Инструкция по развертыванию

автоматизированной системы

«Система воспроизведения аудиовизуального контента в сети Интернет»

Оглавление

[1 Аннотация 3](#_Toc149921199)

[2 Используемые термины и сокращения 3](#_Toc149921200)

[3 Развертывание АС 4](#_Toc149921201)

[3.1 Подготовка к установке 4](#_Toc149921202)

[3.1.1 Установка Node.js и менеджера пакетов 4](#_Toc149921203)

[3.1.2 Установка TypeScript 4](#_Toc149921204)

[3.1.3 Настройка доступа к Nexus для сотрудников 5](#_Toc149921205)

[3.2 Установка АС 5](#_Toc149921206)

[3.2.1 Установка пакета АС 5](#_Toc149921207)

[3.2.1.1 Для сотрудников 6](#_Toc149921208)

[3.2.1.2 Для внешних потребителей 6](#_Toc149921209)

[3.2.2 Подключение компонентов АС 6](#_Toc149921210)

[3.2.2.1 Player 7](#_Toc149921211)

[3.2.2.2 Core 7](#_Toc149921212)

[3.2.2.3 Плагины 8](#_Toc149921213)

[3.2.3 Типы событий в АС 8](#_Toc149921214)

[3.2.4 Сборка артефактов и деплой 11](#_Toc149921215)

[3.2.5 Ошибки при установке АС 12](#_Toc149921216)

[3.2.6 Ошибки при эксплуатации АС 12](#_Toc149921217)

[3.2.7 Резервное копирование АС 12](#_Toc149921218)

[3.2.8 Проверка целостности сохраненных данных 12](#_Toc149921219)

[3.3 Обновление АС 12](#_Toc149921220)

[3.4 Удаление АС 12](#_Toc149921221)

# Аннотация

Данный документ содержит сведения по развертыванию автоматизированной системы «Система воспроизведения аудиовизуального контента в сети Интернет» (далее – АС) на стороне Пользователя.

# Используемые термины и сокращения

| **Термин** | **Определение** |
| --- | --- |
| Base64 | Стандарт кодирования двоичных данных при помощи только 64 символов ASCII. |
| HLS | Протокол для передачи аудиовизуального контента. |
| HLS.js | Библиотека языка программирования JavaScript. |
| JavaScript | Mультипарадигменный язык программирования. |
| Node.js | Кроссплатформенная среда выполнения JavaScript с открытым исходным кодом. |
| NPM | Менеджер пакетов JavaScript, входящий в состав программной платформы Node.js. |
| Stencil | Программное обеспечение для разработки и объединения компонентов автоматизированной системы. |
| TypeScript | Язык программирования, расширяющий возможности языка JavaScript. |
| Yarn | Менеджер пакетов JavaScript. |
| АС | Автоматизированная система. |
| Менеджер пакетов | Набор программных средств, позволяющих управлять процессом установки, удаления, настройки и обновления компонентов программного обеспечения. |
| Парсинг | Автоматический сбор информации. |
| Плагин | Программный модуль, который подключается к основной программе и дополняет возможности программы. |
| Пользователь | Пользователь АС. |
| Стейт (State, состояние) | Поведенческий шаблон проектирования, который позволяет объектам менять поведение в зависимости от своего состояния. |

# Развертывание АС

## Подготовка к установке

### Установка Node.js и менеджера пакетов

Необходимо загрузить и установить программную платформу [Node.js](https://nodejs.org/ru/download) версии 14 и выше. Вместе с Node.js установится менеджер пакетов npm. Также можно установить менеджер пакетов [yarn](https://yarnpkg.com/getting-started/install).

Чтобы проверить наличие установленных программ, необходимо выполнить в командной строке:

node -v # для Node.js

npm -v # для npm

yarn -v # для yarn

Если установка прошла успешно, командная строка отобразит версии программ. Например:

v18.17.1 # для Node.js

9.6.7 # для npm

1.22.19 # для yarn

### Установка TypeScript

Для установки TypeScript выполните команду:

* через npm:

npm install typescript --save-dev

* через yarn:

yarn add typescript --dev

### Настройка доступа к Nexus для сотрудников

Зависимости находятся в локальном репозитории registry (Nexus). Для каждого сотрудника доступы к Nexus выдаются индивидуально. После получения доступов, необходимо сгенерировать токен стандарта Base64. Для этого необходимо открыть консоль браузера и ввести:

btoa('login:pass')

Если используется VS Code и его встроенный терминал, переменную можно указать в settings.json (terminal.integrated.env.\*). Например, для macOS это будет выглядеть так:

"terminal.integrated.env.osx": {

"NEXUS\_TOKEN": "auth\_token",

},

Если используется WebStorm, необходимо:

1. Открыть WebStorm → **Preferenses** → **Tools** → **Terminal** и в поле **Environment Variables** указать:

NEXUS\_TOKEN=auth\_token

1. Перезагрузить WebStorm.

## Установка АС

### Установка пакета АС

#### **Для сотрудников**

АС упакована в единый пакет. Все зависимости находятся внутри пакета. Чтобы установить АС, необходимо:

1. Перейти в папку, в которой должна быть установлена АС.
2. Выполнить команду:

* через npm:

npm install @player/plugins-stencil @player/player-stencil

* через yarn:

yarn add @player/plugins-stencil @player/player-stencil

#### **Для внешних потребителей**

Чтобы установить АС из архива, необходимо:

1. Распаковать архив с АС.
2. В код страницы добавить:

import 'АБСОЛЮТНЫЙ-ПУТЬ-ДО-ПАПКИ-АРХИВА/@player/player/dist/player/player.esm';

import 'АБСОЛЮТНЫЙ-ПУТЬ-ДО-ПАПКИ-АРХИВА/@player/plugins/dist/esm/plugins';

<mini-player style={{ position: 'relative', width: '100%' }}>

<player src="video url">

</player>

</mini-player>

### Подключение компонентов АС

#### **Player**

Player – это главный веб-компонент для использования АС. Чтобы подключить его, в код страницы необходимо добавить:

|  |
| --- |
| <player src="video-url" autoplay videostyles>  </player> |

Атрибуты компонента Player:

| **Название** | **Тип данных** | **Описание** | **Значение по умолчанию** |
| --- | --- | --- | --- |
| src | string | Ссылка на видео. | - |
| autoplay | boolean | Включение или выключение автоматического воспроизведения. | false |
| custom | boolean | Использование кастомных пользовательских элементов. | false |
| hotkeysenabled | string | Включение или выключение горячих клавиш | - |

#### **Core**

Core (ядро) – основная часть АС, класс, к которому подключаются плагины. Core состоит из:

* Стейт;
* Шина событий – конвейер, по которому поступают события. Шина событий объединяет компоненты АС на основе событий, и это упрощает создание масштабируемой системы, управляемой событиями. Шина событий позволяет осуществлять обмен данными между компонентами с помощью механизма публикации и подписки, при котором компоненты явно не взаимодействуют друг с другом;
* Проинициализированные плагины.

Чтобы подключить Core, в JS-коде необходимо:

1. Импортировать класс Core:

import { Core } from '@player/player/dist/player/index.esm';

1. Создать экземпляр класса Core:

class CoreInstance extends Core {}

const core = new CoreInstance();

#### **Плагины**

Плагины добавляются и реализуются в папке @player/plugins. Чтобы импортировать плагины, выполните команду:

import ‘@player/plugins/dist/esm/plugins’;

### Типы событий в АС

**Play** – вызывается при нажатии кнопки play, при включенном autoplay или при нажатии на контейнер плеера (если плеер в состоянии паузы);  
**Pause** – вызывается при нажатии кнопки pause или при нажатии на контейнер плеера (если плеер в состоянии воспроизведения);  
**TimeUpdate** – вызывается при инициализации плеера или при событии элемента <video> “timeupdate”;  
**TimeChange** – вызывается при изменении времени воспроизведения, посредством клика/перемещения курсора по прогресс-бару;  
**TimeChangeActive** – вызывается при перемещении ползунка (во время активной фазы) по прогресс-бару;  
**VolumeChange** – вызывается при изменении звука с помощью ползунка звука;  
**Mute** – вызывается при нажатии иконки звука (если звук включен);  
**Unmute** –  вызывается при нажатии иконки звука (если звук выключен);  
**Lock** – вызывается при нажатии иконки открытого замка в плагине plugin-lock (для блокировки интерфейса плеера, чтобы сделать его некликабельным);  
**Unlock** – вызывается при нажатии иконки закрытого замка в плагине plugin-lock (для разблокировки интерфейса плеера, чтобы сделать его кликабельным);  
**NextVideo** – вызывается при клике иконки перехода к следующему видео (ui-компонент для переключения видео на следующее);  
**LoadedMetaData** – вызывается при вызове нативного эвента <video> элемента “loadedmetadata” для передачи duration (времени воспроизведения видео), также вызывается при инициализации плеера и при смене src-атрибута корневого компонента плеера;  
**OpenFullScreen** –  вызывается при нажатии иконки полноэкранного режима, также вызывается при открытии полноэкранного режима с помощью горячей клавиши;  
**CloseFullScreen** – вызывается при нажатии иконки закрытия полноэкранного режима, также вызывается при закрытии полноэкранного режима с помощью горячей клавиши;  
**SetPlayerMode** – вызывается при инициализации плеера при определении мобильного или десктопного плеера, также вызывается при смене режима плеера на мини-режим и обратно;  
**SpeedChange** – вызывается при смене скорости видео через модальное окно настроек (которое открывается через иконку с шестеренкой);  
**VideoElementCreated** – вызывается при инициализации плеера, также вызывается  при смене src-атрибута корневого компонента плеера;

**VideoElementDestruct** – вызывается при анмаунте/деактивации плеера, также вызывается при смене src-атрибута корневого компонента плеера;

**InitializeQuality** – вызывается при парсинге HLS-манифеста при получении качества видео;  
**SetQuality** – вызывается при выборе качества видео через модальное окно настроек;  
**OpenMiniPlayer** – вызывается при нажатии иконки открытия режима миниплеера;  
**CloseMiniPlayer** – вызывается при закрытии миниплеера через иконку крестика;  
**ExpandMiniPlayer** – вызывается при возвращении в обычный режим плеера из мини-режима с помощью иконки со стрелочкой;  
**BufferAppended** – вызывается при инициализации видео (для зануления), при смене src-аттрибута корневого компонента плеера (для зануления) и при событии “progress” <video> элемента для добавления буфера (загруженной части видео);  
**SetTrickModeUrl** – вызывается при инициализации плеера для установки ссылки на trickmode, тажке при смене src-атрибута корневого компонента плеера;  
**ControlsVisible** – вызывается при смене src-атрибута корневого компонента плеера (скрывает контроллы), также вызывается при движении мыши по зоне плеера (показывает контроллы), также при бездействии мыши в течение 3 секунд (скрывает контроллы);  
**ControlsHold** – вызывается при зажатии прогресс-бара (касанием) на мобильном плеере со значением (true), также при отжатии касания вызывается (false);  
**ControlsHoldWithTimeout** – вызывается при отмене касания прогресс-бара на мобильном плеере для скрытия контроллов;  
**TogglePlayPauseOnContainer** – вызывается при открытии модального окна настроек, для блокировки плей/паузы клика по контейнеру;  
**Buffering** – событие для сторонних плагинов, для передачи в прогресс-бар информации о буферизации;  
**VideoStarted** – вызывается при первом воспроизведении видео (из состояния стоп в состояние воспроизведения);  
**VideoPlayed** – вызывается при вызове нативного события <video> элемента “play”;  
**VideoPaused** – вызывается при вызове нативного события <video> элемента “pause”;  
**VideoEnded** – вызывается при вызове нативного события <video> элемента “ended”;  
**PlayError** – вызывается при неудачной попытке вызова функции “play()” <video> элемента;  
**PlayPause** – вызывается при необходимости сделать быстрый play -> pause по плееру;  
**StartLoad** – вызывается при вызове нативного события <video> элемента “loadstart”;  
**SetHotkeysEnabled** – вызывается при включении горячих клавиш посредством разблокировки/блокировки UI с помощью  дополнительного компонента замка;  
**ManifestParsed** – вызывается при парсинге манифеста HLS.js библиотекой;  
**Online** – вызывается при вызове события window “online” для определения наличия интернет-соединения;  
**Offline** – вызывается при вызове события window “offline” для определения отсутствия интернет-соединения;  
**CanPlayThrough** – вызывается при вызове нативного события <video> элемента “canplaythrough”;  
**NativeError** – вызывается при вызове нативного события <video> элемента “error”;  
**HlsError** – вызывается при вызове события Error библиотеки HLS.js;

**Playing** – вызывается при вызове нативного события <video> элемента “playing”;  
**PreloaderShow** – вызывается при рендере (отображении) preload-компонента;  
**PreloaderHide** – вызывается при анмаунте (деактивации) preload-компонента.

### Сборка артефактов и деплой

Для сборки артефактов необходимо выполнить команду:

pnpm run build

Деплой осуществляется с помощью [Gitlab Pipelines](https://docs.gitlab.com/ee/ci/pipelines/).

### Ошибки при установке АС

Если в результате установки АС возникла ошибка, необходимо:

* Для сотрудников:

Удалить установленную АС (подробнее см. раздел Удаление АС) и установить ее заново (подробнее см. раздел Для сотрудников).

* Для внешних потребителей:

Удалить распакованный архив и распаковать его заново.

### Ошибки при эксплуатации АС

Если в результате эксплуатации АС возникла ошибка, необходимо перезапустить сборку приложения, удалить и затем заново установить АС.

### Резервное копирование АС

Чтобы создать резервную копию АС, необходимо распаковать архив с АС или переустановить пакет, последовательно выполнив команды:

npm uninstall @player

npm install @player

### Проверка целостности сохраненных данных

Проверить целостность сохраненных данных можно с помощью команды npm audit или сравнив файлы из архива с папкой АС.

## Обновление АС

Для обновления АС необходимо установить нужную версию через npm install/yarn add.

## Удаление АС

Для удаления развернутой АС необходимо выполнить команду:

* через npm:

npm uninstall @player

* через yarn:

yarn remove @player