ULTIMATE SOLID

2017 R1

Изменения, нарушающие обратную совместимость

ver. 6.0.0

www.ultimatesolid.ru

Оглавление

Изменения, нарушающие обратную совместимость

Релизы платформы могут содержать изменения, нарушающие обратную совместимость. Все такие изменения перечислены в данном разделе. Для удобства они сгруппированы по релизам.

Релиз платформы 4.15.243 (6)

1. В файле конфигурации сервера печати появилась новая настройка – *RetryCount*, которая задает количество попыток печати задания в случае возникновения ошибки. Это может стать причиной проблем при загрузке из кэша заданий и шаблонов печати. Чтобы их избежать, необходимо удалить содержимое соответствующих папок на всех серверах печати. Пути к данным папкам с кэшем задаются параметрами *TasksRepositoriesPath* и *TemplatesRepositoryPath* в том же файле конфигурации сервера печати.
2. Поменялись сигнатуры методов класса *ToolbarPanel*. Было:

```
public void EditSelectedRecord(object item = null);

protected virtual void InternalInsertRecord(InsertRecordEventArgs args);
protected virtual void InternalEditRecord(object selectedItem);
protected virtual void InternalDoubleClickRecord(object selectedItem);

protected bool OnInsertRecord(InsertRecordEventArgs args);
protected bool OnDeleteRecords(IEnumerable<object> items);
protected bool OnEditRecord(object item);
protected bool OnDoubleClickRecord(object item);
```

   Стало:

```
public Task EditSelectedRecord(object item = null);

protected virtual Task InternalInsertRecord(InsertRecordEventArgs args);
protected virtual Task InternalEditRecord(object selectedItem);
protected virtual Task InternalDoubleClickRecord(object selectedItem);

protected Task<bool> OnInsertRecord(InsertRecordEventArgs args);
protected Task<bool> OnDeleteRecords(IEnumerable<object> items);
protected Task<bool> OnEditRecord(object item);
protected Task<bool> OnDoubleClickRecord(object item);
```

   Исправление существующего кода для восстановления совместимости в большинстве случаев сводится к замене *async void* на *async Task*.

3. Изменен метод *BaseEditForm.ValidateRecord()*. Было:

```
protected virtual void ValidateRecord(ValidationErrorCollection errors)
```

   Стало:

```
protected virtual Task ValidateRecord(ValidationErrorCollection errors)
```

   Исправление существующего кода для восстановления совместимости в большинстве случаев сводится к замене *async void* на *async Task*.

Релиз платформы 4.16.22 (7)

1. Появилась поддержка асинхронности в *HandledEventArgs*, *InsertRecordEventArgs* и прочих, сами классы перенесены из *UltimaInterfaces* в *ClientInterfaces*, пространство имен сменилось с *Ultima* на *Ultima.Client*. В клиентском коде необходимо выполнить поиск и замену (в основном в файлах *.Designer.cs). Было:

```
Ultima.HandledEventArgs
Ultima.InsertRecordEventArgs
Ultima.ProcessRecordsEventArgs
Ultima.ProgressEventArgs
Ultima.FileUpdateProgressEventArgs
```

   Стало:

```
Ultima.Client.HandledEventArgs
Ultima.Client.InsertRecordEventArgs
Ultima.Client.ProcessRecordsEventArgs
Ultima.Client.ProgressEventArgs
Ultima.Client.FileUpdateProgressEventArgs
```

2. В *CommonControl* и *CommonForm* добавились виртуальные методы *OnLoadAsync*. Когда нужны асинхронные вызовы, следует переопределять этот метод, а не *OnLoad*. Отличия от стандатного метода *OnLoad*:
   - асинхронный, возвращает *Task*;
   - не вызывается в design-режиме, соответственно, не требуются проверки флага *DesignMode*.

   Для исправления существующего кода следует найти все вхождения *async void OnLoad*:

```
protected override async void OnLoad(EventArgs e)
{
	base.OnLoad(e);
	if (!DesignMode)
	{
		await ...;
	}
}
```

   Заменить вызовы на *async Task OnLoadAsync* и убрать проверку флага *DesignMode*:

```
protected override async Task OnLoadAsync(EventArgs e)
{
	await base.OnLoadAsync(e); // не забыть также заменить и вызов base-метода
	await ...;
}
```

Релиз платформы 4.18.55 (8)

1. Удалены методы *WhereIf* классов *IQueryable* и *IEnumerable*. Для исправление существующего кода необходимо найти все вхождения метода *WhereIf* в скриптах и переписать их с помощью обычного *Where*.

2. Реализовано центральное журналирование на базе библиотеки Serilog. Теперь это основной способ журналирования вместо NLog. Если не внести в файлы конфигурации каждого приложения соответствующие правки, журналирование на NLog перестанет работать. Чтобы этого избежать, следует добавить в файлы конфигураций следующий блок:

```
<appSettings>
  <add key="serilog:minimum-level" value="Verbose" />
  <add key="serilog:enrich:with-property:ProjectName" value="YourProjectName" />
  <add key="serilog:using:NLog" value="Serilog.Sinks.NLog" />
  <add key="serilog:write-to:NLog.restrictedToMinimumLevel" value="Verbose" />
</appSettings>
```

Приведенное изменение конфигурации всего лишь восстановит журналирование на NLog. Чтобы получить преимущества централизованного структурированного хранения логов с возможностью их чтения посредством встроенного в клиент инструмента просмотра, следует изучить раздел "Аппаратный комплекс / Журналирование" документации администратора системы.

Релиз платформы 4.20.56 (9)

1. Расширенная информация об ошибках web сервиса, которая передается клиенту в HTTP заголовке *UltimaErrorText*, теперь экранируется. Это улучшение реализовано для возможности передавать любой текст. По этой причине на клиенте необходимо выполнить обратную операцию, если, конечно, сообщение об ошибке необходимо.
2. Метод BaseParamForm.RetrieveParameters теперь асинхронный.

   2.1. После обновления перестанут компилироваться проекты конфигураций.
   Необходимо будет добавить await к вызову указанной функции. Пример:

```
var parameters = await BaseParamForm.RetrieveParameters(command);
```

   2.2. В форме параметров теперь можно переопределить метод

```
protected virtual Task<bool> ShowParametersForm()
```

По умолчанию он показывает форму и возвращает **true**, если нажали кнопку OK. Таким образом, в зависимости от логики можно принять решение, стоит и показывать форму параметров или сразу переходить к выполнению команды.

Релиз платформы 5.2.0.0 (11)

В этом релизе исправлена ошибка, из-за которой предикаты применялись в unrestricted-режиме. Все команды, для которых есть права доступа, выполняются в unresticted mode, от имени администратора. В этом режиме все проверки прав возвращают true, а предикаты текущего пользователя не применяются.

Чтобы предикаты пользователя применялись, нужно использовать AuthManager.SetRestrictedMode(). Вот как будет выглядеть код, если у текущего пользователя включен предикат на товаре 500430:

```
public void Execute(IDictionary<string, object> parameters, IList<ClientAction>
clientActions)
{
    // shouldn't throw exceptions
    DictionaryManager.GetRecord<Article>(500430);

    // set restricted mode to enable user's predicates
    using (AuthManager.SetRestrictedMode())
    {
        try
        {
            // should throw exceptions
            DictionaryManager.GetRecord<Article>(500430);
            throw new UltimaException(@"Predicate wasn't applied!");
        }
        catch (InvalidOperationException)
        {
            // record not found is thrown as expected
        }
    }

    throw new UltimaWarningException(@"Everything is fine!");
}
```

Релиз платформы 5.3.0.0 (12)

В этом релизе полностью обновлен весь технологический стек, включая .NET Framework 4.6 и DevExpress 15.1. Все нарушения обратной совместимости касаются обновления DevExpress, поэтому при обновлении требуются обширные правки в клиентских модулях. Большинство нарушений исправляются тривиально: при компиляции выдается сообщение вида «метод X устарел, используйте метод Y». В этом случае надо просто сделать, что требует компилятор. Если жалобы касаются файлов Designer.cs, то чаще всего достаточно открыть компонент в дизайнере форм, чтобы все само поправилось и перегенерировалось, как надо.

В отдельных случаях, однако, этого недостаточно.

1. В элементах управления для картографии все ссылки на GeoPoint заменены ссылками на CoordPoint. Из-за этого потребовалось:
   a. Добавить свойство UltimaMapControl.InnerMap, возвращающий MapControl
   b. Добавить приведения типа CoordPoint к GeoPoint (CoordPoint — абстрактный тип, реальный тип данных везде по-прежнему GeoPoint).

2. Там же — VectorItemsLayer больше не содержит свойство Items. Вместо него DevExpress предлагает использовать MapItemStorage. Везде в коде, где требуется работа с векторными слоями, придется добавить обращение к (MapItemStorage)VectorItemsLayer.Data.

3. В событиях GridView.CustomDisplayText параметр CustomColumnDisplayTextEventArgs больше не содержит свойства RowHandle. Получить RowHandle можно так: e.Column.View.GetRowHandle(e.ListSourceRowIndex).

Все изменения, которые требуется сделать, рекомендуется проверять по базе знаний на сайте DevExpress. На все вопросы, которые возникают при обновлении базового торгового модуля, ответы там уже имеются.

Релиз платформы 5.4.0.0 (13)

1. Метаданные — может потребоваться перегенерация классов метаданных.

2. Серверная часть:

2.1. Добавились методы BeforeClone в общие обработчики справочников и документов. При апгрейде они будут обновлены. Эти изменения потребуется затянуть на все рабочие ветки метаданных. Пока изменения не затянуты, общие обработчики будут выдавать ошибку компиляции.

2.2. Изменился формат списка безлимитных пользователей в файле лицензии. Теперь пароли таких пользователей проверяются по файлу лицензии, а не по базе данных. Если в проекте используются безлимитные пользователи, вам потребуется перевыпуск лицензии!

2.3. Старые синхронные методы IEmailService.SendMail и ISmsService.Send помечены как устаревшие. Нужно заменить их новыми методами для отсылки уведомлений с помощью очередей:

```
// отсылка письма по электронной почте
IEmailService.SendMail(EmailMessage message, EmailOptions options = null);

// отсылка SMS-сообщения
ISmsService.SendMessage(string phone, string message, SmsOptions options = null);
```

3. Веб-сервисы — деструктивных изменений нет.

4. Клиентская часть:

4.1. Переделано клонирование записей. Необходимо проверить функционал во всех справочниках, где используется клонирование (товары, контрагенты и тому подобные).

4.2. Кастомные журналы документов, в конструкторе которых не задается DocumentType, не смогут корректно загрузить команды. Для исправления во всех кастомных журналах нужно задать DocumentType.

Релиз платформы 5.5.0.0 (14)

1. Метаданные — может потребоваться перегенерация классов метаданных.

2. Серверная часть

2.1. Изменился формат файла лицензии, потребуется перевыпуск.

2.2. Для задач рассылки СМС и электронной почты появились серверные настройки: количество потоков рассылки и размеры пакета. По умолчанию количество потоков 0, для сервера приложений, запускающего задачи, нужно выставить как минимум в 1. Убедитесь также, что указаны ненулевые размеры пакетов. По умолчанию для СМС размер пакета 1, для электронной почты — 100.

2.3. Удаление колонок в SlimTable запрещено (раньше оно просто не работало). Вместо удаления колонок теперь можно только создать новую таблицу. Если в скриптах используется SlimTable.Columns.Remove/RemoveAt — придется править.

2.4. На стороне сервера теперь должны лежать некоторые клиентские сборки для компиляции клиентских скриптов. При обновлении сервера достаточно будет скопировать все файлы из дистрибутива, ничего нигде не меняя.

2.5. Для компиляции клиентских скриптов требуется, чтобы сервер мог грузить клиентские сборки. Пути к этим сборкам нужно добавить в конфиг-файл сервера. Убедитесь, что в разделе runtime -> assemblyBinding узел probingPath содержит все нужные пути:

```
<probing
privatePath="ThirdParty;ThirdParty/DevExpress;Client;Client/ClientModules;Client/ClientModules/Base"/>
```

3. Веб-сервисы — деструктивных изменений нет.

4. Клиентская часть

4.1. Устаревший StyleCop заменен анализатором StyleCop.Analyzers для Visual Studio 2015. Чтобы перейти на использование нового анализатора, нужно удалить из проектов все упоминания старого StyleCop (это делается руками, в обычном текстовом редакторе). Затем надо добавить в проекты ссылки на сборки нового анализатора, которые лежат в папке ThirdParty\StyleCop.Analyzers (это делается в Студии: Add reference...).

4.2. Метод EditableGridPanel.PopulateGridColumns теперь асинхронный. Если он переопределен в ваших клиентских модулях (например, в табличных частях), может потребоваться правка.

4.3. Extension-метод string.RemoveHeadingSpaces перенесен в класс StringExtensions. Если к нему есть прямые обращения, может потребоваться правка, если нет — нужна только перекомпиляция клиентских модулей.

Релиз платформы 5.6.0.0 (15)

1. Метаданные

1.1. Требуется перегенерация классов метаданных.

1.2. Для ссылок на документы теперь генерируются свойства-ссылки не общего типа Document, а конкретных типов документов.

2. Серверная часть.

2.1. Константы класса User перенесены в класс User.Constants. В скриптах все обращения к этим константам нужно поправить: User.WebSite -> User.Constants.WebSite.

2.2. В драйверах итогов коллекция DetailedTransactions более недоступна. Для поиска рассчитанных проводок теперь есть метод FindDetailedTransactions. Пример использования:

```
var result = FindDetailedTransactions<SaleDetailedTransaction>(s => s.IncomeDocumentID == id);
var first = result.FirstOrDefault();
```

2.3. Задачи теперь запускаются на языке, указанном в пользователе TaskExecutor. Раньше запускались на дефолтовом языке операционной системы. Проверьте, что у пользователя TaskExecutor (код 15) в профиле указан русский язык.

2.4. Обновилась версия Zyan. Нужно обновить его для сборки клиента, чтобы версия на сервере совпадала с версией на клиенте.

3. Веб-сервисы — деструктивных изменений нет.

4. Клиентская часть — старые сохраненные настройки шаблонных форм могут быть потеряны после обновления.

Релиз платформы 5.7.0.0 (16)

1. Метаданные

1.1. Требуется перегенерация классов метаданных.

2. Серверная часть.

2.1. Драйвер CurrencyExchange теперь использует переменную CurrencyAmount, как все другие драйверы денежных итогов. Если в вашем итоге обмена валюты стоит Quantity, придется переименовать переменную (колонку в базе данных при этом переименовывать не нужно).

2.2. Драйверы итогов теперь не пропускают проводки с дробными копейками, при проведении документа будет сообщение об ошибке. Если в бизнес-процессах проекта реально используются дробные копейки (что крайне маловероятно), эту проверку можно отключить, переопределив в скриптах драйверов соответствующих итогов метод ValidateAmount:

```
protected virtual void ValidateAmount(TransactionValue transaction)
{
   // nothing here
}
```

2.3. В интерфейсе ILogger методы Trace, TraceException, InfoException, DebugException, WarnException, ErrorException, FatalException помечены как устаревшие. Чтобы поправить, выбросите из названия метода слово Exception, а вместо Trace используйте Verbose.

3. Веб-сервисы — деструктивных изменений нет.

4. Клиентская часть

4.1. В табличных частях и связующих таблицах ссылки на справочники и документы теперь отображаются с помощью элементов DictionaryLookupEdit, DictionaryLookupTreeEdit и DocumentEllipseEdit. В некоторых табличных частях ссылки на справочники отображаются в двух отдельных колонках: код записи и текст записи. После перехода на новые элементы управления окажется, что коды записей в списке дублируются. Чтобы этого избежать, придется поправить соответствующие контролы табличных частей.

Для большинства документов и табличных частей делать ничего не нужно. Дальнейшая инструкция касается только тех документов, в которых кастомные табличные части показывают коды товаров (офисов, ЦФО и т. п.) в отдельной колонке. В базовом решении это табличные части в формах PaymentRequestEditForm, SaleEditForm и InterstoreTransferEditForm.

В кастомных контролах табличных частей переопределить метод **CreateColumnEditor**:

```
protected override async Task<RepositoryItem> CreateColumnEditor(IScalarPropertyDescriptor
prop)
{
    var editor = await base.CreateColumnEditor(prop);

    var edit = editor as IRepositoryItemRecordEdit;
    if (edit != null)
    {
        edit.IDEditVisible = false;        // (1)
        edit.DisplayTextIDVisible = false; // (2)
    }

    return editor;
}
```

Рекомендация следующая:

- используйте вариант (1) для тех документов, где колонка ID редактируемая;

- используйте вариант (2) для тех документов, где колонка ID не редактируемая.

Релиз платформы 6.0.0.0 (17)

1. Метаданные

1.1. Требуется перегенерация классов метаданных.

2. Серверная часть.

2.1. Настройка конфиг-файла сервера **UseManagedDataProvider** заменена ключом **DataProviderName**. Вместо True — *OracleManaged,* вместо False — *OracleUnmanaged*.

2.2. Добавлена настройка **DataProviderName**. Может принимать значения: *OracleManaged, OracleUnmanaged, Postgres.*

2.3. Метод ISqlService.GetSysdate переименован в GetDatabaseTime, чтобы не использовать название функции, специфичное для Oracle.

2.4. Конечный таймаут транзакций, указанный в machine.config теперь обрабатывается как ошибка, а не предупреждение. Чтобы отключить таймаут транзакций, поправьте файл machine.config на сервере следующим образом:

```
<configuration>
	<system.transactions>
		<machineSettings maxTimeout="00:00:00" />
	</system.transactions>
</configuration>
```

2.5. Типы всех колонок DATE изменились на TIMESTAMP(0) WITH LOCAL TIME ZONE. Это может привести к проблемам в некоторых SQL запросах (в базовом решении таких проблем не найдено).

2.6. При конвертации дат таблицы журналов изменений, сессий и статистики пришлось пересоздать. Старые таблицы были переименованы, поэтому для поиска по истории придется какое-то время привлекать программистов (пока таблицы с историей не будут заполнены новыми данными). Приносим извинения за доставленные неудобства. Вот список переименованных таблиц:

```
LOG_CHANGES -> DT_LOG_CHANGES
LOG_FIELDS_CHANGED -> DT_LOG_FIELDS_CHANGED
PRINT_STATS -> DT_PRINT_STATS
SESSIONS -> DT_SESSIONS
STAT_COMMAND_EVENTS -> DT_STAT_COMMAND_EVENTS
TASK_STATS -> DT_TASK_STATS
WEB_SERVICE_SESSIONS -> DT_WEB_SERVICE_SESSIONS
```

3. Веб-сервисы — деструктивных изменений нет.

4. Клиентская часть

4.1. Методы AfterCreate в клиентских скриптах поменяли сигнатуры. Вместо Dictionary<string, object> теперь они принимают в качестве параметра IDictionary<string, object>, аналогично методам BeforeCreate:

```
// было
protected override Task AfterCreate(Dictionary<string, object> parameters)
```

```
// стало
protected override Task AfterCreate(IDictionary<string, object> parameters)
```

4.2. Метод TextPrintHelper.PrintText стал асинхронным и поменял свое название на TextPrintHelper.PrintTextAsync.

4.3. Метод PrintTextForm.Print переименован в PrintTextForm.PrintAsync.

4.4. DevExpress обновлен до версии 16.2.4. Многие изменения ломают обратную совместимость. Вот список изменений, которые потребовались в базовых модулях UTRADE:

4.4.1. Класс ImageTilesLayer заменить на ImageLayer (UltimaMapControl.cs, GeocodingTestParamForm.Designer.cs)

4.4.2. Удалить использование "using DevExpress.Map.BingServices". BingServices теперь является частью XtraMap (DeliveryDriverEditForm.cs).

4.4.3. Класс Resource теперь является интерфейсом. Resource.Empty больше нельзя использовать, вместо него теперь ResourceEmpty (DeliveryManagerResponsibilitiesForm.cs):

```
// было
cell.Resource == Resource.Empty

// стало
ResourceEmpty.Resource.Equals(cell.Resource)
```

4.4.4. Все сравнения с EmptyResource теперь следует оформлять следующим образом (SupplierDispatchAppointmentEditForm.cs):

```
// было
apt.ResourceId is EmptyResource

// стало
apt.ResourceId == ResourceEmpty.Id
```

4.4.5. Класс Appointment теперь является интерфейсом. Если нужно унаследоваться от бывшего класса Appointment и расширить его свойства, следует использовать AppointmentInstance (SupplierDispatchAppointment.cs).

4.4.6. Класс Resource теперь является интерфейсом. Вместо него теперь используется ResourceInstance (SupplierDispatchResource.cs).

4.4.7. Вместо свойства Appointment.LabelId нужно использовать Appointment.LabelKey (WarrantySupplierDispatchScheduleForm.cs).

4.4.8. Из файлов, сгенерированных дизайнером форм, нужно удалить строки следующего вида (WarrantySupplierDispatchScheduleForm.Designer.cs):

```
// удалить похожие строки из файлов Designer.cs
this.SchedulerStorage.Appointments.Labels.Add(new
DevExpress.XtraScheduler.AppointmentLabel(...));
```

4.5. Предупреждения о синхронных вызовах теперь нельзя отключить для разработчиков (пользователей с правом Developer). Предупреждения теперь показываются как всплывающие уведомления, а не ошибки.

5. Утилита апгрейда базы данных

5.1. Изменился формат файла конфигурации. Строчку соединения с БД (connection string) теперь нужно указать один раз. Учетные записи переехали в отдельный раздел конфиг-файла, как показано ниже:

```
<setting name="DatabaseUsers" serializeAs="Xml">
```

```
    <value>
        <!-- Name="user" Value="password" or Value="login:password", i.e.
"globalultima:pa55w0rd" -->
        <SerializableStringDictionary>
            <Pair Name="kernel" Value="kernelpass" />
            <Pair Name="sys" Value="sysutrade:syspass" />
            <Pair Name="ultima" Value="ultimapass" />
        </SerializableStringDictionary>
    </value>
</setting>
```

5.2. Утилите апгрейда теперь требуется еще и учетная запись **ultima**. Если имя учетной записи отличается от **kernel, ultima** и **sys**, его можно указать, как показано ниже:

```
<Pair Name="ultima" Value="globalultima:pa55w0rd" />
```

5.3. Перед запуском утилиты апгрейда все текущие изменения на всех рабочих ветках должны быть протолкнуты на главную ветку (обычно Default или Production). В противном случае не все объекты прикладной схемы будут преобразованы, что приведет к проблемам в будущем!