставляют ориентированные на работу с документами программные интерфейсы (API), работающие с *Axd-документами*. Примерами таких интерфейсов для службы работы с заказами на продажу могут быть *создание заказа на продажу, чтение заказа на продажу, удаление заказа на продажу* и т.п. Каждый из приведенных интерфейсов работает с экземпляром документа заказа на продажу, находящимся в хранилище данных Microsoft Dynamics AX, и возвращает идентификатор сохраненного заказа на продажу.

Службы документов полезны в сценариях, требующих обмена жестко структурированными деловыми документами, такими как заказы на продажу. В этих сценариях передаваемые данные должны быть переданы полностью (т.е. являются транзакционными). Важен строгий контроль значений данных (их валидация). Обмен данными обычно связан с большими накладными расходами (пересекается граница предприятия). Не требуется быстрого отклика. Иногда отклик даже не ожидается (односторонняя передача).

Модель программирования для служб документов поддерживает изменение созданных артефактов. В Microsoft Dynamics AX уже включен готовый к использованию набор служб документов. Однако вы можете изменить их для лучшего соответствия потребностям вашего бизнеса. Модель разработки также поддерживает новые возможности уровня доступа

к данным Microsoft Dynamics AX 2012, такие как добавление суррогатных ключей, наследование таблиц и действительные даты. Другими словами, инфраструктура служб Microsoft Dynamics AX полностью поддерживает разработку сервисов, работающих с таблицами, использующими новую функциональность.

Артефакты служб документов

Как и произвольные службы, все службы документов в Microsoft Dynamics AX 2012 требуют контракта службы, контракта данных и ее реализации. Для служб документов эти артефакты создаются на основе запросов *Axd*. Таким образом, их обычная реализация подчиняется определенным соглашениям и выглядит так.

- **Контракт службы.** Относящиеся к службе метаданные (кроме кода), хранящиеся в узлах AOT, подчиненных узлу *Services,* такие как *SalesSalesOrderService*. Это метаданные включают:
  - доступные внешним клиентам операции служб;
  - ссылку на X++-класс реализации службы, который, собственно, и реализует эти операции службы.
- **Реализация службы.** Код, который реализует публикуемую вовне бизнес-логику. Для создаваемых служб документов их реализация включает следующие ключевые составные части.
  - **Класс реализации службы.** Класс X++, унаследованный от *AifDocumentService* и реализующий операции службы, опубликованные посредством контракта службы. Например, *SalesSalesOrderService* – это класс реализации службы для контракта *SalesSalesOrderService*.
  - **Класс *Axd*<Document>.** **Класс X++, унаследованный от** *AxdBase. Классы Axd*<Document> **управляют валидацией и заданием** значений по умолчанию для таблицы, хранящей записи документа. Для каждой службы документов существует один класс *Axd*<Document>. Например, *AxdSalesOrder* – это класс *Axd*<Document> для *SalesSalesOrderService*. Класс *AxdBase*, помимо всего прочего, содержит код для упаковки в XML.
  - **Прочие артефакты.** Опционально мастер создания служб документов AIF может создать дополнительные артефакты, такие как классы *Ax*<Table>.

Примечание. В старых версиях Microsoft Dynamics AX класс *Ax*<Table> создавался для каждой таблицы, входившей в запрос, исходя из которого создавался класс *Axd*<Document>. По умолчанию в Microsoft Dynamics AX 2012 классы *Axd*<Document> для доступа к таблицам *Ax*<Table> используют класс *AxCommon*. Этот класс предоставляет реализацию по умолчанию для всей функциональности *Ax*<Table> классов. Так что эти классы *Ax*<Table> теперь нужны только для нестандартных сценариев, например когда существует специфическая потребность в преобразовании значений при их присвоении полям таблицы.

- **Объект данных.** Класс X++, олицетворяющий тип параметра и выступающий в роли контракта данных. Типы параметров, генерируемые мастером создания служб документов AIF, наследуются от *AifDocument* и представляют собой бизнес-документы. Например, *SalesSalesOrder – это объект данных, созданный для SalesSalesOrderService.*

Для получения полного списка артефактов служб документов обратитесь к разделу «Services and Application Integration Framework (AIF)» в Microsoft Dynamics AX 2012 SDK (*http://msdn.microsoft.com/en-us/library/gg731810.aspx)*.

В следующем разделе затронуты несколько избранных тем относительно классов *Axd*<Document> и *Ax*<Table>. Дополнительная информация имеется в разделе «AIF Document Services» в Microsoft Dynamics AX 2012 SDK (*http://msdn.microsoft.com/en-us/library/bb496530.aspx)*.

Классы *Axd*<Document>

Классы *Axd*<Document> (такие как *AxdSalesOrder*) унаследованы от X++-класса *AxdBase*. Кроме всего прочего, классы *Axd*<Document> занимаются следующими задачами.

- Упаковывают в XML объекты данных.
- Вызывают механизмы подстановки значений.
- Управляют проверкой и заданием значений по умолчанию для таблиц, входящих в документ.

Классы *Axd*<Document> предлагают реализацию по умолчанию для сериализации в XML всех используемых объектов данных. Эти классы получают определение XML-схемы, используемой для сериализации, пря-

мо из структуры запроса, лежащего в основе документа. Код упаковки в XML использует такие концепции Microsoft Dynamics AX, как расширенные типы данных (EDT) для жесткого ограничения допустимых XML-схем и улучшения контроля за их корректностью. Кроме того, при генерации XML-схем классы *Axd*<Document> учитывают возможности уровня доступа к данным, появившиеся в Microsoft Dynamics AX 2012. Например, созданные определения XML-схемы отражают действительные на дату поля таблицы, поля с расширенными аналитиками и структуру наследования таблиц, использованных в конкретном *Axd*-запросе, если эти новые возможности используются. Если для таблицы настроена замена суррогатных внешних ключей, то они будут заменены альтернативными ключами.

Классы *Axd*<Document> всегда оперируют с таблицами посредством классов *Ax*<Table>. Во время сериализации классы *Axd*<Document> для сохранения и чтения данных в таблице основываются на функциональности *AxCommon или вашего Ax*<Table>-класса. На рис. 12-2 изображено соответствие между запросом, использованным для класса *AxdSalesOrder (типа Axd*<Document>), и сгенерированным определением XML-схемы.

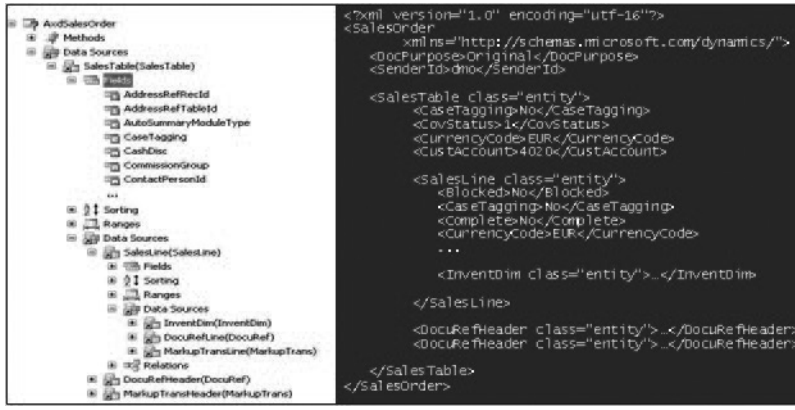

Рис. 12–2. Соответствие между запросом в AOT и структурой XML-документа

Классы *Axd*<Document> также предоставляют программный интерфейс для организации задания значений по умолчанию и валидации табличных полей. В них также может быть реализована логика проверки корректности значений и их задания по умолчанию, специфичная только для конкретного класса *Axd*<Document> (но не для всех классов *Axd*<Document>, использующих одну и ту же таблицу).

Экземпляры *Axd*<Document> могут быть однозначно определены по *AifEntityKeys*, состоящих из названия таблицы (имени корневой таблицы для запроса *Axd*), названий полей уникального индекса этой таблицы и значений этих полей в выбранной записи. Вдобавок, в *AifEntityKeys* входит код выбранной записи – *RecId)*.

Классы *Ax*<Table>

Классы *Ax*<Table> (такие как *AxSalesTable и AxSalesLine*) наследуются от X++-класса *AxInternalBase*. В отличие от предыдущих версий Microsoft Dynamics AX, класс *Ax*<Table> больше не нужно создавать для каждой таблицы, используемой в службе документа. Взамен в Microsoft Dynamics AX 2012 входит *Ax*<Table>-класс *AxCommon*, по умолчанию используемый классами *Axd*<document> для доступа к таблицам.

Примечание. Службы документов, поставляемые с Microsoft Dynamics AX 2012, могут по-прежнему основываться на специфичных классах *Ax*<Table> для таблиц из их запросов, особенно если эти службы были созданы еще в ранних версиях Microsoft Dynamics AX, до появления класса *AxCommon*.

Однако есть сценарии, в которых нужны собственные классы *Ax*<Table>, например, в случае, когда необходимо, чтобы *parm*-методы в используемой таблице делали следующее.

- Поддерживали вычисляемые поля для таблицы на базе класса *Ax*<Table>.
- Поддерживали особую подстановку значений, отличающуюся от реализации по умолчанию в *AxCommon*.

Примечание. Классы *Ax*<Table> часто в коде и документации называют еще классами *AxBC*.

Несмотря на необязательность классов *Ax*<Table>в Microsoft Dynamics AX 2012, мастер создания служб документов AIF может создать и их в процессе создания службы документов.

Генерация служб документов

Службы документов генерируются на основе запросов *Axd c* использованием мастера служб документов AIF. В этом разделе обсуждаются аспекты создания и поддержки служб документов.

Создание *Axd*-запросов. К *Axd*-запросам применимы общие рекомендации по работе с запросами Microsoft Dynamics AX. Помимо этого, для них существуют свои рекомендации и требования по соответствию дополнительным ограничениям.

- Название запроса Microsoft Dynamics AX, используемое для служб документов, должно начинаться с префикса *Axd* и продолжаться названием документа. Например, для запроса службы документа *SalesOrder* оно должно быть *AxdSalesOrder*. Это рекомендация Best Practice.
- В каждом запросе допустима только одна корневая таблица. Вы можете связать уникальный ключ сущности, используемый для различения экземпляров документа с этой корневой таблицей. Например, ключ сущности *SalesId* определен в *AxdSalesOrder* из корневой таблицы *SalesTable*.
- Если источники данных в запросе соединены внутренним объединением (inner join), вы должны использовать режим выборки (fetch mode) 1:1. Если же они соединены внешним объединением (outer join), то используйте режим выборки 1:*n*. В случае неиспользования этих установок ваш запрос и операции службы могут привести к непредсказуемым результатам.
- Если вы хотите, чтобы ваша служба обладала возможностью сохранять данные обратно в базу данных (то есть поддерживала операцию *update)*, установите у всех источников данных, входящих в базовый запрос, свойство AOT *Update* в *Yes*.

Примечание. По соображениям безопасности, проверки в коде X++ запрещают участвовать системным таблицам в запросах, используемых в службах документов.

Генерация служб документов. Для создания службы документов по существующему *Axd-запросу* вы можете использовать мастер создания служб документов AIF. Для его запуска в меню Сервис выберите AIF > Создать службу документов. Этот раздел представляет общее описание мастера создания служб документов AIF и несколько важных замечаний по его использованию.

В мастере вы можете выбрать операции службы, которые она будет поддерживать: *create*, *read*, *update*, *delete*, *find*, *findKeys, getKeys* и *getChangedKeys*. Если вы отметите пункт Создание классов *AxBC* в процессе прохож-

дения шагов мастера, он создаст новые классы *Ax*<Table> с соответствующими *parm*-методами для полей таблиц, участвующих в запросе.

Мастер создания служб документов AIF использует вводимое вами на первом шаге название документа для назначения имен создаваемым артефактам. Вы можете изменить название документа в мастере (и, следовательно, имена порождаемых артефактов) до их, собственно, создания. Длина названий объектов AOT ограничена 40 символами. Если вы введете имя документа, которое приведет к присвоению слишком длинных имен каким-либо создаваемым артефактам, то можете получить сообщения об ошибках.

После завершения своей работы мастер покажет отчет о всех созданных артефактах и ошибках, если таковые были. Необходимо будет исправить все ошибки перед началом кастомизации кода, созданного мастером.

Совет. Для каждой создаваемой службы мастер создает отдельный проект. В него помещаются все создаваемые артефакты.

Для обновления существующих служб документов (например, для добавления действия службы, не созданной ранее) используйте диалоговое окно Обновить службу документов.

Примечание. В принципе, вы можете создавать и обновлять службы документов и вручную. Однако делать это не рекомендуется. Вместо этого для создания новых служб документов по запросам AOT и быстрого обновления существующих служб документов постоянно используйте мастер создания служб документов AIF и диалоговое окно Обновить службу документов.

В Microsoft Dynamics AX 2012 уже включено свыше 100 готовых к использованию служб документов. Сюда входят такие службы, как *SalesOrderService* и *CustomerService*. В узле AOT *Services* вы можете увидеть их полный список. Он также есть в тематическом разделе «Standard Document Services» в Microsoft Dynamics AX 2012 SDK (*http://msdn.microsoft.com/en-us/library/aa859008.aspx).*

Для всестороннего рассмотрения мастера создания служб документов AIF и генерации классов *Axd*<Document> и *Ax*<Table> обратитесь к разделу «AIF Document Services» в Microsoft Dynamics AX 2012 SDK (*http://msdn.microsoft.com/en-us/library/bb496530.aspx).*

Изменение служб документов

Зачастую для более точного соответствия потребностям предприятия приходится вносить изменения в службы документов, созданные вами или входящие в Microsoft Dynamics AX 2012. Этот раздел затрагивает некоторые из наиболее часто встречающихся сценариев изменения служб документов, включая кастомизацию таблиц или запросов, операций служб, проверок значений, задания значений по умолчанию и безопасности.

Изменение таблиц. При изменении таблиц, используемых в службе документов (например, при добавлении столбца), вам для отражения этих изменений нужно обновить реализацию службы, т.е. классов *Axd*<Document>, *Ax*<Table> и связанных объектов данных.

Совет. Всегда держите включенными проверки инструмента рекомендаций Best Practice для отслеживания потенциальных несоответствий между структурой таблицы и реализациями служб, основанных на ней. Если проверки на соблюдение рекомендаций для какой-либо измененной таблицы не пройдут, вы сможете сразу же устранить ошибки с использованием диалогового окна Обновить службу документов. Обновив класс *Axd*<Document>, классы *Ax*<Table> и связанные объекты данных, вы отразите в них внесенные в структуру таблицы изменения.

Внимание. Так как службы документов основываются на запросах Microsoft Dynamics AX, то изменения в запросе, используемом как основа службы документов (например, добавление столбца в таблицу, входящую в запрос), приведет к неумышленному изменению контракта данных этой службы документов. Такие изменения во внешних интерфейсах (интерфейсах служб) могут потенциально привести к сбою интеграции, построенной на исходном контракте данных. Всегда старайтесь оценивать влияние изменений запросов или таблиц, используемых в службах документов, и применяйте общие правила ведения разработки для обеспечения корректного изменения интерфейсов служб, не удаляя их операции, поля контрактов данных и добавляя только необязательные поля.

Совет. Если вы используете фиксированный набор полей в запросе, по которому сгенерирована служба *Axd* документа, то тем самым вы можете предотвратить неявное изменение контракта данных службы документа *Axd* после добавления в таблицу нового поля.

Добавление произвольных действий службы. Вы можете изменить поведение любого действия службы путем модификации ее X++-реализации. Кроме того, вы можете добавить любое произвольное действие в службу, следуя точно такими же шагами, как и в произвольных службах.

Изменение логики проверки значений. Логика проверки значений критически важна для обеспечения корректности и целостности данных. В идеале, в хранилище данных Microsoft Dynamics AX не должно храниться некорректных данных.

Совет. Для достижения этой цели всегда проверяйте наличие и работу логики проверки значений в каждом действии службы, которое создано или изменено вами, чтобы убедиться в полном ее соответствии вашим требованиям к корректности данных.

Хорошо спроектированная логика контроля значений имеет следующие характеристики.

- **Повторно используемая.** В идеале, в клиенте Microsoft Dynamics AX и службах Microsoft Dynamics AX должна использоваться одна и та же общая логика проверки значений. Помните, что возможна ситуация и различного кода валидации, работающего только в клиенте Microsoft Dynamics AX или только в службах Microsoft Dynamics AX.
- **Производительная.** Код проверки значений работает при любом изменении соответствующих сущностей Microsoft Dynamics AX. Как следствие, одна из ключевых задач при его написании – обеспечение достаточной производительности.
- **Достаточная.** Логика проверки значений должна обеспечивать достаточный уровень корректности данных. В некоторых случаях, в зависимости от задач вашего приложения, возможен компромисс между требуемым уровнем корректности данных и производительностью.

Код валидации в основном состоит из следующих элементов.

- Код, управляющий проверкой данных в связанных таблицах, вызывающий соответствующую логику проверки для каждой из таблиц. Он реализуется в методах соответствующего класса *Axd*<Document>, таких как *prepareForSave*, *prepareForUpdate* и *prepareForDelete*. Эти методы *prepareForXxx* вызываются по одному разу для каждого класса *Ax*<Table>, используемого в классе *Axd*<Document>.
- Код, осуществляющий проверку данных на уровне таблицы, реализуется табличными методами *validateField* и *validateWrite*, что обеспечивает возможность его повторного использования (в частности, в клиенте Microsoft Dynamics AX и Enterprise Portal). Эти методы вызывают конкретные, специфичные для таблицы методы, вроде *checkCreditLimit* на таблице *SalesTable*.
- Код, проверяющий данные на уровне документа, находится в методе *validateDocument* класса *Axd*<Document>. Он вызывается непосредственно перед сохранением данных в таблицы и после вызова методов *prepareForXxx* для каждого из классов *Ax*<Table>.
- Код, выполняющий проверку значений уже после сохранения данных в таблице, реализован в методе *updateNow* класса *Axd*<Document>.

В следующем фрагменте кода показан метод *prepareForSave* класса *AxdSalesOrder*, как пример, выполняющий проверку данных в связанных таблицах. Он вызывает методы проверки из классов *Ax*<Table>, таких как *AxSalesTable* и *AxSalesLine* (наряду с остальными классами *Ax*<Table>, для упрощения не показанных):

```
public boolean prepareForSave(AxdStack _axdStack, str _dataSourceName)
{

    // ...

    switch (classidget(_axdStack.top()))
    {
        case classnum(AxSalesTable) :
            axSalesTable = _axdStack.top();
            this.checkSalesTable(axSalesTable);
            this.prepareSalesTable(axSalesTable);
            return true;
```

```
            case classnum(AxSalesLine) :
                axSalesLine = _axdStack.top();
                this.checkSalesLine(axSalesLine);
                this.prepareSalesLine(axSalesLine);
                return true;

            // ...
        }

        return false;
    }
```

Изменение логики установки значений по умолчанию. Вы можете изменить логику установки значений по умолчанию для полей таблиц, выполняющуюся во время создания или обновления их строк. Эта логика помогает увеличить удобство использования как интерактивных клиентов, так и интерфейсов служб Microsoft Dynamics AX. Она задает начальные значения для полей таблиц на основании других данных – таких как значения других полей – и, соответственно, не требует явного указания значений для каждого поля в записи таблицы. Кроме того, она помогает сократить количество данных, нужное для работы с более сложными сущностями, вроде заказов на продажу, уменьшая вероятность ввода ошибочных данных.

Хорошо спроектированная логика установки значений по умолчанию имеет следующие характеристики.

- **Повторно используемая.** Логику установки значений по умолчанию следует разрабатывать таким образом, чтобы ее можно было использовать повторно в других участках системы независимо от того, какой клиент Microsoft Dynamics AX используется для создания или обновления записи (пользовательский интерфейс или клиент службы). В некоторых ситуациях может понадобиться такое задание значений по умолчанию, которое будет зависеть от того, используется ли интерактивный клиент Microsoft Dynamics AX (приложение клиента) или не интерактивный (запрос от клиента службы).
- **Производительная.** Поскольку логика задания значений по умолчанию используется всякий раз при создании или обновлении полей, время ее выполнения прямо влияет на время обработки сущностей (например, заказов на продажу). Поэтому следует избегать выполнения излишних операций установки значений, вроде установки одного

значения и затем перезаписи этого значения другим в той же цепочке выполнения кода.

- **Достаточная.** Для сокращения числа полей, которые нужно обязательно заполнить для работы с сущностью, следует заполнять значениями по умолчанию возможно большее количество полей, не забывая и об обеспечении требуемой производительности.

В Microsoft Dynamics AX 2012 все также поддерживается способ установки значений по умолчанию, существовавший в предыдущих версиях Microsoft Dynamics AX. Однако в Microsoft Dynamics AX 2012 в таблицы были добавлены механизмы отслеживания состояния полей (такие как *not set* и *defaulted*). Это означает, что есть возможность реализовывать логику задания значений по умолчанию прямо на классах таблиц. Тем самым появляется возможность ее использования не только в классах *Axd*<document>, но и в формах и где-либо еще. Отметим, что вследствие существования возможности реализации этой логики прямо на таблицах, класс *Ax*<Table> становится ненужным для реализации обычного кода задания значений по умолчанию.

Больше подробностей о реализации и модификации логики задания значений по умолчанию в Microsoft Dynamics AX 2012 и информацию об общем описании процесса разработки служб документов можно найти в разделе «AIF Document Services» в Microsoft Dynamics AX 2012 SDK (*http://msdn.microsoft.com/en-us/library/bb496530.aspx).*

Аспекты безопасности

Действия служб являются точками входа для внешних приложений, отправляющих запросы от лица пользователей. Как говорилось ранее, все методы X++, публикуемые как действия служб, должны быть отмечены X++-атрибутом *SysEntryPointAttribute*, указывающим, что вызов метода должен осуществляться в контексте вызывающего пользователя. Раз так, то должны быть проведены проверки в таблицах, к которым осуществляется доступ из вызванного метода. Помимо этого, к службам и их операциям должны также применяться все концепции основанной на ролях схемы разделения доступа.

Системные службы обычно доступны и вызываются в контексте вызывающего пользователя.

На вас, как на разработчике, лежит ответственность за указание атрибута *SysEntryPointAttribute* на всех создаваемых вами операциях произ-

вольных служб и за создание при необходимости соответствующих разрешений.

При генерации служб документов с помощью мастера служб документов все создаваемые им действия службы автоматически отмечаются *SysEntryPointAttribute*. Мастер также автоматически настроит все основные настройки безопасности для сгенерированного действия.

Совет. При использовании мастера служб документов AIF всегда проверяйте созданные им артефакты на соответствие вашим требованиям к безопасности и корректируйте их в случае несоответствия.

Публикация служб Microsoft Dynamics AX

После того как вы создали и реализовали вашу службу, для того чтобы внешние приложения смогли ее использовать, вам осталось ее опубликовать. Разработка службы и ее публикация представляют собой два разных, независимых друг от друга процесса.

С помощью AIF вы можете опубликовать службу Microsoft Dynamics AX посредством нескольких технологий транспортов. AIF также предлагает богатые возможности для настройки администраторами способа публикации интерфейсов службы. Эта глава ограничивается рассмотрением AIF в части публикации служб только через *основные порты интеграции*. Для более подробного изложения обратитесь к документации по администрированию служб Microsoft Dynamics AX 2012 на TechNet (*http://TechNet.microsoft.com/en-us/library/hh209600.aspx).* Там же вы найдете руководство по разработке, настройке и использованию концепций политик данных, преобразований, подстановки значений и стадии конвеерной обработки сообщений. (В локализации термин «конвеерная обработка сообщений», к сожалению, переведен как «компоненты стадии процесса продаж». – *Прим. перев.*)

Для нужд разработки и отладки вы можете прямо из AOT легко опубликовывать зарегистрированные произвольные службы и службы документов через основные порты интеграции. Вы можете также использовать группы служб, чтобы обеспечить развертывание и активацию служб сразу после запуска AOS, указав свойство *AutoDeploy* соответствующей группе служб. Это пригодится, если вам нужно начать использование службы без вмешательства администратора системы, к примеру, для автоматического

включения службы сразу после окончания развертывания Microsoft Dynamics AX 2012.

Для опубликования службы через основной порт интеграции, нужно добавить ее в *группу служб* в AOT. Затем вы можете опубликовать группу служб с конфигурацией по умолчанию, используя привязку *NetTcpBinding* из WCF, прямо из AOT. Подробнее см. в статьях на TechNet «Services, service operations, and service groups» (*http://technet.microsoft.com/en-us/library/gg731906.aspx*) и «Using basic integration ports» (*http://technet.microsoft.com/en-us/library/hh496420.aspx).*

Опубликованные через основные порты интеграции службы Microsoft Dynamics AX могут располагаться только прямо на сервере приложений (AOS). При использовании основных портов интеграции доступен ограниченный набор возможностей настройки служб. Из формы Входящие порты (Администрирование системы > Настройка > Services and Application Integration Framework > Входящие порты) вы можете включать и выключать основные порты интеграции, использовать *SvcConfigUtil* для изменения параметров настройки WCF и управлять ведением журнала для соответствующих портов.

Примечание. Если вам нужно опубликовать службу, используя привязку WCF, отличную от *NetTcpBinding*, или есть потребность в отправке односторонних сообщений (исходящих сообщений), или вы хотите больше возможностей контроля над обработкой сообщений, или, например, вам нужно еще и использовать преобразования XSLT, то необходимо создать расширенный порт. Создание *расширенных портов* интеграции доступно в формах Исходящие порты и Входящие порты, соответственно.

Рассуждения в этой главе, главным образом, основываются на предположении, что службы уже опубликованы через основные порты интеграции, если не оговорено иное. Подробности о публикации служб с использованием привязок, отличных от *NetTcpBinding,* посредством расширенных портов интеграции (например, через Message Queuing или адаптеры файловой системы), инструкции о создании портов для исходящих сообщений и остальных доступных возможностях настройки, можно найти в документации о службах и AIF для Microsoft Dynamics AX 2012 на TechNet (*http://technet.microsoft.com/en-us/library/gg731810.aspx).*

Использование служб Microsoft Dynamics AX

После того как вы опубликовали свои службы Microsoft Dynamics AX, внешние клиентские приложения получают возможность их использования и вызова предоставляемой ими бизнес-логики. Так, сразу после публикации службы *SalesOrderService* клиентские приложения могут использовать ее для создания и чтения заказов на продажу Microsoft Dynamics AX.

Данный раздел освещает аспекты использования служб Microsoft Dynamics AX из других клиентских приложений. Как говорилось ранее, описание ведется, основываясь на настройке служб через основные порты интеграции на AOS. Службы, опубликованные через них, становятся доступны путем использования Net.tcp.

Полное описание способов публикации служб Microsoft Dynamics AX, включая использование асинхронных адаптеров и сопутствующих технологий, можно найти в документации по службам и AIF для Microsoft Dynamics AX 2012 на TechNet (*http://technet.microsoft.com/en-us/library/gg731810.aspx*).

Пример WCF-клиента для службы *CustCustomerService*

Если вы хотите использовать службу Microsoft Dynamics AX, опубликованную через основной интеграционный порт, вам нужно сгенерировать прокси-классы по схеме WSDL той службы, которую будете использовать. Обычно это делается из вашего инструмента разработки (Microsoft Visual Studio) или путем использования инструментов командной строки, таких как *SvcUtil*.

После создания прокси-классов по WSDL и добавления их в среде разработки в проект следует написать код, который должен сделать следующие шаги:

- создать и проинициализировать параметры;
- если нужно, создать и проинициализировать контекст вызова;
- создать экземпляр прокси-класса службы;
- использовать нужное действие службы;
- оценить полученный ответ;
- обработать ошибки и исключения.

В этом разделе приведен пример, иллюстрирующий возможный вид кода для использования действия *find()* службы документов *CustCustomerService* (входящую в Microsoft Dynamics AX 2012).

В последующих примерах предполагается, что служба документов *CustCustomerService* опубликована в группе служб *MyServiceGroup* и уже создан проект Visual Studio. В Visual Studio уже добавлена ссылка на службу *MyServiceGroup,* созданная по схеме WSDL для основного порта интеграции *MyServiceGroup.* Более детальная информация о том, где Microsoft Dynamics AX публикует файлы WSDL, см. в разделе «Locating the WSDL for Services» в Microsoft Dynamics AX 2012 SDK (*http://msdn.microsoft.com/en-us/library/gg843514.aspx*).

Приведенные ниже фрагменты кода на C# показывают шаги использования операции *find* службы документов *CustCustomerService Microsoft Dynamics AX.*

Во-первых, вам нужно создать и проинициализировать параметры, требующиеся для вызова. Операция *find*-службы принимает два входных параметра: необязательный контекст вызова и критерии запроса, указывающие, какие записи клиентов должны быть возвращены. В следующем примере выбираются все записи клиентов в компании CEU с кодами клиента, большими или равными 4000.

```
// создание экземпляра и инициализация параметров

// параметр: контекст вызова
    MyServiceGroup.CallContext cc = new MyServiceGroup.CallContext();
    cc.Company = "CEU";

    // параметр: критерий запроса
    MyServiceGroup.QueryCriteria qc = new MyServiceGroup.QueryCriteria();
    MyServiceGroup.CriteriaElement[] qe = { new MyServiceGroup.CriteriaElement() };

qe[0].DataSourceName = "CustTable";
qe[0].FieldName = "AccountNum";
qe[0].Operator = MyServiceGroup.Operator.GreaterOrEqual;
qe[0].Value1 = "4000";
qc.CriteriaElement = qe;
```

Совет. Можно использовать объект *CallContext* для выполнения запроса в контексте, отличном от контекста по умолчанию (когда *CallContext* в запросе равен *null* или не указан). В объекте *CallContext* можно указать компанию, язык и прочее.

Далее вам нужно создать экземпляр прокси-класса службы и вызвать ее операцию *find*, которая выполнит запрос и вернет подходящие записи:

```
// создание экземпляра прокси-класса службы
MyServiceGroup.CustomerServiceClient customerService =
    new MyServiceGroup.CustomerServiceClient();

// вызов действия службы find()
MyServiceGroup.AxdCustomer customer = customerService.find(cc, qc);
```

И, наконец, осталось оценить ответ сервера, который может быть либо результатом запроса, либо исключением и сообщениями об ошибке:

```
// обработка ошибок (нужно правильно обработать исключения)
if (null == customer)
{
    // обработка ошибок...
}

// оценка ответа
MyServiceGroup.AxdEntity_CustTable[] custTables = customer.CustTable;
if (null == custTables || 0 == custTables.Length)
{
    // обработка пустого ответа...
}
foreach (MyServiceGroup.AxdEntity_CustTable custTable in custTables)
{
    custTable...
}
```

Примечание. Обработка исключений и прочие практики хорошего тона разработки клиентов веб-сервисов здесь были опущены для упрощения примера.

Дадим несколько советов для работы со службами документов.

- Многие службы документов поддерживают и действия *find*() (возвращающие все *Axd*-документы в возвращаемом наборе данных), и дей-

ствия *findKeys* (которые возвращают только ключи сущностей *Axd*-документов). Если вы ожидаете, что ответное сообщение на вызов *find* будет очень большим, есть смысл использовать *findKeys*, чтобы выбрать только ключи сущностей. Затем, к примеру, вы можете реализовать постраничную выборку (выборку порциями) для получения соответствующих *Axd*-документов частями более приемлемых размеров.

- При разработке новых служб обычно приходится включать журналирование на стороне сервера. Чтобы сделать это, откройте форму Входящие порты, выключите порт интеграции, через которую опубликована группа служб, содержащая вашу службу, включите ведение журнала в разделе Устранение неполадок формы и снова включите ваш порт интеграции.
- Если действие службы возвращает большие ответные сообщения, вам, скорее всего, понадобится изменить настройки по умолчанию в файлах конфигурации WCF как для службы, так и для клиента. Умолчания в конфигурации службы и клиент допускают сообщения размером до 65536 байт. Максимальные размеры сообщения и буфера задаются параметрами *maxReceivedMessageSize и maxBufferSize* в секции привязок (bindings) стандартного конфигурационного файла WCF. Перед изменением этих параметров обратитесь к документации разработчика .NET Framework для понимания последствий этих изменений и понимания корректных значений этих параметров. .NET Framework Developer Center находится по адресу: *http://msdn.microsoft.com/en-us/netframework/aa496123*.

Все остальные действия произвольных служб или служб документов могут быть использованы подобными же путями. Еще больше информации и примеров кода можно найти в Microsoft Dynamics AX 2012 SDK по адресу: *http://msdn.microsoft.com/en-us/library/aa496079.aspx*.

Использование системных служб

В отличие от произвольных служб и служб документов, системные службы публикуются автоматически (на сервере приложений посредством *NetTcpBinding)* и готовы для использования клиентскими приложениями сразу после запуска AOS.

Как все службы Microsoft Dynamics AX, системные службы публикуют метаданные в форме файлов WSDL, которые можно использовать для создания прокси-классов (как в предыдущих примерах). Однако, в то вре-

мя как служба информации о сеансе пользователя публикуется исключительно через порт интеграции (*UserSessionService)*, подобно произвольным службам и службам документов, для службы запросов и службы метаданных не существует порта интеграции.

Чтобы показать пример работы со службой метаданных и службой запроса, приведем пример кода, делающего следующее.

- Получение метаданных запроса (определения запроса *MyQuery)* из Microsoft Dynamics AX, используя службу метаданных.
- Преобразование метаданных запроса из контракта данных, используемого службой метаданных, в контракт данных, используемых службой запроса. Несмотря на то что оба контракта данных имеют идентичную структуру (см. метод *ConvertContract* в примере), преобразование необходимо выполнить.
- Добавление ограничения в объект метаданных; в этом случае выбираются все строки со значением большим, чем 1996, в столбце *Year.*
- Выполнение преобразованного определения запроса через службу запросов.

В вашем коде .NET эти шаги можно проделывать в том же порядке, что и в приводимом ниже примере. Предположим, что уже создан проект Visual Studio и в него посредством файлов *WSDL* добавлены ссылки на *MetadataService* и *QueryService* для службы метаданных и службы запросов.

Более детальную информацию о том, где Microsoft Dynamics AX публикует файлы WSDL, смотрите в разделе «Locating the WSDL for Services» в Microsoft Dynamics AX 2012 SDK (*http://msdn.microsoft.com/en-us/library/gg843514.aspx*).

```
// создание прокси
var metadataClient = new MetadataServiceReference.AxMetadataServiceClient();
var queryClient = new QueryServiceReference.QueryServiceClient();

// загрузка метаданных запроса
MetadataService.QueryMetadata[] query =
    metadataClient.GetQueryMetadataByName(new string[] { "MyQuery" });

// преобразование метаданных запроса
QueryService.QueryMetadata convertedQuery = ConvertContract
    <MetadataService.QueryMetadata, QueryService.QueryMetadata>(query);

// добавление к метаданным запроса ограничения
```

```
QueryDataRangeMetadata range = new QueryDataRangeMetadata()
{
    Enabled = true,
    FieldName = "Year",
    Value = ">1996"
};
convertedQuery.DataSources[0].Ranges = new QueryRangeMetadata[] { range };

// включение постраничной выборки (возвращается по 3 записи или менее)
QueryService.Paging paging = new QueryService.ValueBasedPaging();
((QueryService.ValueBasedPaging)paging).RecordLimit = 3;

// создание прокси службы
QueryService.QueryServiceClient queryService = new QueryService.QueryServiceClient();

// выполнение преобразованного запроса с ограничением, получение результата выборки
в .NET dataset
System.Data.DataSet ds = queryClient.ExecuteQuery(convertedQuery, ref paging);
```

Несмотря на то что определение *QueryMetadata* идентично в службе запросов и в службе метаданных, генератор прокси-классов создал идентичные классы в двух разных областях имен, по одному для каждой службы. Метод *ConvertContract*, реализующий преобразование двух контрактов одинаковой структуры с использованием общих типов (generics), будет выглядеть подобно коду, приводимому ниже:

```
static TTargetContract ConvertContract<TSourceContract, TTargetContract>
        (TSourceContract sourceContract)
            where TSourceContract : class
            where TTargetContract : class
{
    TTargetContract targetContract = default(TTargetContract);
    var sourceSerializer = new DataContractSerializer(typeof(TSourceContract));
    var targetSerializer = new DataContractSerializer(typeof(TTargetContract));
    using (var stream = new MemoryStream())
    {
        sourceSerializer.WriteObject(stream, sourceContract);
        stream.Position = 0;
        targetContract = (TTargetContract)targetSerializer.ReadObject(stream);
    }
    return targetContract;
}
```

Как говорилось ранее, для переопределения контекста по умолчанию (включающего компанию и язык), в котором выполняется запрос, используется *CallContext*. Вообще говоря, *CallContext* необязателен для всех запросов к службе; если он в запросе не указан, то запрос будет выполнен с использованием значения по умолчанию для свойств объекта *CallContext*.

В Microsoft Dynamics AX 2012 файлы WSDL службы запросов и службы метаданных не содержат определений XML-схемы для *CallContext*. Следовательно, прокси-классы, созданные по файлам WSDL для этих служб, не включают прокси-класса для *CallContext*. Но тем не менее *CallContext* все равно может быть использован для рассматриваемых служб так же, как и для всех остальных служб. Чтобы использовать *CallContext* в запросах к службе метаданных или к службе запросов, вам придется добавить ссылку на службу (такую как *UserSessionService*) в порт интеграции, что приведет к генерации нужных прокси-классов для *CallContext*. Тогда вы сможете создать, проинициализировать объект *CallContext* и добавить его в свой запрос, как показано в следующем коде:

```
// получаем OperationContextScope (см. документацию по WCF)
using (System.ServiceModel.OperationContextScope ocs =
        new System.ServiceModel.OperationContextScope((queryService.InnerChannel))) {

    // создаем и инициализируем CallContext (используя класс от другой службы)
    CustomerService.CallContext callContext = new CustomerService.CallContext();
    callContext.Company = "CEU";

    // явно задаем вставку заголовка "CallContext" к множеству заголовков исходящих
    // сообщений
    System.ServiceModel.Channels.MessageHeaders messageHeadersElement =
    System.ServiceModel.OperationContext.Current.OutgoingMessageHeaders;
    messageHeadersElement.Add(
        System.ServiceModel.Channels.MessageHeader.CreateHeader(
            "CallContext",
            "http://schemas.microsoft.com/dynamics/2010/01/datacontracts",
            callContext));

    // включаем постраничную выборку (по 3 записи или менее)
    QueryService.Paging paging = new QueryService.ValueBasedPaging();
    ((QueryService.ValueBasedPaging)paging).RecordLimit = 3;

    // создаем прокси службы
    QueryService.QueryServiceClient queryService = new QueryService.QueryServiceClient();
```

```
    // вызываем службу, используя CallContext
    System.Data.DataSet ds = queryService.ExecuteStaticQuery("MyQuery", ref paging);
}
```

Примечание. Служба запросов возвращает свои результаты порциями, определенными в обязательном параметре *paging*. Алгоритмы постраничной выборки исходят из предположения, что запросы построены на таблицах с отношениями в режиме выборки *FetchMode*, который установлен в 1:1 (в свойствах источника данных AOT). Служба запросов выдаст сообщение об ошибке, если в запросе будут использоваться отношения с *FetchMode,* установленным в 1:*n*.

Обратитесь к документации по системе для получения детальной информации по *CallContext* или о возможностях системных служб.

Обновление бизнес–документов

Во многих случаях требуется обновлять данные в уже существующих *Axd*-документах – или чтобы добавить строки в заказ на продажу, или чтобы обновить адрес клиента. Посредством действия *update*-служб документов предоставляется поддержка различных семантик обновления документов: *полное обновление* и *частичное обновление.*

В приводимых ниже примерах предполагается, что стандартная служба документов *SalesSalesOrderService* уже добавлена в группу служб *MyServiceGroup* и опубликована посредством основного порта интеграции *MyServiceGroup.*

Полное обновление

Полное обновление – это поведение по умолчанию службы документов. Чтобы использовать этот режим, ваш код в клиентском приложении должен сделать следующее:

- прочитать документ;
- внести в него нужные изменения;
- отправить обновленный документ обратно на сервер;
- обработать ошибки в случае их появления.

Следующий код C# представляет собой концептуальный пример того, как осуществить полное обновление существующего заказа на продажу:

```
// создание и инициализация callContext, entityKeys, serviceOrderService
MyServiceGroup.EntityKey[] entityKeys = ...
MyServiceGroup.CallContext callContext = ...
MyServiceGroup.SalesOrderServiceClient salesOrderService = ...
...

// чтение заказа(ов) на продажу (включая хэши документа(ов)) с использованием entityKeys
MyServiceGroup.AxdSalesOrder salesOrder =
    salesOrderService.read(callContext, entityKeys);

// обработка ошибок, исключений; обработка заказа на продажу, обновление данных
...
// сохранение изменений на сервере (обработка ошибок не показана)
salesOrderService.update(callContext, entityKeys, salesOrder);
```

Частичное обновление

Во многих случаях полное обновление записей – не самый эффективный путь. Представьте себе большой заказ на продажу с множеством строк (заказ, имеющий более 1000 строк – это не редкость). Если вы будете использовать полное обновление, вам придется выбрать весь заказ со всеми его строками, внести изменения в одну из них, которую вы хотите обновить, и затем отправить его целиком, включая и все, оставшиеся неизменными строки. Такая операция будет весьма затратной, особенно если учесть логику валидации и задания значений по умолчанию, вызываемую на сервере для каждой строки заказа.

Вместо осуществления полного обновления вы можете сделать частичное. Частичное обновление использует то же действие службы, что и полное: *update*. Однако в частичных обновлениях вы можете отправить неполные документы, содержащие только измененные (добавленные или удаленные) данные. Для подчиненных элементов документы, отправляемые в запросах частичного обновления, содержат инструкции по их обработке, указывающие, как обрабатывать каждую (подчиненную) запись, включенную в частичный документ, чтобы избежать неоднозначности. Следовательно, процесс обновления документов с использованием частичного обновления содержит один добавочный шаг по сравнению с полным обновлением:

- чтение документа;
- внесение изменений в документ (чтобы получить преимущество от частичного обновления, убедитесь, что вы отправляете обратно на

сервер только те поля, которые являются обязательными, либо измененные вами);

- явный запрос режима частичного обновления и указание инструкций по обработке;
- отправка обновленного документа в запросе обновления;
- обработка ошибок в случае их появления.

Приведенный ниже код содержит концептуальный пример проведения частичного обновления заказа на продажу:

```
// создание и инициализация callContext, entityKeys, serviceOrderService
MyServiceGroup.EntityKey[] entityKeys = ...
MyServiceGroup.CallContext callContext = ...
MyServiceGroup.SalesOrderServiceClient salesOrderService = ...
...
// чтение заказа(ов) на продажу (включая хэши документа(ов)) с использованием entityKeys
MyServiceGroup.AxdSalesOrder salesOrder =
    salesOrderService.read(callContext, entityKeys);

// обработка ошибок, исключений; обработка заказа на продажу, обновление данных
...
// например: обновим первую строку заказа и отметит ее для частичного обновления
AxdEntity_SalesTable[] salesTables = salesOrder.SalesTable;
salesOrder.SalesTable = new AxdEntity_SalesTable[] { salesTables[0] };
// директива на уровне документа, запрашивающая частичное обновление
salesOrder.SalesTable[0].action = AxdEnum_AxdEntityAction.update;
// директива на уровне таблицы, запрашивающая удаление первой строки заказа
AxdEntity_SalesLine[] salesLines = salesOrder.SalesTable[0].salesLine;
salesOrder.SalesTable[0].SalesLine = new AxdEntity_SalesLine[] { salesLines[0] };
salesOrder.SalesTable[0].SalesLine[0].action = AxdEnum_AxdEntityAction.delete;

// удаление подчиненных не обновлявшихся источников данных (DocuRefHeader, и.т.п.)
// из salesTable
...

// сохранение изменений на сервере (обработка ошибок не показана)
salesOrderService.update(callContext, entityKeys, salesOrder);
```

Примечание. В XML-сообщениях запроса эти инструкции по обработке отражены появлением XML-атрибута *action*. Это верно как для XML-сообщений, отправляемых в асинхронные адап-

теры, так и для SOAP-сообщений, отправляемых к синхронным службам WCF. Подробнее см. в разделе «AIF Document Services» в Microsoft Dynamics AX 2012 SDK (*http://msdn.microsoft.com/en-us/library/bb496530.aspx).*

Оптимистичный контроль совпадений

Инфраструктура служб опирается на оптимистичный контроль совпадений (optimistic concurrency control, OCC) при появлении конкурентных множественных запросов на обновление. Чтобы иметь возможность определить, был ли изменен документ с момента последнего его чтения, и во избежание непреднамеренной перезаписи таких изменений, инфраструктура служб использует хэши документов для различения его версий.

Хэши документа вычисляются для конкретного экземпляра документа на основе его содержания. Они основаны не только на источнике данных корневого уровня (такого как заголовок заказа на продажу), но и на основе содержимого всех присоединенных источников данных (таких как строки заказа). Другими словами, если поле в любой таблице, входящей в бизнес-документ, будет изменено, то изменится и хэш документа.

Чтобы получить хэш какого-то определенного бизнес-документа, ваш код сначала должен прочитать документ. Затем вы можете использовать хэш, который был возвращен в документе для последующего запроса на обновление.

Совет. Кэширование документа в течение длительного времени внутри клиента службы без его обновления увеличивает вероятность отклонения системой запросов на обновление из-за встречных обновлений других клиентских приложений.

Асинхронный вызов произвольных служб

Так как публикация сервиса отделена от его разработки, то через поддерживаемые транспортные механизмы могут быть опубликованы как произвольные службы, так и службы документов. Выражаясь более точно, операции произвольных служб и служб документов могут быть опубликованы синхронно (например, используя протоколы Net.tcp или HTTP) через основные порты интеграции, как в предыдущих примерах, или же асинхронно (например, используя адаптер файловой системы или Message

Queuing), через расширенные порты интеграции. Администраторы системы имеют широкий выбор возможностей настройки группировки служб и режимов их публикации, журналирования и прочего прямо в ходе эксплуатации системы. Больше информации про публикацию служб через расширенные порты интеграции можно найти в документации по службам и AIF Microsoft Dynamics AX 2012 на TechNet (*http://technet.microsoft.com/en-us/library/gg731810.aspx*).

При синхронном использовании служб Microsoft Dynamics AX типовой сгенерированный прокси службы занимается вопросами создания и использования XML, которым обмениваются приложения клиента и Microsoft Dynamics AX. Однако при использовании служб Microsoft Dynamics AX через асинхронные транспорты вы должны быть уверены, что сообщения запросов отвечают определениям XML-схем для AIF-конверта сообщений и бизнес-документа, как ожидается инфраструктурой служб Microsoft Dynamics AX. Более детальную информацию про то, как получить определения схемы XML (XSD) для конвертов сообщений AIF, можно найти в разделе «AIF Messages» в Microsoft Dynamics AX 2012 SDK (*http://msdn.microsoft.com/en-us/library/aa627117.aspx*).

Приводимый далее пример кода показывает образец сообщения XML, которое может быть отправлено асинхронно из клиентского приложения в Microsoft Dynamics AX для использования действия *MyService.HelloWorld(MyParam in)* произвольной службы, которая обсуждалась в предыдущем примере (см. раздел «Произвольные службы» ранее в этой главе). Он иллюстрирует соответствие названия службы, названия ее действия и структуры входных параметров соответствующим элементам XML в сообщении запроса. Он также показывает, как можно указать контекст исполнения запроса: путем указания элемента *Header*, который включает те же самые свойства, что и *CallContext* в случае синхронных интерфейсов служб.

```
<?xml version="1.0" encoding="UTF-8"?>
<Envelope
    xmlns="http://schemas.microsoft.com/dynamics/2011/01/documents/Message">
  <Header>
    <!-- Действие службы: "MyService.HelloWorld(MyParam)" -->
    <Company>CEU</Company>
    <Action>http://tempuri.org/MyService/HelloWorld</Action>
  </Header>
<Body>
    <MessageParts
```

```
        xmlns="http://schemas.microsoft.com/dynamics/2011/01/documents/Message">

        <!-- Сложный входной параметр: "MyParam in" -->
        <in xmlns="http://tempuri.org"
            xmlns:i="http://www.w3.org/2001/XMLSchema-instance"
        xmlns:b="http://schemas.datacontract.org/2004/07/Dynamics.Ax.Application">

                <!-- Свойства сложного входного параметра: "in.b" -->
                <b:intParm>0</b:intParm>
            </in>
        </MessageParts>
    </Body>
</Envelope>
```

Примечание. Чтобы запустить этот пример, вам нужно создать расширенный порт интеграции, настроенный на асинхронное получение файлов. Этот порт интеграции должен публиковать действие службы *MyService.HelloWorld*.

Таким образом, в этой главе мы рассмотрели, как функциональность Microsoft Dynamics AX публикуется через службы для использования их внешними клиентскими приложениями и как, собственно, эти внешние приложения используют эти службы. Но что делать, если требуется организовать одностороннюю отправку данных из Microsoft Dynamics AX? В следующих двух разделах мы обсудим, как использовать инфраструктуру отправки односторонних сообщений для асинхронной односторонней передачи.

Инфраструктура отправки односторонних запросов

AIF предоставляет интерфейсы программирования и инфраструктуру для использования служб Microsoft Dynamics AX для целей отправки односторонних сообщений. Клиент Microsoft Dynamics AX имеет возможности, такие как кнопка Отправить… на ряде форм, позволяющая отправить пользователям деловые документы (например накладные) в виде односторонних сообщений через исходящий порт интеграции. О том, как настроить исходящие порты интеграции, см. в документации по службам и AIF Microsoft Dynamics AX 2012 в TechNet (*http://technet.microsoft.com/en-us/library/gg731810.aspx*).

Microsoft Dynamics AX не полагается на внешние определения схем документов, предоставляемые удаленными приложениями-получателями. Она использует собственный формат – тот самый XSD, основанный на классе *Axd*<Document>, который используется в качестве контрактов данных для опубликованных служб Microsoft Dynamics AX. Реализация односторонних сообщений опирается на следующие два шага.

- Реализация триггера для передачи (разрабатываемая часть).
- Настройка расширенного исходящего порта интеграции для отправки документов (часть администрирования).

Реализация триггера для передачи сообщения

Вы можете реализовать триггер для передачи сообщения, используя API AIF Send или API AxdSend.

API AIF Send

Интерфейс программирования Send имеет набор методов для отправки односторонних сообщений из Microsoft Dynamics AX посредством портов интеграции, через которые потребители могут получить эти сообщения. Данный интерфейс отправляет сообщения по одному: тело сообщения содержит XML, сгенерированный вызовом действия службы *read* службы документа AIF, указанной *serviceClassId* (оно должно указывать класс, унаследованный от *AifDocumentService)* с параметром *entityKey*.

Чтобы увидеть работающий пример, использующий этот **API, посмотрите** на код метода *clicked кнопки* SendXmlOriginal на форме *CustInvoiceJournal*. Методы API определены в классе *AifSendService* и включают метод *submitDefault*.

```
public static void submitDefault(
    AifServiceClassId serviceClassId,
    AifEntityKey entityKey,
    AifConstraintList constraintList,
    AifSendMode sendMode,
    AifPropertyBag propertyBag = connull(),
    AifProcessingMode processingMode = AifProcessingMode::Sequential,
    AifConversationId conversationId = #NoConversationId
)
```

Используя два необязательных параметра из указанной сигнатуры, *processingMode* и *conversationId*, вы можете использовать преимущества параллельной обработки сообщений для асинхронных адаптеров.

- ***processingMode.*** Указывает, может ли сообщение быть перенесено из очереди обработки исходящих сообщений AIF в очередь шлюза AIF независимо от других (*AifProcessingMode::Parallel*) или же для всех сообщений (*AifProcessingMode::Sequential*) должен соблюдаться порядок first-in-first-out (FIFO).
- ***conversationId.*** При его указании AIF переносит сообщение из очереди обработки исходящих сообщений AIF в очередь шлюза AIF в порядке FIFO относительно остальных сообщений с тем же *conversationId.* Соблюдение порядка относительно других сообщений, с другими *conversationId*, не гарантируется.

API AxdSend

API AxdSend предоставляет функциональность отправки односторонних сообщений. Выбор исходящего порта интеграции, через который будут отправлены документы, определяется пользователем во время его каждодневной работы. Если требуется отправка нескольких документов, то конкретные документы из них также выбираются пользователем. Эта возможность доступна для нескольких служб документов Microsoft Dynamics AX, таких как *AxdPricelist* и *AxdBillsOfMaterials*.

Инфраструктура *AxdSend* содержит готовые диалоговые окна для выбора портов интеграции и списков сущностей, с помощью которых генерируются XML-документы, содержащие несколько записей одновременно. Но можно также использовать возможности инфраструктуры по поддержке специфичных диалоговых окон для документов, требующих указания пользователем дополнительной информации, отсутствующей в стандартной реализации диалоговых окон.

Стандартная реализация диалоговых окон включает ниспадающий список выбора порта интеграции и, возможно, кнопку Выбор для открытия стандартной формы запроса. Выводимый в ней запрос определяется на основе класса *Axd*<Document>, указанного вызывающим объектом. В AIF может быть настроено много портов интеграции, но только некоторые из них могут получить текущий документ. Ниспадающий список показывает только те порты интеграции, которые доступны для данного доку-

мента, согласно ограничениям, настроенным на действие службы *read* для текущего документа.

Для поддержки нового документа в инфраструктуре требуется минимум кода. Если для нового документа достаточно дать пользователю выбрать порт интеграции и заполнить диапазоны запроса, то большая часть функциональности уже есть в инфраструктуре и ее не нужно кодировать.

Пример диалогового окна в инфраструктуре *AxdSend* показан на рис. 12-3.

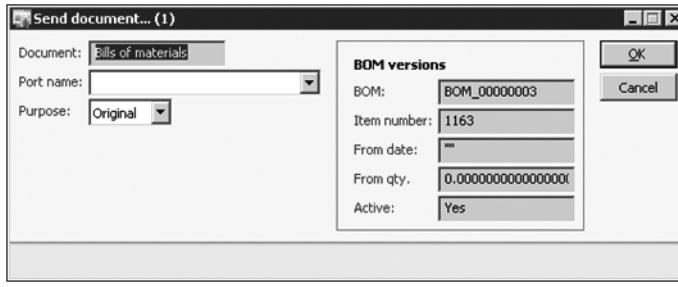

Рис. 12–3. Диалоговое окно Отправить... для спецификации

Если для какого-то *Axd*<document> требуется более специфичное диалоговое окно, вы наследуете класс *AxdSend* и создаете нужное взаимодействие с пользователем в методе определения диалогового окна. В следующем примере кода в диалоговое окно добавлено еще одно поле. Вам остается добавить одну строку кода, чтобы вызвать *parmShowDocPurpose* из класса *AxdSend,* чтобы это поле появилось в диалоговом окне:

```
static public void main(Args args)
{
    AxdSendBillsOfMaterials     axdSendBillsOfMaterials;
    AifConstraintList           aifConstraintList;
    AifConstraint               aifConstraint;
    BOMVersion                  bomVersionRecord;

    axdSendBillsOfMaterials     = new AxdSendBillsOfMaterials();
    aifConstraintList           = new AifConstraintList();
    aifConstraint               = new AifConstraint();

    aifConstraint.parmType(AifConstraintType::NoConstraint);
    aifConstraintList.addConstraint(aifConstraint);
```

```
    if (args && args.record().TableId == tablenum(BOMVersion))
    {
        bomVersionRecord = args.record();
        axdSendBillsOfMaterials.parmBOMVersion(bomVersionRecord);
    }

    // добавлена строка кода, заставляющая появиться поле в диалоговом окне
    axdSendBillsOfMaterials.parmShowDocPurpose(true);

    axdSendBillsOfMaterials.sendMultipleDocuments(
        classnum(BomBillsofMaterials),
        classnum(BomBillsofMaterialsService),
        AifSendMode::Async,
        aifConstraintList);
}
```

В инфраструктуре *AxdSend* не поддерживаются сортировки и фиксирована структура запроса, чтобы обеспечить совпадение получаемого запроса с запросом, заданным в инфраструктуре XML-документа. Из-за необходимости такого соответствия, в классе *AxdSend* ограничиваются возможности сортировки и изменения структуры запроса.

Диалоговое окно запроса показывает только поля в таблицах верхнего уровня, из-за использования запросов с типом внешнего объединения (outer join). Получаемый набор данных может отличаться от того, что ожидает получить пользователь. Например, фильтры на подчиненных источниках данных ограничат только их, но не их родительские источники. Эти ограничения, налагаемые на интерфейс пользователя, связаны с необходимостью соответствия ограничениям, налагаемым на запрос при использовании действия *find* службы документа.

Настройка механизмов передачи

Для более детального ознакомления с настройкой расширенных исходящих портов интеграции и других административных действий, относящихся к асинхронной отправке односторонних сообщений с использованием инфраструктуры отправки Microsoft Dynamics AX, обратитесь к документации по службам и AIF Microsoft Dynamics AX 2012 в TechNet (*http://technet.microsoft.com/en-us/library/gg731810.aspx*).

Использование внешних веб-служб

Веб-службы – это популярный и широко известный способ интеграции приложений, размещенных внутри периметра организации или в интранете.. Примерами таких приложений могут быть приложения для управления ресурсами предприятия (ERP), приложения CRM, офисные приложения, такие как Office, и т.д.

Интеграция приложений со сторонними веб-службами через Интернет также стала распространенным и зачастую предпочтительным способом быстрого добавления новой функциональности в сложные приложения. Веб-службы могут варьироваться от простой проверки корректности адреса или проверки кредитной карты до более сложных расчетов налогов или услуг казначейства.

Подобно отправке односторонних асинхронных сообщений с использованием инфраструктуры Send Microsoft Dynamics AX, вы можете использовать Microsoft Dynamics AX для отправки запросов к внешним веб-службам. Другими словами, доступно использование внешних веб-служб. Поскольку взаимодействие с ними означает достаточно тесную связь с соответствующей веб-службой (и обычно включает создание прокси для нее), а также из-за наличия в Visual Studio богатого набора инструментов для построения таких интеграций, вам нужно создать проект Visual Studio и сформировать динамическую библиотеку .NET (DLL), содержащую код, использующий внешнюю веб-службу. Затем вы добавляете эту библиотеку как ссылку в Microsoft Dynamics AX и пишете код X++, вызывающий методы этой библиотеки .NET.

Примечание. Инфраструктура служб Microsoft Dynamics AX не содержит никаких инструментов, ориентированных на написание кода, использующего внешние веб-службы. Концепция ссылок на службы, которая существовала в Microsoft Dynamics AX 2009, была исключена из Microsoft Dynamics AX 2012 и соответствующий узел AOT более не существует.

Аспекты производительности

Для достижения требуемого уровня производительности в конкретном сценарии реализации Microsoft Dynamics AX чрезвычайно важно планирование и оценка мощности используемых аппаратных средств. Для по-

лучения руководства по корректной оценке ваших разворачиваемых ресурсов обратитесь к «*Microsoft Dynamics AX Implementation Planning Guide*» (*http://www.microsoft.com/en-us/download/details.aspx?id=4007*).

По умолчанию порты интеграции последовательно обрабатывают все полученные сообщения. Это верно как для исходящих, так и для входящих сообщений запросов. Чтобы увеличить число обрабатываемых сообщений, вы можете использовать возможности параллельной обработки AIF наряду с развертыванием дополнительных серверов приложения. Для подробной информации о настройке входящих портов для параллельной обработки и использовании расширений интерфейса программирования AIF Send обратитесь к разделу системного администратора «Services and AIF Operations» в документации Microsoft Dynamics AX 2012 на TechNet (*http://technet.microsoft.com/en-us/library/gg731830.aspx*).

Обратите внимание, что для синхронных служб WCF обработка запросов, по существу, ведется параллельно.