

RMG 365 | Dev

Tools

Набор инструментов разработчика ELMA365

- [Установка и настройка решения](#)
- [Логирование в скриптах](#)
 - [Пример вызова в скрипте через Imports и Namespace.action](#)
 - [Пример вызова в скрипте](#)
 - [Логирование данных в обработчике событий](#)
- [Получения режима окружения](#)
- [Полезное от WhatDaELMA365](#)
 - [Custom Loader](#)

Установка и настройка решения

Поддержка решения осуществляется на добровольной основе в канале [Telegram](#)

Настройки:

< **Модули** > Dev Tools 

Окружение*

Development

Просмотр файлов

xml

json

Адрес ELMA365

https://t0 [blurred]

Токен ELMA365

g [blurred]

Логин ELMA365

m [blurred]

Пароль

.....

Проверить

Сохранить

Выключить

Управление

где:

- Окружение - режим окружения системы (кроме Production будет отображаться вверху страницы). Доступные значения:
 - Development
 - Testing

- Stage
- PreProd
- Production
- Просмотр файлов - включает предпросмотр для файлов типа XML и JSON.
- Адрес ELMA365 и Токен ELMA365 - может использоваться как хелпер для вызова Web API.
- Логин ELMA365 и Пароль - может использоваться для вызова "ручек" фронта от имени конкретного пользователя

Кнопка проверить используется для проверки корректности ввода кредов выше:

Проверить

Публичный API: **200 OK**

```
{"success":true,"error":"","statusItems":[],"groupItems":[{"id":"00000000-0000-0000-0000-000000000000","code":"__default","name":""}]}
```

Внутренний API: **200 OK**

```
{"result":[],"total":46}
```

Логирование в скриптах

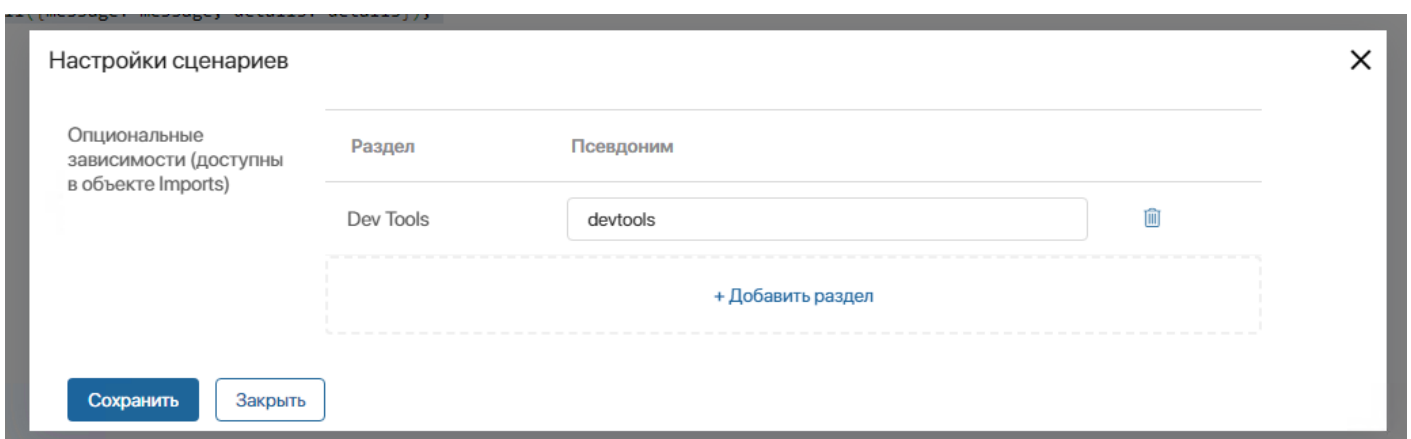
Пример вызова в скрипте через Imports и Namespace.action

Доступно с версии 2024.2

Вариант с использованием Imports:

```
async function action(): Promise<void> {  
    const message =  
    `${Context.data.__item!.namespace}.${Context.data.__item!.code}.${Context.data.__item!.id}`;  
    const details = JSON.stringify((<any>Context).json(), null, 2);  
    Imports?.devtools?.action?.save_to_log?.call({message: message, details: details});  
}
```

Нужно подключить зависимость и определить псевдоним, в данном примере это **devtools**:



Пример вызова в скрипте

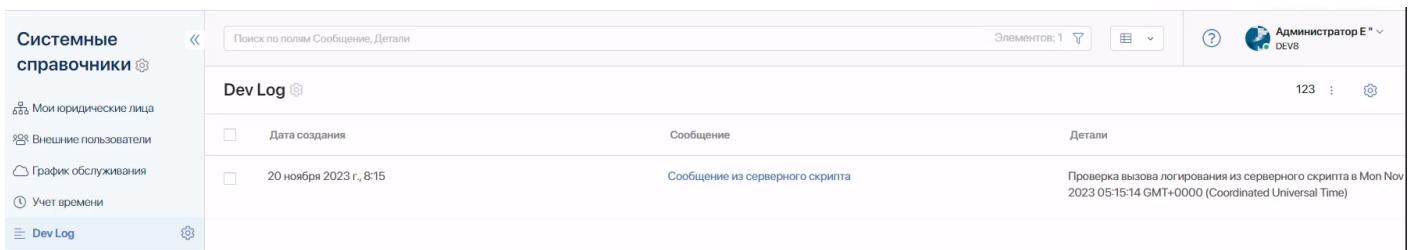
Логирование поддерживается как в клиентских, так и серверных скриптах.

На текущий момент нами был обнаружен единственный способ для решения данной задачи. Он не является изящным, но "на безрыбье и рак щука". Отдельное спасибо за идею Раису.

Для того, чтобы в лог (приложение в разделе системных справочников) попала информация, в скрипте необходимо вызвать следующий код:

```
const message = 'Сообщение из серверного скрипта';  
const details = 'Проверка вызова логирования из серверного скрипта в ' + Date().toString();  
await System.templater.generateText(Context, `{ExtText('82fe0bc6-0564-4b8a-a7d4-4c136cf8a949',  
'logger', '${message}', '${details}')}`);
```

Результатом выполнения будет следующее:



The screenshot shows a web application interface for 'Dev Log'. On the left is a sidebar with 'Системные справочники' and various system tools. The main area displays a table with one log entry. The table has columns for 'Дата создания', 'Сообщение', and 'Детали'. The entry shows a date of '20 ноября 2023 г., 8:15', a message 'Сообщение из серверного скрипта', and detailed information about the log call.

Дата создания	Сообщение	Детали
20 ноября 2023 г., 8:15	Сообщение из серверного скрипта	Проверка вызова логирования из серверного скрипта в Mon Nov 2023 05:15:14 GMT+0000 (Coordinated Universal Time)

Сообщение из серверного скрипта



Детали Проверка вызова логирования из серверного скрипта в Mon Nov 20 2023 05:15:14 GMT+0000 (Coordinated Universal Time)

Дата создания 20 ноября 2023 г., 8:15

Автор  [Администратор ELMA](#)

Задачи ^

Текущие Все

+ Задача

Лента ^



Сообщение

Удалить

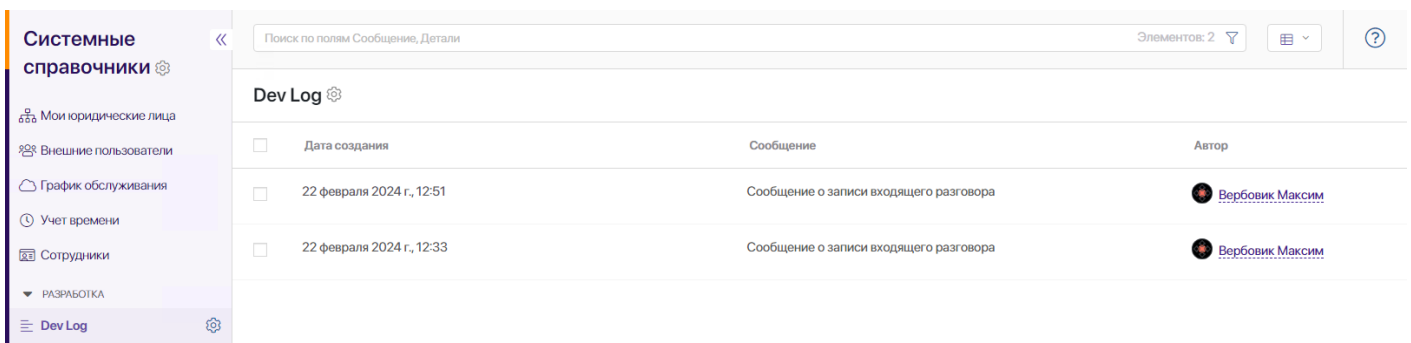


Логирование данных в обработчике событий

Для того, чтобы понять, какие данные приходят в обработчик событий в модуле, можно использовать следующий код:

```
async function action(): Promise<void> {  
    const message = 'Сообщение о записи входящего разговора';  
    const details = JSON.stringify(<any>Context.json(), null, 2);  
    await System.templater.generateText(Context, `${ExtText('82fe0bc6-0564-4b8a-a7d4-4c136cf8a949', 'logger', '${message}', ${details})}`);  
}
```

В результате, в логе будет следующая информация:



The screenshot shows a web application interface with a sidebar on the left containing navigation items like 'Системные справочники', 'Мои юридические лица', 'Внешние пользователи', 'График обслуживания', 'Учет времени', and 'Сотрудники'. The main content area displays a 'Dev Log' table with a search bar at the top. The table has columns for 'Дата создания', 'Сообщение', and 'Автор'. Two log entries are visible, both dated '22 февраля 2024 г.' and authored by 'Вербовик Максим'.

Дата создания	Сообщение	Автор
22 февраля 2024 г., 12:51	Сообщение о записи входящего разговора	Вербовик Максим
22 февраля 2024 г., 12:33	Сообщение о записи входящего разговора	Вербовик Максим

Сообщение о записи входящего разговора

Дата создания 22 февраля 2024 г., 12:51

Автор  [Вербовик Максим](#)

```
{
  "__id": "26caaff-cb42-483f-96ee-78257505ca92",
  "__item": {
    "namespace": "test",
    "code": "testdoc",
    "id": "2e6f6d87-f5bc-4e32-a1ba-706d085f325d"
  },
  "__itemName": "C-0015 ELMA Upgr2023 - ТехСУ - ИКС Холдинг.docx",
  "__name": "add_comment",
  "__description": "Добавление комментария к элементу приложения \"C-0015 ELMA Upgr2023 - ТехСУ - ИКС Холдинг.docx\"",
  "__requestId": "44392b162520af90",
  "__data": {
    "topicId": "5e2a7a50-748f-47b9-b336-25dd45a6ba00",
    "message": "<p>сообщение от меня в ленту</p>",
    "topicIndex": 0,
    "files": [],
    "createdAt": "2024-02-22T09:51:46.703016406Z",
    "isManual": true,
    "authorID": "f269c084-f7a3-48dc-85a4-a67860517d35",
    "title": ""
  },
  "__createdAt": "2024-02-22T09:51:46Z",
  "__createdBy": "f269c084-f7a3-48dc-85a4-a67860517d35",
  "__createdByName": "Вербовик Максим Арнольдович",
  "__createdByIp": "192.168.56.253",
  "topicId": "5e2a7a50-748f-47b9-b336-25dd45a6ba00",
  "message": "<p>сообщение от меня в ленту</p>"
}
```

Удалить



Задачи ^

Текущие Все

+ Задача

Лента ^



Сообщение

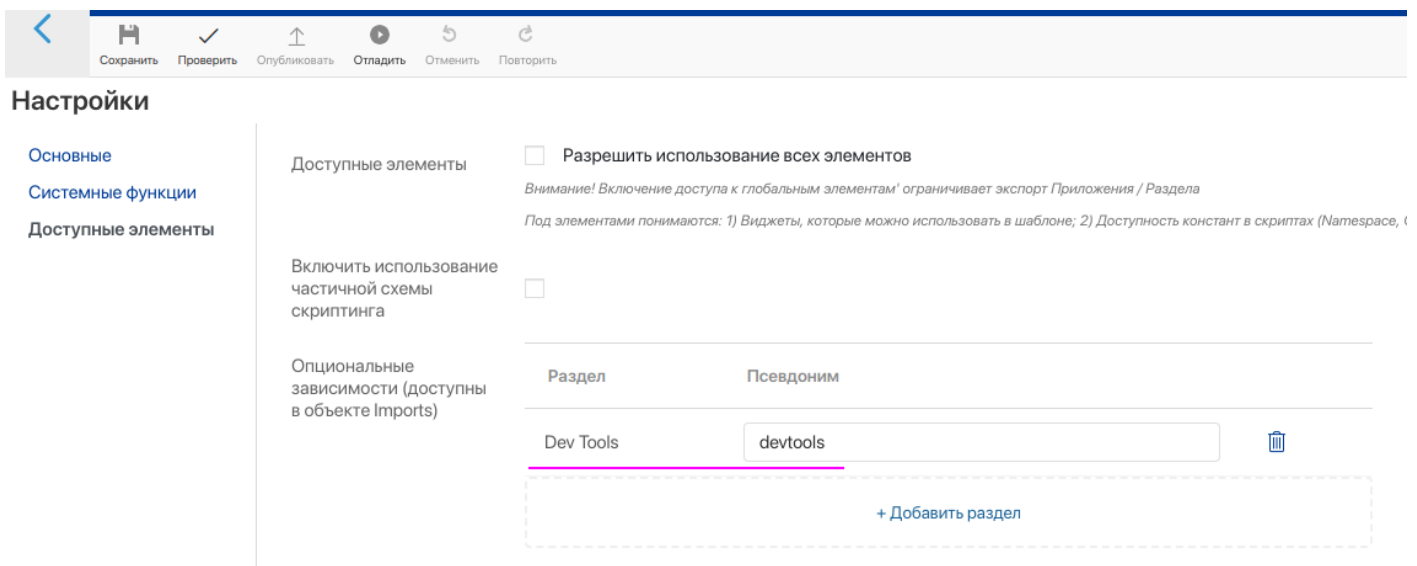
Получения режима окружения

Чтобы в своих решениях понять в каком режиме работает система необходимо в скриптах вызвать следующий метод:

```
let env_mode = 'Production';
const result = await Imports?.devtools?.action?.env_mode?.call( { for_service: true } )
if (result?.mode)
    env_mode = result.mode;
```

for_service указывает возвращать ли только 2 варианта: Staging или Production (Development не возвращается, т.к. зарезервировано для отладки), если не указано или false, то возвращаются все значения согласно настроек модуля.

devtools для импорта указан для примера и задается здесь:



The screenshot shows a settings interface with a top navigation bar containing icons for Save, Check, Publish, Debug, Cancel, and Repeat. Below the navigation bar, the 'Настройки' (Settings) section is active, with a sidebar menu showing 'Основные', 'Системные функции', and 'Доступные элементы'. The main content area is titled 'Доступные элементы' and contains the following options:

- Разрешить использование всех элементов
Внимание! Включение доступа к глобальным элементам ограничивает экспорт Приложения / Раздела
Под элементами понимаются: 1) Виджеты, которые можно использовать в шаблоне; 2) Доступность констант в скриптах (Namespace, t
- Включить использование частичной схемы скриптинга
- Оptionальные зависимости (доступны в объекте Imports)

Below these options is a table with two columns: 'Раздел' (Section) and 'Псевдоним' (Alias). The table contains one entry:

Раздел	Псевдоним
Dev Tools	devtools

At the bottom of the table, there is a dashed box containing the text '+ Добавить раздел' (Add section).

Полезное от WhatDaELMA365

[Сайт](#)

[TG канал](#)

Ежедневные находки и приёмы для работы в ELMA365: скрипты, шаблоны, автоматизация и быстрые решения.

Custom Loader

Пример, как выглядит режим загрузки данных:

В клиентский скрипт нужно добавить следующий код в самом начале:

```
declare const window: any;
```

Для показа и скрытия анимации необходимо использовать следующий код, например:

```
async function updateOrganizations(): Promise<void> {  
  
    // Показать loader  
    window.CustomLoader.show();  
  
    ViewContext.data.show_organizations_update = true;  
    await Server.rpc.updateOrganizations();  
    ViewContext.data.show_organizations_update = false;  
  
    // Скрыть loader  
    window.CustomLoader.hide();  
}
```

Данный код работает только в клиентских скриптах и не работает в серверных.