

Содержание

Страница создания нового триггера	3
Немного о создании триггеров в целом	4
Вызов API в триггерах	5
Локальные функции	5
Встроенные переменные	6

Создание триггера

Триггер в ADVANTA содержит указание, когда он должен сработать, набор проверок, когда нужно выполнить набор действий, и сам набор действий.

Кнопка создания триггера расположена в разделе «**Управление триггерами**».

The screenshot shows the 'Management of triggers' interface. On the left, there is a sidebar with search fields for 'Поиск', 'Тэги', 'Автор', and date ranges for 'Период создания триггера' and 'Период публикации триггера'. Below these are checkboxes for trigger status: 'Состояние триггера' (unchecked), 'С ошибками', 'Завершено по таймауту', 'Имеет неопределенные задачи', and 'Не опубликованные'. On the right, three trigger versions are listed:

- Версия № 10 C6568A** (опубликована 23 декабря 2021 г.)
- 01. Запрос утверждения БП** #БП
- 0 Успешно, 0 Ошибка, 0 В очереди, 0.008С Среднее время
- Системный Администратор
- Версия № 2 3F33C7** (опубликована 22 ноября 2021 г.)
- 02. Создание следующего периодического совещания** #Периодическое совещание
- 0 Успешно, 0 Ошибка, 0 В очереди, 0.000С Среднее время
- Системный Администратор
- Версия № 6 AFA46** (опубликована 23 декабря 2021 г.)
- 03. Статус проекта в дискуссии**

A blue circle with a white plus sign is overlaid on the bottom right corner of the interface, indicating where to click to create a new trigger.

Страница создания нового триггера

The screenshot shows the 'Create new trigger' form. At the top, there is a breadcrumb navigation: Главная > Триггеры > Добавить. To the right, it says 'Системный Администратор'. The form contains the following fields:

- Название ***: input field
- Описание**: input field
- Тэги**: dropdown menu
- Тайм-аут, сек**: input field
- Порядок обработки событий**: dropdown menu
- Через очередь данного триггера**: dropdown menu
- Выполнять повторно при сбое**: toggle switch (unchecked)
- События**: dropdown menu
- Условие**: code editor containing the following code:

```
1 return true;
```

Содержит следующие поля:

1. **Название**
2. **Описание**
3. **Тэги** — используются для дальнейшего поиска триггера в списке на странице «Управление триггерами»
4. **Таймаут, сек**

5. Выпадающий список «**Порядок обработки событий**» с 3 вариантами:

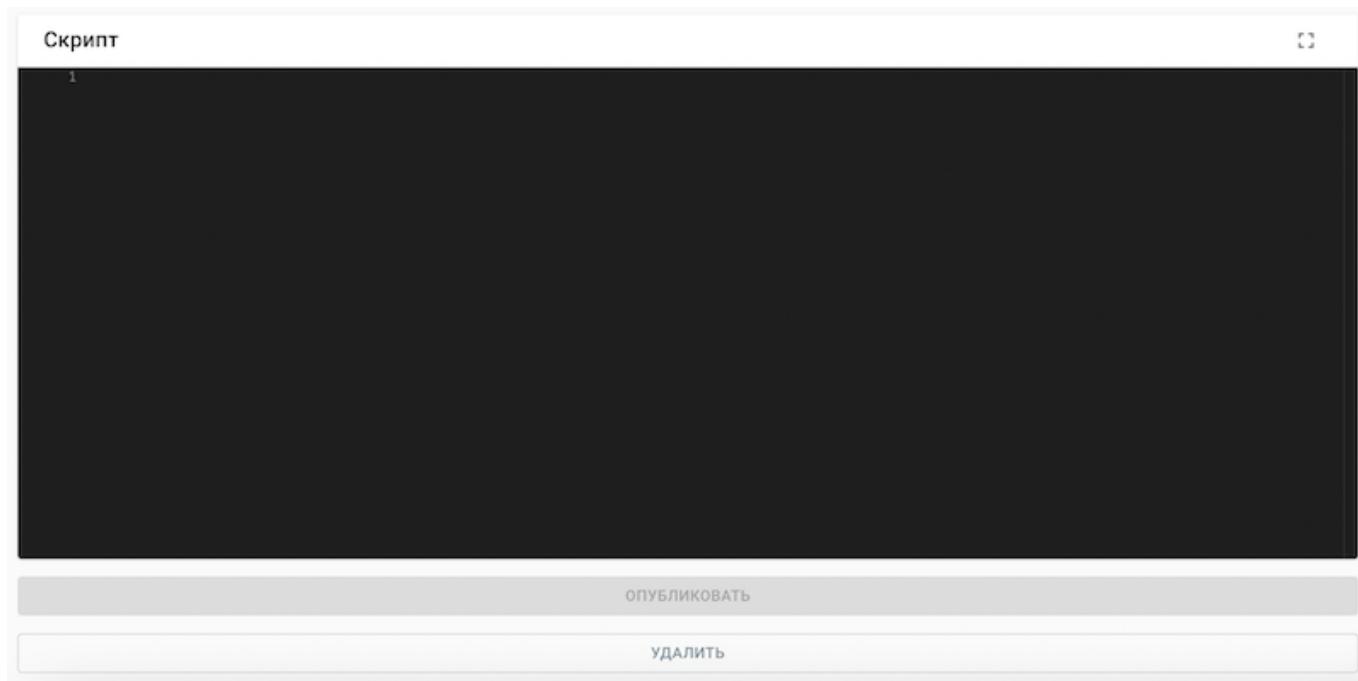
- **Через очередь данного триггера** – соответствующие события, на которые подписан триггер, обрабатываются в рамках одной очереди данного триггера: новое событие не будет обработано, пока не завершится обработка предыдущего события этого триггера.
- **Через глобальную очередь** – все события, на которые подписан созданный триггер, обрабатываются в рамках одной глобальной очереди модуля триггеров (каждый соответственно своим триггером).
- **Параллельно** – все события, на которые подписан триггер, обрабатываются параллельно (каждый соответственно своим триггером).

6. Переключатель «**Выполнять повторно при сбое**».

7. Выпадающий список «**События**» – перечень всех событий, которые генерирует Система при выполнении каких либо действий в ней. Исчерпывающий перечень событий на которые можно создать триггер.

8. Окно «**Условие**» – окно для записи, отображения и изменения кода C#.

В данном окне необходимо написать код условия, который должен возвращать либо значение false, либо значение true. Необходимо вернуть false, если скрипт выполнять не требуется.



9. Окно «**Скрипт**» – окно для записи, отображения и изменения кода C#.

В данном окне необходимо написать код основной логики (действий) триггера, который будет выполняться при выполнении Условия.

10. Кнопки «**Опубликовать**» и «**Удалить**».

Немного о создании триггеров в целом

Половина работы триггера: это понять, на какие события он должен реагировать.

Фильтрация событий проходит в несколько этапов с помощью 3 основных инструментов,

которые дополняют друг друга.

1. Простой фильтр на события, и связанные с ними объекты, который есть прямо в интерфейсе модуля триггеров в **выпадающем списке**.
2. **LINQ-запрос(ы)** - используются (при необходимости) в фильтрации событий и в самих скриптах. Подробнее о настройке LINQ-запросов - [здесь](#).
3. Дополнительная логика, прописанная в окне «**Условие**» для последней тонкой фильтрации событий.

События, которые попали под условия фильтров, запускают написанный скрипт. Результатом работы скрипта могут быть небольшие вычисления, создание новых записей в справочнике, новых объектов, изменение статусов, изменение значений реквизитов, их блокировка и т.д. Здесь мы описали [примеры готовых решений](#) на основе триггеров.

В Системе можно создать [источник данных LINQ](#) и задать ему референсный ключ, чтобы в дальнейшем по нему вызывать этот источник в **Условии** триггера или в самом **Скрипте**.

Вызов API в триггерах

Триггеры работают через [интеграционное API](#) ADVANTA ⇒ Всё, что можно сделать через API, триггер может сделать в системе.

Перед вызовом API нужно везде указывать `await`, иначе произойдет инициация кода вызова без ожидания результата.

Создание нового объекта - пример:

```
var newItem = new Api.Projects.CreateProjectDataContract
{
    // some code ...
};

// create new Item using ADVANTA API
var newItemID = await Api.Projects.CreateProjectAsync(newItem);
```

Триггер по мере необходимости создает SOAP-клиенты для взаимодействия с SOAP API (иначе пришлось бы перед каждым вызовом API создавать нужные SOAP-клиенты). По завершению триггера все открытые триггером SOAP-клиенты закрываются.

Если не дождаться через `await` вызова API, то может произойти завершение работы триггера раньше, чем завершится вызов.

В итоге произойдет ошибка при вызове.

Локальные функции

Если требуется определить собственную локальную функцию/процедуру для использования в скрипте триггера, то ее описание должно быть сделано с использованием префикса `async` в

следующем формате

```
async Task<string> MyFunctionName()
{
    // your code ...
    return result;
}
```

где вместо `string` необходимо указать тип переменной, которая будет возвращаться из локальной функции.

Если возвращать из функции ничего не требуется, то определение функции будет в формате:

```
async Task MyFunctionName()
{
    // your code ...
}
```

Для вызова созданной таким образом локальной функции/процедуры внутри скрипта триггера, рекомендуется использовать префикс `await` для ожидания ее завершения.

```
var result = await MyFunctionName();
```

Пример объявления локальной функции и ее вызова:

```
async Task CreateLog(string Text)
{
    TriggerConsole.WriteLine(Text);
    await System.Threading.Tasks.Task.Delay(1000);
}

for(int i = 0; i < 10; i++)
{
    await CreateLog(i.ToString());
}
```

Встроенные переменные

`Context.ApplicationId` - ID приложения Адванты, если настроено

`Context.EventId` - ID сообщения в шине

`Context.EventSentTime` - Время отправки сообщения из системы

`Context.Host` - Адрес системы, по которому она установлена

`Context.PersonId` - ID пользователя вызвавшего появление события

`Context.PrincipalId` - ID внутреннего пользователя системы (обычно не используется)

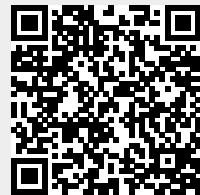
`Context.SessionId` - ID сессии, от которой модуль триггеров работает в системе. Нужен для передачи в API методы

Event - Содержит параметры события, на которое сделан триггер.
Контекстно определяется тем событием, на которое написан триггер.
В общем случае разные типы событий содержит разный набор полей.

[← К оглавлению](#)

[Мониторинг→](#)

From:
<https://wiki.a2nta.ru/> - Wiki [3.x]



Permanent link:
<https://wiki.a2nta.ru/doku.php/product/triggers/new>

Last update: **01.02.2024 11:41**