# Глава 18

# Инфраструктура сервера пакетных заданий

**В этой главе**

- Введение
- Пакетная обработка в Microsoft Dynamics AX 2012
- Создание и выполнение пакетного задания
- Управление процессом выполнения пакетного задания
- Отладка пакетных задач

## Введение

Инфраструктура пакетных заданий Microsoft Dynamics AX 2012 представляет собой серверную асинхронную среду выполнения задач. Она позволяет пользователям параллельно выполнять асинхронные задачи на множестве серверов приложений (AOS). В этой версии системы инфраструктура пакетных заданий была еще более расширена по сравнению с Microsoft Dynamics AX 2009. Помимо прочих улучшений, эта инфраструктура выполняет весь работающий в ней код на промежуточном языке .NET (CIL) и дает администраторам и разработчикам системы расширенные возможности управления пакетными заданиями, обеспечивая лучшую производительность и надежность их выполнения.

Microsoft Dynamics AX 2012 включает в себя несколько инструментов поддержки среды пакетных заданий. Форма Пакетные задания предлагает администраторам системы большую гибкость в создании, настройке и выполнении сложных пакетных заданий. Вдобавок эта форма обеспечивает возможность добавления нескольких пакетных задач в одно пакетное задание с заданием зависимостей между добавляемыми задачами. Доработанный интерфейс программирования пакетных заданий (API) дает разработчикам X++ расширенные средства управления сложными пакетными заданиями вкупе с возможностью обработки пакетных заданий прямо из бизнес-логики приложения.

В этой версии системы также была представлена новая инфраструктура SysOperation. Она позволяет создавать логику приложения, допускающую работу как в интерактивном режиме, так и в инфраструктуре пакетных заданий. Она является модернизацией инфраструктуры RunBase и допускает большую гибкость в создании новых пакетных заданий. Больше информации об инфраструктуре SysOperation можно найти в главе 14.

# Пакетная обработка в Microsoft Dynamics AX 2012

Пакетная обработка представляет собой неинтерактивную технику обработки задач, в которой пользователи создают единицы обработки в виде заданий нужных им типов. Пакетная обработка имеет ряд важных преимуществ: она позволяет пользователям создавать расписание пакетных заданий и задавать условия начала их выполнения, добавлять задачи в очередь и настраивать их для последующей автоматической обработки на пакетном сервере. После завершения их выполнения пакетный сервер журнализирует любые произошедшие ошибки и рассылает оповещения. Пакетное задание может включать в себя печать отчетов, закрытие склада или проведение периодического обслуживания. Назначая пакетное задание, выполняющее подобные тяжелые обработки, на часы, когда система разгружена, можно избежать замедления работы системы в рабочее время.

В табл. 18-1 описано, как стандартные концепции пакетной обработки представлены в Microsoft Dynamics AX. Эти концепции будут детально рассмотрены по ходу изложения этой главы.

**Табл. 18–1.** Концепции пакетной обработки в Microsoft Dynamics AX 2012

| Концепция | Описание |
|---|---|
| Пакетная задача | Минимальная единица объема работы, которая может быть выполнена в инфраструктуре пакетных заданий. Это класс, имеющий возможность пакетного выполнения и содержащий бизнес-логику, выполняющую некоторое действие. Классы Microsoft Dynamics AX, используемые в качестве пакетных задач, рассчитаны для работы на сервере. Эти задачи могут выполняться автоматически как часть пакетного задания на AOS. В этой версии системы поддержка заданий, выполняющихся на клиенте, ограничена. Для полного использования всех возможностей Microsoft Dynamics AX 2012 рекомендуется использовать серверные пакетные задания. Больше сведений об этом содержится в разделе «Создание класса с поддержкой пакетного выполнения» далее в этой главе |

**Табл. 18–1.** Концепции пакетной обработки в Microsoft Dynamics AX 2012 (*окончание*)

| Концепция | Описание |
|---|---|
| Пакетное задание | Законченный процесс, достигающий некоторой цели, например печать отчета или выполнение процесса закрытия склада. Пакетное задание состоит из одной или нескольких пакетных задач |
| Группа пакетов | Логическая категоризация пакетных задач, позволяющая администраторам системы указывать, на каком сервере приложений будет выполняться конкретная задача. Задачи, которые не назначены явно какой-либо пакетной группе, назначаются *пустой* группе (группе по умолчанию) |
| Сервер обработки пакетных заданий | Экземпляр сервера приложений, обрабатывающий пакетные задания. О серверах приложений можно прочитать в главе 1. За сведениями о настройке экземпляра AOS для работы в качестве пакетного сервера обратитесь к разделу «Настройка пакетного сервера» далее в этой главе |

## Цели использования инфраструктуры пакетных заданий

Предприятия используют инфраструктуру пакетных заданий для реализации разнообразных сценариев выполнения асинхронных операций.

Обычно пакетные задания создаются для удовлетворения следующих потребностей.

- **Обеспечение гибкого расписания.** Инфраструктура пакетных заданий может выполнять периодические задачи на основе заданного расписания, такие как очистка данных или обработка накладных. К примеру, чтобы запустить обработку накладных в конце каждого месяца, вы можете настроить повторяющееся пакетное задание, работающее в полночь последнего рабочего дня каждого месяца. Пакетная инфраструктура автоматически выберет это задание и обработает ожидающие обработки накладные согласно указанному расписанию.
- **Управление порядком выполнения задач.** С помощью инфраструктуры пакетных заданий вы можете создать документооборот или выполнить сложный процесс преобразования данных именно в той последовательности, которую укажете. Также возможно создание зависимостей между задачами и создание дерева зависимостей, обеспечивающее последовательное выполнение определенных задач, тогда как остальные будут выполняться параллельно.

- **Условная обработка.** Деревья решений могут быть полезны в реализации надежного способа обработки данных. Разработчики или администраторы системы могут задавать зависимости между задачами таким образом, что результат выполнения одних задач (успех или неудача) будет влиять на то, какие задачи будут выполнены далее. (Рис. 18-4 показывает пример дерева зависимостей.) Администраторы системы могут также настроить оповещения, рассылаемые при неудачном завершении задания.
- **Улучшение производительности с использованием параллельной обработки.** Инфраструктура пакетных заданий позволяет использовать преимущество многопоточности, тем самым обеспечивая полное использование возможностей современных процессоров. Это чрезвычайно важно для таких процессов, как закрытие склада. Можно еще улучшить производительность, разбив весь процесс на задачи и выполняя их на различных экземплярах AOS, тем самым увеличивая пропускную способность обработки и уменьшая общее время выполнения.
- **Реализация расширенной журнализации и профилирования.** Инфраструктура пакетных заданий дает возможность увидеть, какие ошибки и исключения были выброшены во время последнего запуска пакета, а также сколько времени потребовало его выполнение. Расширенное журналирование и новые возможности профилирования будут полезны при измерении производительности и аудите безопасности.

## Производительность

Новая возможность запуска больших и более сложных пакетных заданий потребовала улучшения производительности всей инфраструктуры пакетных заданий. В Microsoft Dynamics AX 2012 инфраструктура пакетных заданий разработана как серверный компонент. Это позволяет разрабатывать и управлять многопоточными серверными процессами. Настраивая число параллельных нитей выполнения и серверов, задавая число и порядок выполнения задач и устанавливая расписание обработки, вы можете достичь большой степени масштабируемости по вашему аппаратному обеспечению.

Как говорилось ранее, инфраструктура пакетного выполнения сейчас рассчитана на выполнение кода пакетных заданий X++, скомпилированных в код .NET CIL. Это значительно увеличивает быстродействие по

сравнению с Microsoft Dynamics AX 2009, выполнявшей интерпретируемый код. По сравнению с интерпретируемым кодом, была улучшена сборка мусора, к тому же благодаря сеансовому пулу создание новых пакетных заданий стало менее требовательным к ресурсам. Появилась возможность профилирования пакетных заданий с помощью Microsoft Visual Studio Performance Profiler.

Разработчики Microsoft используют пакетную инфраструктуру как фундамент для многих требовательных к производительности процессов, таких как максимизация масштабируемости по оборудованию во время обновления данных и увеличение пропускной способности разноски журналов. Больше информации о том, как Microsoft использует инфраструктуру пакетных заданий для критичных к производительности процессов, вы можете узнать из технического описания «Journal Batch Posting», доступного по адресу: *http://www.microsoft.com/en-us/download/details.aspx?id=13379*.

## Создание и выполнение пакетного задания

В Microsoft Dynamics AX 2012 уже включено довольно много пакетных заданий. Среди них – генерация отчетов, создание накладных по заказам на продажу, обработка журналов. Однако компаниями зачастую требуется создавать их собственные пакетные задания. Инфраструктура пакетных заданий обладает высокой гибкостью в создании различных типов заданий. В данном разделе мы пройдем по шагам, необходимым для создания, выполнения и управления пакетными заданиями.

1. Создание класса с поддержкой пакетного выполнения.
2. Создание пакетного задания и определение расписания его выполнения.
3. Настройка пакетного сервера и создание группы пакетов.
4. Управление пакетным заданием.

### Создание класса с поддержкой пакетного выполнения

Первым шагом в разработке пакетного задания является создание определения класса, который может выполняться в пакетном режиме. Многие классы, входящие в Microsoft Dynamics AX 2012, уже поддерживают пакетную обработку. Вы также можете создать класс с поддержкой пакетного выполнения, как показано в примере ниже.

```
public class ExampleBatchTask extends RunBaseBatch
```

Для запуска пакетной задачи, класс должен реализовывать интерфейс Batchable. Наилучшим путем реализовать контракт будет унаследовать абстрактный класс *RunBaseBatch*, содержащий большую часть нужной инфраструктуры для создания класса с поддержкой пакетного выполнения. Альтернативой этому является использование инфраструктуры SysOperation, обладающей рядом преимуществ по сравнению с расширением класса *RunBaseBatch*. Более подробно об инфраструктуре SysOperation рассказано в главе 14.

Класс *RunBaseBatch* – это расширение инфраструктуры RunBase, так что ваш пакетный класс должен следовать шаблонам и рекомендациям классов расширяющих *RunBase* (детальное описание см. в главе 14). Табл. 18-2 описывает методы, которые должны быть реализованы при расширении класса *RunBaseBatch*. В последующих разделах мы рассмотрим их подробнее.

**Табл. 18–2.** Методы, требуемые в наследниках класса *RunBaseBatch*

| Метод | Описание |
|---|---|
| *run* | Содержит основную логику вашей пакетной задачи |
| *pack* | Упаковывает список переменных, использованных в классе |
| *unpack* | Распаковывает список переменных, использованных в классе |
| *canGoBatch-Journal* | Определяет возможность появления класса в форме Пакетные задания |

### Метод *run*

Основная логика вашего класса реализуется в методе *run*. Этот метод вызывается инфраструктурой пакетных заданий для выполнения определенной в нем задачи. Вы можете выполнить в нем большую часть кода X++, с некоторыми ограничениями. Например, вы не можете вызывать в нем никакой клиентской логики или диалоговых окон. Но вы все так же можете использовать класс *Infolog*. Весь Infolog и сообщения исключений при выполнении класса будут захвачены и сохранены в таблице задач. Вы можете просмотреть их позже в формах Пакетные задания или Журнал пакетных заданий, находящихся в Администрирование системы > Запросы > Пакетные задания.

**Примечание.** Если в Infolog было выведено сообщение об ошибке, это не означает, что задание закончилось неудачно. Для сигнализирования о неудачном завершении следует выбрасывать исключение.

### Методы *pack* и *unpack*

Класс, унаследованный от *RunBaseBatch,* должен также реализовывать методы *pack* и *unpack* для того, чтобы класс мог быть сериализован. При создании пакетной задачи, переменные – члены класса упаковываются с помощью метода *pack* и сохраняются в таблицу *Batch*. Позже, когда сервер пакетных заданий выбирает задачу для выполнения, он распаковывает переменные – члены класса с использованием метода *unpack*. Так что крайне важно указать правильный список переменных, которые необходимы для выполнения класса. Если какая-то из переменных не может быть упакована, то и сам класс, ее содержащий, не может быть сериализован и десериализован в это же самое состояние.

Следующий пример иллюстрирует реализацию методов *pack* и *unpack*.

```
public container pack()
{
    return [#CurrentVersion,#CurrentList];
}

public boolean unpack(container _packedClass)
{
    Version version       = RunBase::getVersion(_packedClass);
    switch (version)
    {
        case #CurrentVersion:
            [version,#CurrentList] = _packedClass;
            break;
        default:
            return false;
    }
    return true;
}
```

Макросы *#CurrentList* и *#CurrentVersion*, используемые в приведенном коде, должны быть определены в объявлении класса. Использование макрокоманд упрощает управление переменными в классе. Если вы добави-

те или удалите переменные в процессе модификации класса, то сможете легко изменить этот список, изменив макрос. Макрос *#CurrentList* хранит список упаковываемых переменных – членов класса, как показано ниже.

```
#define.CurrentVersion(1)
#localmacro.CurrentList
        methodVariable1,
        methodVariable2
#endmacro
```

#### Метод *canGoBatchJournal*

Когда администратор системы создает новую пакетную задачу в форме Список задач, метод *canGoBatchJournal* определяет, появится ли данный класс пакетной задачи в списке доступных классов. Чтобы увидеть пример того, как использовать *canGoBatchJournal*, перейдем к следующему разделу.

## Создание пакетного задания

Вторым шагом в разработке пакетного задания станет создание пакетного задания и добавление в него задач. Пакетное задание можно создать тремя путями.

- Используя диалоговое окно класса, поддерживающего пакетное выполнение.
- Используя форму Пакетные задания.
- Используя прикладной интерфейс программирования пакетных заданий.

Используемый вами метод зависит от требующегося уровня гибкости и сложности пакетного задания. Для создания простого пакетного задания, состоящего из одной задачи без зависимостей, обычно достаточно использовать диалоговое окно класса, поддерживающего пакетное выполнение. Для создания более сложного пакетного задания, состоящего из нескольких задач, возможно с зависимостями между ними, используйте форму Пакетные задания. Для создания же весьма сложного или очень большого пакетного задания, или требующего интеграции с остальной бизнес-логикой, используйте интерфейс программирования пакетных заданий. В следующих разделах приводятся примеры использования каждого из этих методов.

### Создание пакетного задания из диалогового окна класса пакетной обработки

Простейший путь для запуска класса с поддержкой пакетной обработки в качестве пакетного задания – это вызов класса с помощью пункта меню. Пункт меню, указывающий на такой класс, автоматически открывает диалоговое окно, позволяющее пользователю создать пакетное задание. На вкладке Пакет диалогового окна установите отметку Пакетная обработка, как для класса *Изменение основных оповещений* (см. рис. 18-1). После отметки Пакетная обработка и щелчка на OK будет создано новое пакетное задание, представляющее данный класс. Затем оно будет асинхронно выполнено в установленную вами дату и время. Вы также можете настроить повторения или оповещения, щелкнув соответствующую кнопку справа в диалоговом окне. Кроме того, возможно указать заданию выполнение в группе пакетов, выбрав нужную в ниспадающем списке.

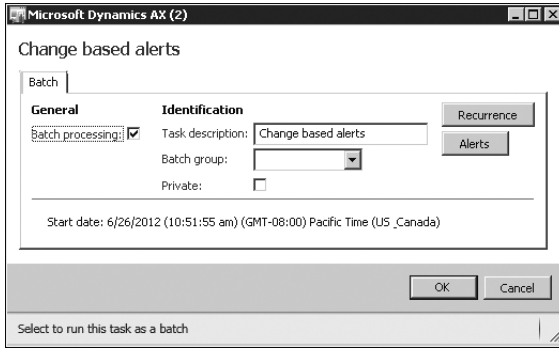


**Рис. 18–1.** Пример вкладки Пакет класса

### Создание пакетного задания с использованием формы Пакетные задания

Вы можете открыть форму Пакетные задания из нескольких мест. Например, вы можете открыть ее, выбрав Администрирование системы > Запросы > Пакетные задания или (для пользователей) выбрав Мои пакетные задания в Домашняя страница > Запросы > Мои пакетные задания. Оба пункта меню откроют одну и ту же форму, но информация, отображаемая в ней, будет различной, смотря из какого пункта меню вы ее открыли. В зависимости от этого и вашего уровня доступа, вам будут доступны либо только задания, созданные вами, либо все пакетные задания, существую-

щие в системе. Для создания нового задания нажмите Ctrl+N и заполните его поля в табличной части вкладки Обзор: описание, а также дату и время, когда вы хотите запустить задание. Кроме того, можно задать повторение пакетного задания, щелкнув Повторение в строке меню и введя диапазон повторения и его шаблон.

**Примечание.** Если вы не введете дату и время в соответствующие поля, то будут автоматически подставлены текущая дата и время.

На рис. 18-2 показана форма Пакетные задания.

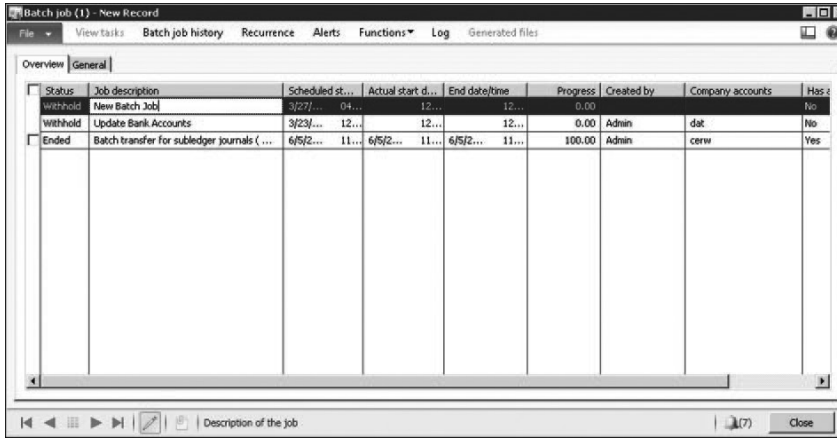


**Рис. 18–2.** Форма Пакетные задания

После того как вы создадите новое пакетное задание, вы можете добавлять в него задачи и создавать зависимости между ними, используя форму Список задач, показанную на рис. 18-3. Эта форма открывается при щелчке на Просмотр задач в строке меню формы Пакетные задания. Из формы Список задач вы также можете изменять статус пакетных задач или удалять ненужные более задачи.

Для создания задачи сделайте следующее.

1. Нажмите Ctrl+N для создания задачи.
2. В поле Описание задачи введите описание задачи.
3. В поле Компании выберите компанию, в которой будет работать задача.

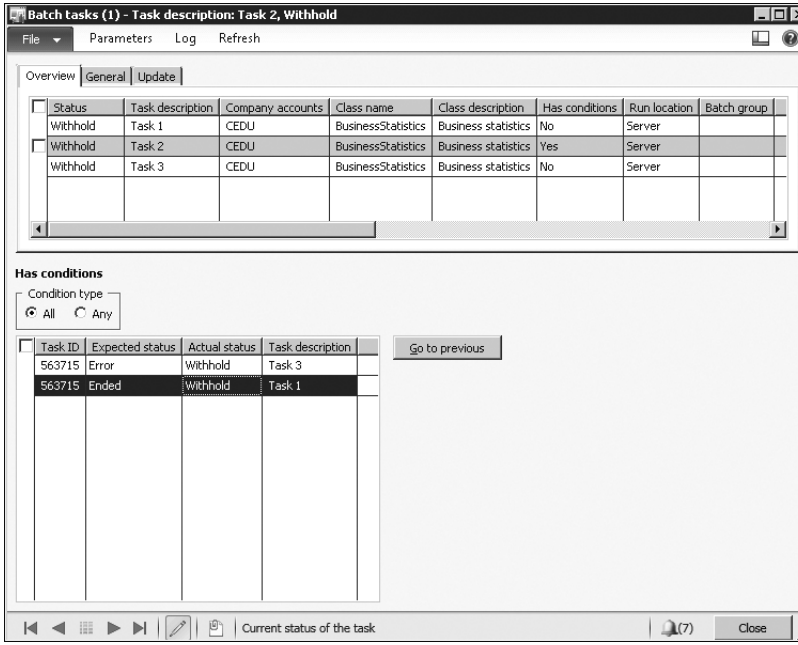


**Рис. 18–3.** Форма Список задач

4. В поле Имя класса выберите процесс, который хотите запустить в этой задаче. Классы появятся в виде ниспадающего списка, содержащего все доступные классы, поддерживающие пакетную обработку. Класс появится в этом списке, только если его свойство *CanGoBatchJournal* возвратит *True*.
5. В поле Группа пакетов выберите группу пакетов для задачи, если необходимо.
6. Сохраните задачу, нажав Ctrl+S.
7. Укажите, если нужно, параметры класса. Как говорилось в предыдущих разделах, каждая задача представляет собой класс с поддержкой пакетного выполнения. Иногда вам потребуется задать для такого класса его параметры. Например, вам может понадобиться указать параметры разноски для процедуры разноски накладных. Чтобы сделать это, нажмите Параметры в строке меню формы.

**Примечание.** Если вы создаете свой класс с пакетной обработкой, то должны создать собственную форму задания параметров. При разработке класса с использованием инфраструктуры SysOperation, этот процесс сильно упрощается. После задания вами параметров и нажатия OK, параметры класса будут упакованы и сохранены в таблицу *Batch*, а затем восстановлены при выполнении класса. Подробнее об инфраструктуре SysOperation можно прочесть в главе 14.

8. Если нужно, установите зависимости между задачами или настройте расширенные возможности последовательного их выполнения.

После создания пакетного задания и помещения в него задач, с помощью формы Список задач можно задать зависимости между задачами. Если зависимостей или условий в задании нет, пакетный сервер автоматически запустит все задачи параллельно. (Для настройки количества параллельно запускаемых задач используйте параметр Максимальное число потоков в пакетном задании в форме Конфигурация сервера.)

Если нужно реализовать сложную последовательность протекания какого-либо бизнес-процесса, вы можете использовать либо форму Список задач, либо интерфейс программирования пакетных заданий. Оба этих инструмента могут быть использованы для построения сложных древовидных зависимостей между пакетными задачами, позволяя назначить им параллельное выполнение, добавить многочисленные зависимости между ними, использовать разные пути их выполнения в зависимости от результатов выполнения предыдущих задач и т. д.

Предположим, что задание JOB1 имеет семь задач: TASK1, TASK2, TASK3, TASK4, TASK5, TASK6, TASK7, и вам нужно настроить для них следующую последовательность с зависимостями:

- TASK1 выполняется первой;
- TASK2 выполняется по завершению (Завершено либо Ошибка) TASK1 (т.е. вне зависимости от успешного или нет завершения TASK1);
- TASK3 выполняется при успешном (Завершено) завершении TASK2;
- TASK4 выполняется при успешном (Завершено) завершении TASK2;
- TASK5 выполняется при ошибочном (Ошибка) завершении TASK2;
- TASK6 выполняется при ошибочном (Ошибка) завершении TASK3;

- TASK7 выполняется при успешном (Завершено) завершении TASK3 и TASK4.

На рис. 18-4 показано дерево зависимостей для JOB1.

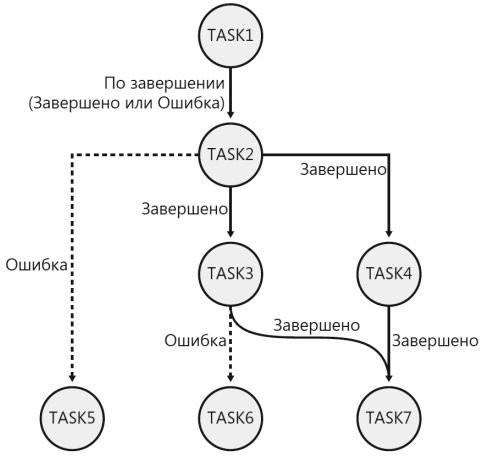


**Рис. 18–4**. Дерево зависимостей для JOB1

Для описания этих зависимостей между задачами и задания системе способа их обработки выберите подчиненную задачу – например, TASK2 – из приведенного списка и сделайте следующее.

1. В форме Список задач щелкните на табличной части в разделе Ограничения пакета и нажмите Ctrl+N для создания новой записи.
2. Выберите код родительской задачи – TASK1.
3. Выберите статус, который должна иметь родительская задача перед тем, как сможет запуститься подчиненная. Например, TASK2 будет запущена, когда статус TASK1 станет Завершение или ошибка.
4. Нажмите Ctrl+S для сохранения условия.
5. Если вы введете больше одного условия и они все должны быть выполнены перед тем, как подчиненная задача начнет свое выполнение, выберите Тип условия – Все.

   Или же, если подчиненная задача должна начать выполнение при выполнении хотя бы одного из указанных условий, установите Тип условия – Любое.

Форму Список задач можно также использовать для определения поведения системы при ошибочном завершении задач. Для игнорирования ошибок какой-либо задачи отметьте Игнорировать сбой выполнения задачи на вкладке Разное. В случае отметки этой опции, ошибка при выполнении задачи не будет означать ошибки выполнения задания в целом. Можно также установить максимальное число повторов попыток выполнения задачи перед признанием ее завершившейся ошибочно.

## Использование интерфейса программирования пакетных заданий

Для продвинутых сценариев, которые требуют сложных или больших пакетных заданий, таких как закрытие склада или обновление данных, инфраструктура пакетных заданий предоставляет интерфейс программирования, позволяющий нужным вам образом создавать или изменять пакетные задания, задачи и их зависимости, а также создавать динамические пакетные задачи. Этот гибкий API помогает автоматизировать создание задач и интегрировать пакетную обработку с другими бизнес-процессами. Он также может быть полезен в случае, если ваше задание или задача требуют дополнительной логики. Итак, для создания пакетного задания с помощью этого API нужно сделать следующие шаги.

1. Использовать класс *BatchHeader* для создания пакетного задания.
2. Изменить параметры пакетного задания.
3. Добавить в него задачи.
4. Определить зависимости между задачами.
5. Сохранить пакетное задание.

### Создание пакетного задания с помощью класса *BatchHeader*

Создайте экземпляр класса *BatchHeader,* представляющий ваше пакетное задание. Пример ниже создает экземпляр *BatchHeader,* названный *sampleBatchHeader.*

```
sampleBatchHeader = BatchHeader::construct();
```

Вы можете также создать объект *BatchHeader* для уже существующего задания, указав необязательный параметр *batchJobId* методу *construct*, как показано здесь:

```
//job1 это уже существующее задание
sampleBatchHeader = BatchHeader::construct(job1.parmCurrentBatch().BatchJobId);
```

### Изменение параметров пакетного задания

Класс *BatchHeader* позволяет вам получить доступ к изменению большей части своих параметров с использованием методов *parm*. Например, вы можете задать повторения и оповещения для вашего пакетного задания, как показано в примере ниже.

```
// Установка повторений пакета
sysRecurrenceData = SysRecurrence::defaultRecurrence();
sysRecurrenceData = SysRecurrence::setRecurrenceStartDateTime(sysRecurrenceData,
DateTimeUtil::utcNow());
sysRecurrenceData = SysRecurrence::setRecurrenceNoEnd(sysRecurrenceData);
sysRecurrenceData = SysRecurrence::setRecurrenceUnit(sysRecurrenceData,
SysRecurrenceUnit::Hour, 1);
sampleBatchHeader.parmRecurrenceData(sysRecurrenceData);

// Установка конфигураций уведомлений пакета
sampleBatchHeader.parmAlerts(NoYes::No, NoYes::Yes, NoYes::No, NoYes::No, NoYes::No);
```

### Добавление задачи в пакетное задание

Вызывая метод *addTask*, вы можете добавлять задачи в пакетное задание. Первый параметр данного метода – это экземпляр класса, поддерживающего пакетное выполнение, который должен быть обработан в рамках пакетного задания.

```
void addTask(Batchable batchTask,
    [BatchConstraintType constraintType])
```

Другой путь создания задачи – использование метода *addRuntimeTask*, создающего динамическую пакетную задачу. Такая задача существует только во время своего выполнения. Она копируется в исторические таблицы и удаляется в конце своего выполнения. Она наследует установки, такие как группа пакетов и зависимости подчиненных задач из задачи *inheritFromTaskId*.

```
void addRuntimeTask(Batchable batchTask,
RecId inheritFromTaskId,
[BatchConstraintType constraintType])
```

### Установка зависимостей между задачами

Класс *BatchHeader* имеет метод *addDependency*, который используется для задания зависимостей между задачами *batchTaskToRun* и *dependsOnBatchTask*.

Вы можете использовать параметр *batchStatus* для указания типа зависимости. По умолчанию создается зависимость типа *BatchDependencyStatus::Finished*. означающая, что задача начнет свое выполнение только при успешном завершении задачи, от которой она зависит. Другие возможные варианты – это *BatchDependencyStatus::Error* (задача начинает выполняться только при завершении предыдущей задачи с ошибкой) и *BatchDependencyStatus::FinishedOrError* (задача начинает выполнение при завершении предыдущей задачи с любым результирующим статусом). В примере ниже приводится сигнатура метода *addDependency*.

```
public BatchDependency addDependency(
    Batchable batchTaskToRun,
    Batchable dependsOnBatchTask,
    [BatchDependencyStatus batchStatus])
```

### Сохранение пакетного задания

Последним шагом в создании пакетного задания будет вызов метода *batchHeader.save* API для сохранения пакетного задания. Метод *save* вставляет записи в таблицы *BatchJob, Batch* и *BatchConstraints*, из которых пакетный сервер и будет их выбирать для исполнения.

### Пример пакетного задания

Приведенный ниже пример показывает, как с использованием API пакетных заданий создать пакетное задание и добавить в него две задачи. Пример приводится в предположении, что уже существует класс *ExampleBatchTask*, поддерживающий пакетное исполнение.

```
static void ExampleSchedulingJob (Args _args)
{
    BatchHeader         sampleBatchHeader;
    RunBaseBatch        sampleBatchTask;

    // создание заголовка пакетного задания
    sampleBatchHeader = BatchHeader::construct();

    // создание и добавление пакетных задач
    sampleBatchTask1 = new ExampleBatchTask();

    sampleBatchHeader.addTask(sampleBatchTask1);

    sampleBatchTask2 = new ExampleBatchTask();

    sampleBatchHeader.addTask(sampleBatchTask2);
```

```
        // добавление зависимостей между задачами
        sampleBatchHeader.addDependency(sampleBatchTask1, sampleBatchTask2);

        // сохранение пакетного задания в базу данных
        sampleBatchHeader.save();
    }
```

Еще больше примеров программного создания пакетных заданий можно найти в статье «Walkthrough: Extending RunBaseBatch Class to Create and Run a Batch» по адресу: *http://msdn.microsoft.com/en-us/library/cc636647.aspx*.

# Управление процессом выполнения пакетного задания

Заключительный шаг в реализации пакетного задания – это управление процессом его выполнения. Пред тем как задание будет выполнено на экземпляре сервера приложений, вы должны сконфигурировать его для выполнения роли сервера пакетных заданий и настроить группы пакетов, задающие системе место выполнения пакетных заданий. Наряду с этими начальными конфигурационными процедурами, вам понадобится управлять пакетными заданиями и задачами: проверять статус, просматривать историю и порой отменять пакетное задание. В некоторый момент времени может возникнуть необходимость отладки пакетной задачи. Последующие разделы описывают, как сконфигурировать экземпляр AOS в качестве пакетного сервера, настроить группы пакетов, управлять пакетными заданиями и производить отладку пакетных задач.

## Настройка пакетного сервера

Вы можете сконфигурировать экземпляр сервера приложения для обработки пакетных заданий, включая задание периода времени, во время которого сервер будет осуществлять обработку пакетных заданий и то, сколько пакетных заданий он может обрабатывать. Все это делается с помощью формы Конфигурация сервера. Она расположена в Администрирование системы > Настройка > Система > Конфигурация сервера. Заметьте, что первый экземпляр AOS автоматически устанавливается как пакетный сервер, но вы можете вручную настроить дополнительные экземпляры AOS в роли серверов обработки пакетных заданий.

**Совет.** Используйте несколько пакетных серверов для организации параллельной обработки и увеличения пропускной способности выполняемых обработок.

1. Выберите сервер в левой части формы Конфигурация сервера.
2. Для включения обработки пакетных заданий на этом сервере отметьте опцию Сервер обработки пакетных заданий, как показано на рис. 18-5.

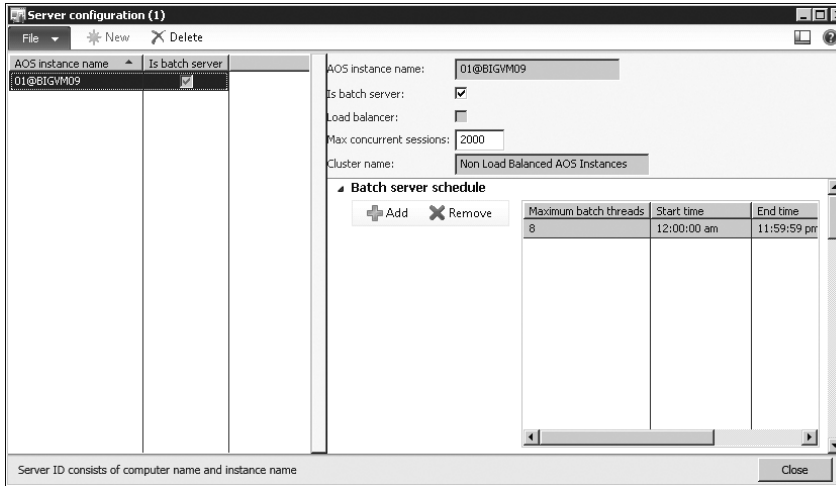


**Рис. 18–5.** Форма Конфигурация сервера

3. На экспресс-вкладке График сервера обработки пакетных заданий щелкните Добавить для создания нового расписания. Задайте максимальное количество одновременно выполняемых на сервере задач. Сервер будет выбирать из очереди задания, пока не достигнет этого максимума.
4. Для указания временного окна, в течение которого сервер будет выполнять пакетные задания, задайте время его начала в поле Время начала и время окончания в поле Время завершения. Нажав Ctrl+N, можно ввести несколько таких временных интервалов.

**Совет.** Хорошей идеей будет освободить сервер от пакетной обработки на время выполнения обычных транзакций. Можно настроить расписания серверов так, что экземпляры сервера при-

ложения будут обрабатывать обычные запросы пользователей днем, а пакетные задания – ночью. Помните, что если сервер будет занят обработкой пакетной задачи на момент окончания выделенного для обработки пакетов временного окна, он продолжит выполнение задачи вплоть до ее завершения. Но он не станет выбирать из очереди новые задачи.

## Создание группы пакетов

Группа пакетов – это способ логического разделения пакетных задач, позволяющий задать пользователю (обычно администратору системы), на каком экземпляре AOS будет выполняться задача. В этом разделе описывается, как создать группу пакетов для дальнейшего назначения выполнения ее задач конкретному серверу. Первым шагом будет создание группы пакетов с использованием формы Группы пакетов, расположенной в Администрирование системы > Настройка > Группа пакетов.

Для создания пакетной группы нажмите Ctrl+N в форме Группа пакетов и введите ее имя и описание. Форма Группа пакетов показана на рис. 18-6.


**Рис. 18–6.** Форма Группы пакетов

**Примечание.** По умолчанию система содержит пустую пакетную группу, которую невозможно удалить. Это группа используется по умолчанию для задач, явно не назначенных какой-либо группе.

После создания нужных групп пакетов нужно назначить каждую группу какому-либо серверу.

1. В форме Конфигурация сервера (см. рис. 18-7) щелкните на экспресс-вкладке Группы серверов пакетной обработки. Список Выбранные группы содержит группы, назначенные для обработки на выбранном сервере.
2. В списке Оставшиеся группы выберите нужную группу и щелкните на кнопке со стрелкой влево, чтобы задать ее обработку на выбранном сервере.

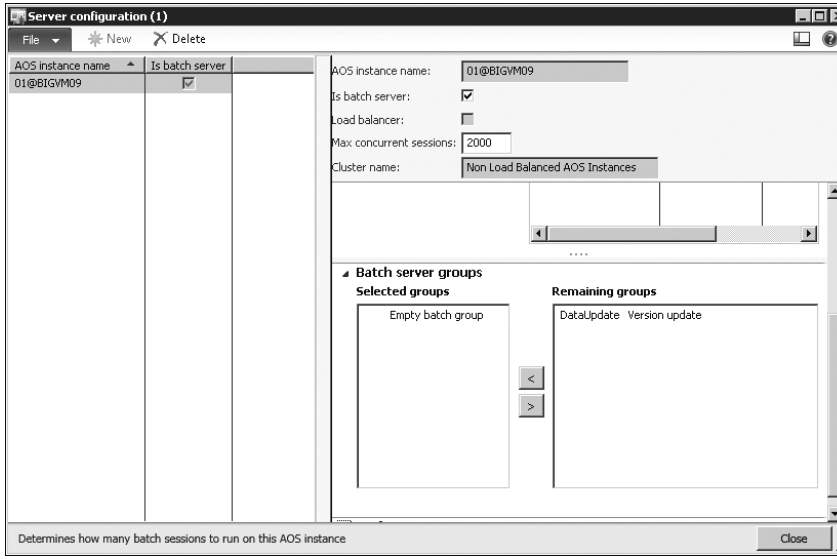


**Рис. 18–7.** Форма Конфигурация сервера

## Управление пакетными заданиями

После создания и указания расписания пакетного задания вам, возможно, потребуется проверить его статус, посмотреть журнал его обработки или отменить это задание. В следующих разделах описываются основные задачи управления пакетными заданиями.

### Просмотр и изменение статуса пакетного задания

Форма списка Пакетные задания показывает мгновенный снимок текущего состояния пакетных заданий. Этот список показывает прогресс выполнения и статус выполняющихся и завершенных задач. Он также показывает задания, назначенные для выполнения в ближайшее время.

Вы можете изменить статус выбранного задания, проделав следующие шаги.

1. Щелкните Функции, затем Изменить статус.
2. В окне Выбрать новый статус выберите нужный новый статус задания. Например, если в настоящий момент его статус – Ожидание, то можно временно исключить задание из списка ожидающих обработки, изменив статус на Отложено.

**Совет.** Если задание завершилось со статусом Ошибка или Завершено и вы хотите повторно запустить задание, измените его статус на Ожидание. Тогда задание будет автоматически выбрано сервером для выполнения.

Вы можете отменить задание, изменив его статус на Отмена. Его задачи в состоянии Готово или Ожидание сменят статус на Не выполнялось, выполняемые в текущий момент задачи будут прерваны и их статус будет изменен на Отменено.

### Задание максимального числа повторов

Если AOS во время выполнения пакетного задания выйдет из строя вследствие отказа оборудования или сбоя электропитания, то на этот случай инфраструктура пакетных заданий имеет способность перезапуска выполнения после перезагрузки AOS. Все задачи, оставшиеся в состоянии Выполнение и не достигшие максимального количества попыток перезапуска, сменят состояние на Готово и будут запущены вскоре после сбоя.

**Совет.** Если вы создали собственные задачи и хотите включить возможность перезапуска их выполнения, постройте их так, чтобы они были идемпотентны – то есть могли быть выполнены несколько раз без неожиданных побочных эффектов.

Вы можете изменить атрибут Максимальное число повторов у каждой задачи на вкладке Разное. По умолчанию там установлено значение 1.

Когда поле Фактические попытки на вкладке Обновить превысит максимальное число повторов, пакетная задача считается завершившейся с ошибкой. В таком случае повторение, установленное для задачи, игнорируется, и статус пакетного задания переходит в Завершено либо Ошибка.

#### Просмотр истории пакетного задания

Просмотр истории выполнения всех завершившихся пакетных заданий доступен в форме Журнал пакетных заданий, расположенной в Администрирование системы > Запросы > Журнал пакетных заданий. Эта форма показывает детальную информацию о статусах заданий, включая любые сообщения, выведенные в процессе их выполнения.

Журнал по каждому заданию доступен в следующих местах.

1. Для просмотра журнала по всему заданию выберите задание и нажмите Журнал.
2. Для просмотра журнала отдельной задачи выберите задание и щелкните Просмотр задач. В форме списка Журнал пакетных заданий выберите задачу и затем нажмите Журнал.

**Совет.** В настройках задания вы можете указать условия, при которых информация журнала задания будет записана в таблицы истории: Всегда (по умолчанию), Ошибки, Никогда. Используйте ошибки или Никогда для сокращения потребления дискового пространства для регулярно выполняющихся задач. Эта настройка расположена на вкладке Разное формы Пакетные задания.

## Отладка пакетных задач

Поскольку пакетные задания выполняются в неинтерактивном окружении и код X++ работает в среде выполнения CLR, для отладки вам придется сделать несколько дополнительных шагов для настройки AOS и отладчика Visual Studio, помимо, собственно, установки точек останова.

Сначала нужно настроить AOS для отладки. Это необходимо сделать по двум причинам. Первая из них – то, что AOS модифицирует сборку X++ для отключения оптимизаций Just-In-Time (JIT) в CLR. Это позволяет увидеть и проанализировать переменные и содержимое объектов в отладчике. Вторая – то, что AOS создает файлы с исходным кодом X++ в папке сервера Bin\XppIL\Source. Затем вы можете открыть эти файлы в Visual

Studio для установки в них точек останова и осуществления обычных при отладке действий, вроде шага с заходом и шага с обходом.

## Настройка AOS для отладки пакетного режима

Используйте утилиту Microsoft Dynamics AX Server Configuration для настройки AOS для отладки пакетных заданий. Она доступна на компьютере, на котором установлен сервер приложений. Для настройки выполните следующие действия.

1. Откройте утилиту Microsoft Dynamics AX Server Configuration. Щелкните Пуск > Все программы > Администрирование > Microsoft Dynamics AX 2012 Server Configuration.
2. Установите отметку Enable Breakpoints to Debug X++ Code Running on This Server.
3. Щелкните OK, чтобы закрыть утилиту, и перезапустите AOS.

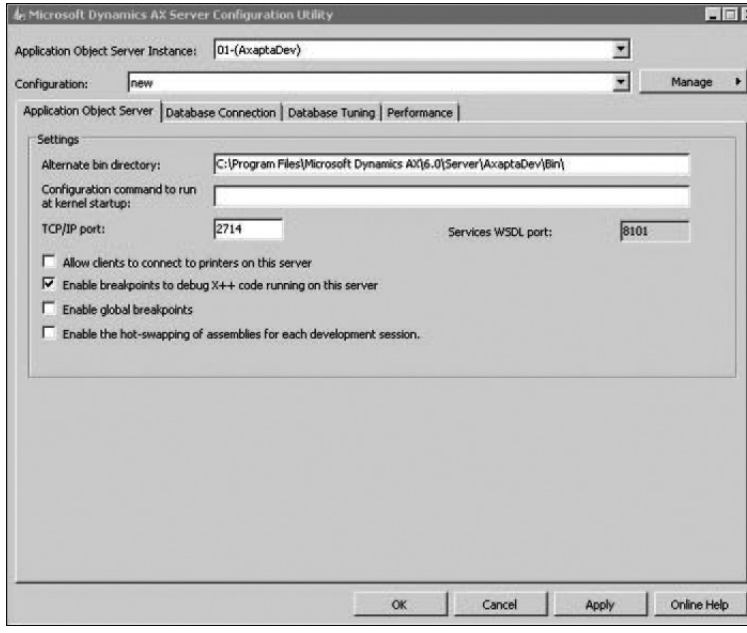


**Рис. 18–8.** Утилита Microsoft Dynamics AX 2012 Server Configuration

## Настройка Visual Studio для отладки X++ в пакетном режиме

Для настройки Visual Studio для отладки пакетного режима подключитесь к процессу AOS Ax32Serv.exe, выполнив следующие шаги.

1. В Visual Studio, в меню Debug щелкните Attach To Process.
2. В открывшемся диалоговом окне Attach To Process (см. рис. 18-9) щелкните Select и выберите Managed (v4.0) code, затем установите следующие отметки:
   - Show Processes From All Users;
   - Show Processes In All Sessions.
3. В списке процессов выберите Ax32Serv.exe и затем нажмите Attach.

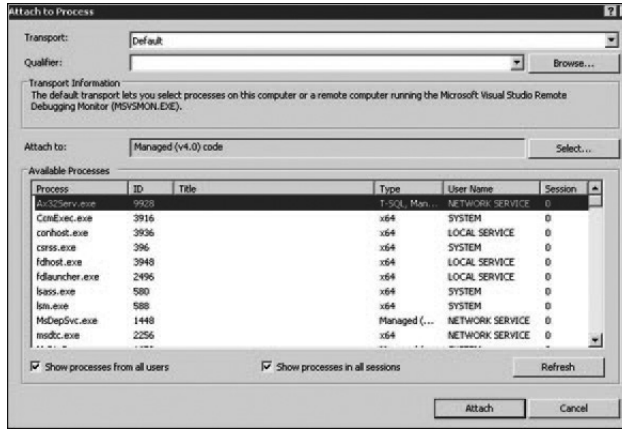


**Рис. 18–9.** Диалоговое окно Attach To Process в Visual Studio

Следующим шагом будет отключение опции Just My Code для того, чтобы отладчик смог остановиться в исходном коде X++. Для этого сделайте еще пару шагов.

1. В меню Tools щелкните Options и перейдите к узлу Debugging\General (показанному на рис. 18-10).
2. Снимите отметку Enable Just My Code (Managed Only) и нажмите OK.

   Пройдя все описанные выше шаги, вы сможете открыть исходный код X++ из папки Bin\XppIL\Source сервера и установить в нем точки останова.

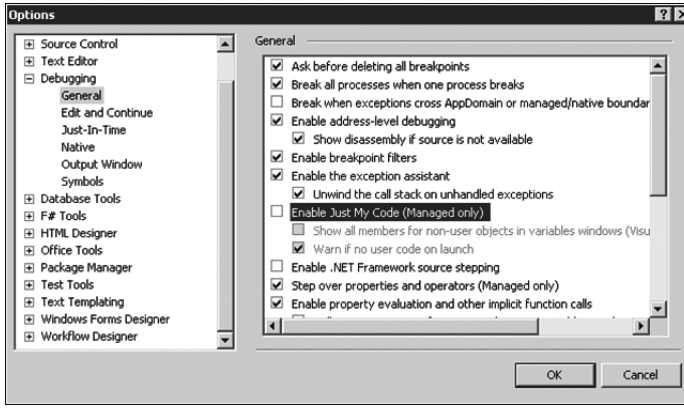


**Рис. 18–10.** Диалоговое окно Options в Visual Studio