xtendx®
corporate media solutions

simplex PDK Documentation

Entities, Modules, Parameters, API

contact xtendx:

support@xtendx.com

xtendx®
corporate media solutions

Table of Content

General Information

The simplex PDK was created to support a seamless player integration into a customer web presence without the need for iframes.

Another reason is a desire to reduce time & required knowledge (for external developers) of new player creation. The goal was to create a new player using HTML & CSS without JS manipulation. The result is the Angular based Player Development Kit (NG PDK).

Angular with its declarative approach forms a flexible product basis.

Users can easily set several attributes on an HTML element & profit from the full functionality.

Basic entities

The NG PDK consists of the following entities:

Directive

An entity that allows to bind required functionality directly to an HTML element. To bind features, directive name are used as HTML element attribute.

Example:

```
<div data-pdk-playbutton="paused" class="play-button paused"></div>
<h1 data-pdk-channel-title></h1>
<div ng-if="enableSearch"></div>
```

Some attributes require additionally passed values for component state changes.

In the first example, the play button element has a default "paused" state and automatically added a class with the same name that switches CSS style of the element.

In case the playback state of the video is changed, the default class is toggled.

The directive from the second example doesn't require any additional values to be passed & just fills up its element with channel title text.

The third example shows an Angular built-in directive, that can be used for the element's conditional visibility switching. In this example the expression can be interpreted as:

"display the element if the enableSearch parameter has value TRUE."

Service

This entity contains basic player functionality like server side communication or internal data handling. End user should not make changes in services.

Controller

An additional entity that must be added to HTML as an attribute. It allows to define player functionality areas like channel view containing element and defines player specific logic.

Filter

Allows to convert passed data to the required representation.

Available modules

The following table contains the available functional modules of NG PDK.

Video modules

Name	Attribute	Description
Video elements container	data-pdk-video data-pdk-priority="html5" data-pdk-live-state-container="ONDEMAND LIVE"	Defines video playback logic. Inserts <video> and <object> (for flash) elements to specified container. Supported states & technology priority can be defined via described attributes.
Big play button	data-pdk-big-playbutton	Directive contains build-in control visibility rules. Has logic of transformation to Big Pause Button
Playback toggle container	data-pdk-video-playbutton	Toggles video playback state (play/pause). Can be used on any element (poster, overlay etc.)
Video poster	data-pdk-poster	Directive contains build-in display rules. Contains functionality of poster URL update
Video subtitles container	data-pdk-subtitles	Contains subtitles rendering logic and display rules
Video speaker container	data-pdk-speaker-container	Contains speaker rendering functionality based on current video time. Directive contains build-in control visibility rules.
Video speaker data	data-pdk-speaker-data="image" data-pdk-speaker-data="title,firstName,lastName" data-pdk-speaker-data="jobtitle,company"	Directive contains functionality of speaker image and text data rendering.
Video speaker delimiter	data-pdk-speaker-delimiter="delimiter-symbol"	Directive is used to set user-defined symbol to separate speaker's data

Control bar modules

Name	Attribute	Description
Control bar container	data-pdk-controlbar="show-class-name" data-pdk-controlbar-hidden-class="hide-class-name"	Defines container for control bar elements storing. Visibility classes setting can be configured via passed attributes values.
Play button	data-pdk-playbutton="state-class-name"	Toggles video playback state (play/pause). Uses "state-class-name" as CSS class for switching control view
Mute button	data-pdk-mute="state-class-name"	Toggles video mute state (muted/unmuted). Uses "state-class-name" as CSS class for switching control view
Volume bar	data-pdk-volumebar	Directive contains build-in control visibility rules and volume level update logic
Volume line	data-pdk-volumebar-current="state-class-name"	Visualizes current volume level. Uses "state-class-name" as CSS class for switching control view bind with mute state
Volume button	data-pdk-volumebar-button	Allows to change volume level via drag & drop
Fullscreen button	data-pdk-fullscreen-button="state-class-name"	Switches player fullscreen state by user click/touch. Uses "state-class-name" as CSS class for switching control view
Fullscreen container	data-pdk-fullscreen-container="state-class-name"	Defines element which will be displayed in fullscreen mode. Directive contains logic of fullscreen mode requesting & canceling. Uses "state-class-name" as CSS class for switching container styles
HD button	data-pdk-hd-button="state-class-name"	Toggles video quality SD/HD. Directive contains build-in control visibility rules. Uses "state-class-name" as CSS class for switching control view

Subtitle button	data-pdk-cc-button="state-class-name"	Toggles subtitles language menu. Directive contains build-in control visibility rules. Uses "state-class-name" as CSS class for switching control view
Subtitle languages menu	data-pdk-cc-langs="state-class-name"	Defines subtitle languages menu that contains list of available subtitle's languages. Directive contains build-in control visibility rules. Uses "state-class-name" as CSS class for switching control view
Subtitle language item	data-pdk-cc-lang="state-class-name"	Defines subtitle language item in subtitles menu. Directive contains build-in control visibility rules. Uses "state-class-name" as CSS class for switching control view
Video seek bar	data-pdk-seekbar	Defines container for seek bar elements. Contains logic of video current time update event triggering by user click/touch
Video progress bar	data-pdk-progress-bar	Defines container that visualizes video loading progress
Current video time bar	data-pdk-seekbar-current	Visualizes played amount of time & current video time
Video seek bar button	data-pdk-seekbar-button	Allows video current time change via button drag & drop
Video seek bar tool-tip	data-pdk-seekbar-tooltip="state-class-name"	Defines container for a tool-tip. Directive contains build-in display rules. Uses "state-class-name" as CSS class for switching control view
Video current time labe	data-pdk-seekbar-currenttime	Defines container for video current time label. Directive contains build-in display rules. Directive uses internal filter called "formatTime" (hh:mm:ss o mm:ss)
Video duration label	data-pdk-seekbar-duration	Defines container for video duration label. Directive contains build-in display rules. Directive uses internal filter called "formatTime" (hh:mm:ss o mm:ss)
Live state indicator	data-pdk-live-indicator	Used for live state indicator visualization. Directive contains build-in display rules.

Channel modules

Name	Attribute	Description
Channel title container	data-pdk-text-ellipsis="channel.active.title"	Defines container for channel name. Contains logic of long names cutting.
Channels list menu	data-ng-controller="ChannelListController" data-pdk-select="channels" data-pdk-select-value="channel.active" data-pdk-select-text="title" data-pdk-select-open-class="state-class-name" data-pdk-select-active-item-class="state-class-name"	Defines container for channel list menu. Added several internal containers in the main element. Contains all required menu logic.
Channel project container	data-pdk-channel-projects data-pdk-channel-projects-per-page="{{videosPerPage || 20}}" data-pdk-active-class="state-class-name"	Defines container for channel project item content. Used as repeatable template element to build projects grid. Contains logic of projects grid rendering & its configuration.
Channel project duration container	data-pdk-channel-project-duration	Used to store project duration in channel mode. Displays duration in HH:MM:SS format.
Channel project title container	data-pdk-channel-projects-title	Defines container for channel project title
Channel project description container	data-pdk-channel-projects-description	Defines container for channel project description
Channel project poster container	data-pdk-channel-projects-poster	Used to define project posterframe in channel mode. Must be set on <img> tag only. Contains logic of placeholder displaying in case any issues with project poster.
Channel project rating container	data-channel-player-rating-container	Used to define project rating container. Stores rating stars.

Channel project rating star container	data-pdk-channel-project-rating="5" data-pdk-channel-project-rating-active-class="state-class-name" data-pdk-channel-project-rating-voted-class="state-class-name" data-pdk-channel-project-rating-notch-range-class="state-class-name"	Defines container for project rating stars in channel mode. Contains logic of stars mode switching (read-only, voting allowed). Used as repeatable template element for all stars. Amount & state design can be configured via attributes.
Channel search input field	data-pdk-search-input	Defines input field for channel search. Should be used on <input> element.
Channel start search button	data-pdk-search-button	Defines control to be used to activate search of passed to search input string.
Channel clear search button	data-pdk-search-clear-button ng-disabled="!searchQuery"	Defines container for search clearing button. Contains logic of search results view resetting & search input clearing. Can be disabled via ng-disabled attribute in case search field is empty.
Paging control container	data-pdk-channel-projects-paging-container	Defines container for channel paging control. Contains logic of control displaying (appears when pages amount is more than 1 page) & control animation functionality.
Paging control back button	data-pdk-channel-projects-paging-back	Defines container for page back switching button. Contains displaying & page switching logic.
Paging control forward button	data-pdk-channel-projects-paging-forward	Defines container for page forward switching button. Contains displaying & page switching logic.
Paging control page buttons wrapper	data-pdk-channel-paginator-wrapper	Defines container for paginator wrapper. Contains logic of paging control resizing.
Paginator container	data-pdk-channel-projects-paginator data-pdk-channel-projects-paginator-button-width="32"	Defines container for paging buttons set. Contains logic of paging buttons size processing.

Paging button container	data-pdk-channel-projects-paging data-pdk-channel-projects-per-page="{{videosPerPage || 20}}" data-pdk-active-class="state-class-name" data-pdk-disable-class="state-class-name"	Defines container for paging buttons. Used as repeatable template element for all paging buttons. Contains logic of pages switching.

Ads modules

Name	Attribute	Description
Ad data container	data-pdk-adv-info-container	Defines a top level container for all ad elements. Contains build-in displaying logic based on ad state.
Ad remain time label container	data-pdk-adv-remain-time	Defines container for ad remain time label. Displays time & specified text.
Ad skip button container	data-pdk-adv-skip-button="10"	Defines container for ad skip button. Its appearance time can be configured via value (in seconds) passing to the attribute.

Project data modules

Name	Attribute	Description
Project title container	data-pdk-project-title	Contains project title rendering & displaying logic
Project description container (no scroll bar	data-pdk-project-description	Contains project description rendering & displaying logic
Project description container (with scroll bar)	data-pdk-scrollbar="metadata.description" ng-if="metadata.description"	Contains project description rendering & displaying logic. Automatically wraps content with scroll bar containers
Project author name container	data-pdk-project-owner	Contains author name container rendering & displaying logic based on availability of first & last author's name

Project modify/create date container	data-pdk-modified-ago	Contains project modify/create date rendering logic. Directive uses localization & "prettyDate" filters
Video podcast download button	data-pdk-download-button	Defines podcast download button displaying & download link construction logic
Project embed code container	data-pdk-share-embed	Contains logic of embed code processing received from SMS. Contains embed code rendering & selection logic. Returns embed code to specified <textarea> element
Project rating star	data-pdk-project-rating="amount-of-stars" data-pdk-project-rating-active-class="state-class-name" data-pdk-project-rating-voted-class="state-class-name"	Defines rating functionality & control rendering logic. It is enough to define only one star element in container and set stars amount. Directive will generate other stars according to defined amount.
Project logo container	pdk-header-logo	Required <img> element to be used for current directive. Directive renders logo according to build-in logic.
Project embed code copy button	data-pdk-clipboard-copy-button	Defines detection logic of HTML5 clipboard functionality & button rendering logic
Project slide container	data-pdk-slide	Directive defines slide rendering & resizing logic based on received data in ondemand & live modes. Directive is available on and <img> elements
Project sharing via e-mail container	data-pdk-share-mail	Contains logic of e-mail sharing URL generation. Directive expects to be used on <a> elements
Project sharing via Google+ container	data-pdk-share-google	Contains logic of G+ sharing URL generation. Directive expects to be used on <a> elements
Project sharing via Facebook container	data-pdk-share-facebook	Contains logic of Facebook sharing URL generation. Directive expects to be used on <a> elements
Project sharing via Twitter container	data-pdk-share-twitter	Contains logic of Twitter sharing URL generation. Directive expects to be used on <a> elements

Project state screen container	data-pdk-live-state-container="PREVIEW "	Defines container for project state screen. Contains build-in displaying logic based on project state. Screen name can be defined via attribute value and can have the following values: "PREVIEW", "LIVE", "POSTLIVE", "ONDEMAND".
Event date counter container	data-pdk-event-counter-container	Defines container for event date counter elements. Container build-in displaying logic based on activated event date in project settings & project state.
Event date counter label	data-pdk-event-counter	Defines container for event date time label storing. Updates element to display decreasing time.

Index panel modules

Name	Attribute	Description
Index panel container	data-pdk-index-container data-pdk-closed-class="state-class-name" data-ng-show="!isIndexPanelEmpty && !loading && showIndexPanel()"	Defines container of index panel & its displaying logic
Index panel tab button	data-pdk-index-tab-button="tab-name"	Defines tab button functionality & displaying rules. Directive can be supplied with the following tab names: cuepoints, slides, attachments, chat, polls, presenter-slides.
Index panel tab container	data-pdk-index-tab-container="tab-name " data-ng-if=" tab-name.length"	Defines tab container functionality & displaying rules. Directive can be supplied with the following tab names: cuepoints, slides, attachments, chat, polls, presenter-slides.
Index panel tab container item	data-pdk-index-tab-items="tab-name " data-pdk-active-class="state-class-name" data-pdk-playing-class="state-class-name"	Defines tab container item functionality & displaying rules. It can be any repeatable item like cuepoints, slides, attachments.
Tab container item time	data-pdk-index-tab-item-time	Defines container for item start time label.
Tab container item title	data-pdk-index-tab-item-title	Defines container for item title label.
Tab container item timeline	data-pdk-index-tab-item-timeline	Defines container for item duration timeline. It displays progress of current active item.
Tab container slide image	data-pdk-index-tab-slide-image	Defines container for slide thumbnail storing & its displaying logic.

Attachment action button container	data-pdk-index-tab-attachment-type="Download Link"	Defines container for attachment action button. Replaces its text & functionality according to passed types: Download & Link.
Attachment container	data-pdk-index-tab-attachment-link	Defines clickable container for attachment content. Should be used on <a> element.
Chat message container	data-pdk-index-chat-message	Define container for chat message & its detailed info. Used as template element for all chat messages. Directive contains functionality of messages list generation & state switching.
Message type label container	data-pdk-index-message-type	Used to store message type label. In default implementation displays only in case private messages send to user. Displays "PRIVATE" label as its content.
Message time label container	data-pdk-index-message-time	Defines container for message time displaying. Uses the following by default: DD:MM:YY HH:MM:SS
Message username label container	data-pdk-index-message-username	Displays message owner name. By default two values are used: Moderator & Guest.
Message text container	data-pdk-index-message-text	Defines container for message text storing.
Messages input field container	data-pdk-index-chat-message-input	Defines container as messages input. Primordial container will be replaced with textarea element containing appropriate attributes.
Message send button container	data-pdk-index-chat-message-send-button	Defines container for messages sending button. Contains functionality of message send event triggering.
Poll question text container	data-pdk-poll-question-text	Defines container for poll question text.
Poll answer container	data-pdk-poll-questionnaire	Defines container for poll answer text & input. Used as repeatable template element for all answers.

Poll answer input container	data-pdk-poll-input="checkbox,radio" name="pdk-poll-input" id="poll_{{currentPoll.id}}_{{$index}}"	Defines container for poll answer input. Uses two types: checkbox & radio button. Should contain generated id to make binding with related label that contains answer text.
Poll answer text container	data-pdk-poll-answer-text for="poll_{{currentPoll.id}}_{{$index}}"	Defines container for poll answer text. Must be put on <label> element only.
Poll vote button	data-pdk-poll-vote-button	Defines container for poll vote button. Contains functionality of poll vote event triggering.
Voted poll message container	data-pdk-voted-poll-message	Defines container for info message displayed after poll has been voted.
Poll results progress bar container	data-pdk-poll-results-progressbar	Used to store progress bar of poll answer result.
Poll results progress bar value container	data-pdk-poll-results-progressbar-value	Defiles container for label that displays percent value of poll answer result progress bar.
Presenter's presentations list drop-down menu container	data-pdk-presenter-select="projectPresentations" data-pdk-presenter-select-value="currentPresentation" data-pdk-presenter-select-text="name" data-pdk-presenter-select-open-class="state-class-name" data-pdk-presenter-select-active-item-class="state-class-name"	Defines container for presentations list drop-down menu. Displayed in presenter mode only. Allows to select available project presentation & initiate rendering of its slides list. Adds several dynamically generated containers.
Presenter's slide container	data-pdk-presenter-items="currentPresentation" data-pdk-active-class="state-class-name"	Defines container for rendering of current presentation slides in presenter mode. Used as repeatable template element for all slides in the panel. Selected slide styles are defined via passed class name in data-pdk-active-class attribute.

Service modules

Name	Attribute	Description
Scroll bar	data-pdk-scrollbar="[data, data]" data-ng-if="data"	Defines scrollable container. The first attribute should contain data set which change will update scroll bar. The second attribute can be used to hide scroll bar on property change.
Pre-loader	data-pdk-preloader	Defines container for with loading animation for not ready content hiding.
Container visibility toggle button	data-pdk-toggle-buttc n="container-visibility -state"	Defines button for passed container visibility switching. Attributes take a container state variable.
Toggle container	data-pdk-toggle-conta iner="container-visibi ity-state"	Defines switching container for toggle button. Attributes take a container state variable.
Fullscreen container	data-pdk-fullscreen-c ontainer="class-name " data-pdk-fullscreen-a dd-class-to="target-el ement"	Defines container with fullscreen mode switching logic. Uses data-pdk-fullscreen-add-class-to attribute as target element that goes fullsceen & adds class with required styles taken from data-pdk-fullscreen-container attribute.
Size holder container	data-pdk-size-holder	Defines container for size holder displaying. Takes video dimension & generates container padding value to hold video block size with correct aspect ratio.
Font scaling	data-pdk-scaled-font= "375" data-pdk-scaled-conta iner=".container-class "	Can be set on text comprising elements link speakers, subtitles etc. Scales font size according to passed container size change & scaling constant. Font scaling is triggered by window resize event.

Notification modules

Name	Attribute	Description
Project error container	data-pdk-project-error	Defines container for project error text storing. Displays in case no project data has been successfully loaded.
Project authorization error container	data-pdk-project-authorization-error	Binds logic of project authorization error message displaying to specified container. Messages texts are passed via template. Displays in case of any issues with authenticated project data access.
Index panel notification container	data-pdk-index-panel-notification data-pdk-index-panel-notification-chat-text="new chat msg" data-pdk-poll-notification-text="new poll"	Defines container for chat & poll notification messages. Text attributes specify notification messages for chat & poll that are displayed near new messages & poll amount labels.
Unread chat messages amount label container	data-pdk-index-tab-chat-notification	Defines container for unread messages amount label. Adds number of messages as text value of specified container.
Unread chat messages text container	data-pdk-chat-notification-text-container	Defines container for unread chat messages notification text. Text is used from text attribute passed to Index panel notification container.
New polls amount notification label	data-pdk-index-tab-poll-notification data-pdk-poll-notification-value="1"	Defines container for new polls amount notification label. Value is always equal to 1 because for now there can be only one active poll at the same time.
New poll notification text	data-pdk-poll-notification-text-container	Defines container for new poll notification text. Text is used from text attribute passed to Index panel notification container.

No flash available message container	data-pdk-live-noflash-message	Defines container for notification messages regarding flash absence in a browser. Contains build-in displaying logic based on flash plugin availability, project state & used browser/device.

Authorization modules

Name	Attribute	Description
Authorization form container	data-pdk-authorization-screen	Defines container for authorization form. Contains displaying logic based on project parameters.
Authorization form username input	data-pdk-authorization-screen-username-input	Defines container for authorization form username input. Should contain <input> element inside with set data-ng-model="userName" attribute to watch user text input actions.
Authorization form password input	data-pdk-authorization-screen-password-input	Defines container for authorization form password input. Should contain <input> element inside with set data-ng-model="authorizationPassword" attribute to watch user text input actions.
Authorization form submit button	data-pdk-authorization-screen-submit-button	Defines container for authorization form submit button. It initializes processing of username and password & project data loading.

Player Parameters

Basic information about parameters usage

When the player is called directly from the project folder on the Simplex server, all listed GET Parameters are optional.

The following GET parameters in this document can be set over data-pdk-config attribute on

a) the root player element,

b) Via URL,

c) in a configuration file or

d) in an index.html file of the player.

The mentioned places of parameters usage are listed by selection priority.

data-pdk-config attribute accepts parameters as a sharing=true&themeColor=6633FF pairs (similar to URL query string parameters format).

To set them in a configuration file, set over URL parameter the location to the configuration file: configUrl=http://your.server.com/filename.js.

The configuration file does need to have a js extension. This allows to generate dynamic configuration files for all your requirements and use cases. It needs to contain the following variable.

Within this variable the all GET parameters listed below can be defined as in this example:

```
var SimplexConfig = {
     "sharing": true,
     "themeColor": "6633FF"
}
```

In case if player instance was instantiated with namespace data-pdk-player="playerName" SimplexConfig variable can be also namespaced:

```
var SimplexConfig = {
     playerName: {
          "sharing": true,
          "themeColor": "6633FF"
     }
}
```

This will allow to configure several apps independently with a single SimlexConfig object. The Player app will try to find and use its own dedicated config in SimpleConfig object.

If there is no namespaced config, global SimplexConfig will be used for configuration.

The same SimplexConfig variable can be added to index.html file of the player and it will transfer all described parameters to the player.

Multiple player instantiation on arbitrary DOM nodes.

It is possible to instantiate multiple independent player apps on arbitrary DOM nodes.

Simply mark the DOM node or nodes with data-pdk-player attribute, like this:

```
<body data-pdk-player>
```

New player app instances will be automatically created on marked nodes. There's no need to add an ng-contoller attribute to the <html> node or do anything else anymore: initializing script will take care of the rest.

data-pdk-player attribute also accepts a string value. If this value is passed, it would be used for namespacing SimplexConfig configuration object in order to make independent player app configuration possible. So, if you're going to instantiate several player apps with different configurations and use single SimplexConfig object to do that, you may do the following:

```
<div data-pdk-player="playerName"></div>
<div data-pdk-player="anotherName"></div>
```

Also there's one more way to pass a number of config params to player app directly from HTML code: data-pdk-config attribute.

In order to do that you may do the following:

```
<div data-pdk-player="playerName" data-pdk-config="param string"></div>
<div data-pdk-player="anotherName"  data-pdk-config="another param string"></div>
```

Supported parameters

Flash specific parameters

Property	Description	Values
progressiveDownload	By setting this parameter to true, the Flash Player will switch to progressive download and display the video download progress-bar in the video controls. In case this parameter is not used or set to false, player will use streaming.	true/false or 1/0
syncLang	When doing a live stream and as well on the ondemand version, sometimes it is required to have two audio sources streamed. For example the original audio source and a translation. On the live stream this is solved by having the original audio source on the left channel and the translation on the right channel of the audio stereo signal. Same for the ondemand video. Our Flash default player has the option to make one channel muted and the other played in both. By setting syncLang=left, the left channel (in this example the original audio source) is audible in both audio channels, while the translation is muted. syncLang=right will have the original audio source muted and the translation audible in both audio channels.	left/right or de/fr
displayHLS	For debugging or control of the HLS behavior, setting this parameter to true, will display in the top left corner an indicator with the video quality delivered over HLS and send events to the browsers developer console. Pay attention: it requires activated debug parameter.	true/false or 1/0
debug	Write extended log to a developer's console of the browser. Log contains information about player loading process, files transfer & state changing.	true/false or 1/0
amf	It is possible to disable the AMF3 calls the Flash Player uses to communicate with the Simplex Server. Please note that when amf=0 is set, live features (chat, slides, state changing, polls) and authentication features will not work.	true/false or 1/0
enableLDAP	Use it to enable LDAP support. After it is enabled project with LDAP protection will be correctly processed by the player. Deprecated starting from NG PDK 1.3 because of authentication method auto detection.	true/false or 1/0
enableQS	Enable/disable quality switcher for Live stream with HLS. The Quality selector for HDS should not be shown by default.	true/false or 1/0
disableStreaming	Enable/disable AMF3 streaming. In case a player uses polling by default you can change its behavior to use streaming only. Latest players use streaming by default.	true/false
multicastfallback	Allows to activate fallback to unicast mode. When activated (has true/1 value passed) and if no connection can be established player will fallback to unicast mode. In case false/0 values have been passed player will not use fallback anymore. By default player doesn't use fallback to unicast.	true/false or 1/0

liveServerUrl	Allows to override server url in live mode for Flex Player. Link to RTMP stream will be generated according to this rule: "rtmp://" + liveServerUrl + "/" + application + "/" + streamName	""

General parameters supported by both Flash & HTML

serverUrl	This parameter needs only to be set, if player is not hosted on the Simplex Server which delivers the project files. Example: serverUrl=http://mediaX.simplex.tv/ - (please note that the last forward slash is required).	serverUrl=http://mediaX.simplex.tv/
videoUrl	This parameter needs only to be set, when video files have to be delivered over a caching server. All other files (.sid files, images etc.) will be taken from the main path.	videoUrl=http://cacheX.simplex.tv/ Here is a ticket with test links & information about this parameter: about videoUrl parameter
cID	Customer ID – Customer Account ID set by the Simplex Server. This ID is mandatory if other than the player in the project folder is called/embedded.	cID=333 (any valid customer ID)
aID	Author ID – Author Account ID set by the Simplex Server. Author Account needs to be a child of the defined customer ID. This ID is mandatory if other than the player in the project folder is called/embedded.	aID=390 (any valid author ID)
pID	Project ID – Project ID set by the Simplex Server. The defined projects needs to be a child of the defined author ID. In other words a video/webcast published by the defined author. This ID is mandatory if other than the player in the project folder is called/embedded. Example: video only project with cID, aID and pID: http://media10.simplex.tv/NubesPlayer/index.html?cID=2&aID=598&pID=26214	pID=48596 (any valid project ID)
channelIDs	Channel ID(s) – ID set by the Simplex Server for each channel. One or more channels (separated by comma) in the desired listing order. Example: channel player with cID and channelIDs. aID is optional and only required if a custom logo has been defined in the player layout. http://nubes.simplex.tv/NubesPlayer/index.html?cID=2&aID=8&channelIDs=390,391,392	channelIDs=390,391,392 (any valid channel ID)
embed	Enable or disable embed mode for the player.	true/false or 1/0
firstChannelID	By default the channel player is always loading first the content of the channel set at first position of your channelIDs list. Setting the ID of the channel as firstChannelID will have the player loading and displaying the content of the defined channel first.	firstChannelID=390 (any ID that was specified by channelIDs parameter)
firstProjectID	By default the channel player view is always loading and displaying the latest published video or webcast in the first channel set with channelIDs. Defining an existing project ID (video/webcast) will have the player to load this video first. Please be aware that the project ID defined as firstProjectID needs to be assigned at least to one of the defined channels.	firstProjectID=48596 (any project ID that is present in the any channel defined by channelIDs parameter)

autostart	Ondemand video: Automatically starts video playback after player is loaded. Autostart is by default set to false. Please note that mobile devices do not allow automated video playback.	true/false or 1/0
autostartLive	Live video: Automatically starts video playback after player is loaded. Autostart is by default set to true.	true/false or 1/0
position	Desired playback position in seconds. Set this parameter if the playback should start on specific position of the video timeline	integer (seconds), example: position=180
sharing	Displays or hides sharing button. The sharing options are only visible in the players complete view and not in embed mode. By default this option is set to true.	true/false or 1/0
rating	Displays or hides rating stars. The stars icon to rate a video are only visible in the players complete view and not in embed mode. By default this option is set to true.	true/false or 1/0
vpdownload	Displays or hides video podcast download button. The video podcast download is only visible in the players complete view and not in embed mode. By default this option is set to true. If video project has been published with Simplex Creator/Pro please get sure, the video podcast format is selected and the MP4 video published. If the MP4 video podcast file is not present in the project, the player will automatically not display the download button, even if this parameter is set to true.	true/false or 1/0
themeColor	Changes color of the following controls: progress bar, volume bar, subtitles menu & button, HD button, slides selection frames in Image Index panel.	color in hexadecimal format (without #), example: themeColor=E8E8E8
bgColor	Changes background color in the players complete view.	color in hexadecimal format (without #), example: bgColor=E8E8E8
language	All labels and messages used by the default player are available in English (en – default), German (de), French (fr) and Italian (it).	en/de/fr/it
hideControls	Displays or hides controls bar. Displays Big Pause Button after video start. By default it is set to false.	true/false or 1/0
loopVideo	For continuous play of the defined video in a loop, set this parameter to true. By default it is set to false.	true/false or 1/0
hideHD	To hide the toggle button to switch between HD and SD video quality, set this parameter to true. By default it is set to false.	true/false or 1/0
quality	By default the player is always loading and displaying the standard web quality, where HLS (HTTP Live Streaming) is not used. To have the high definition (HD) format loaded first, please set this parameter.	quality=40
hideBigPlayButton	To have the big play icon over the video hidden, set this parameter to true. By default it is set to false.	true/false or 1/0

subtitleLanguage (CC)	The player displays in the controls a CC button with the list of available subtitle languages the viewer can choose from. To have by default preselected language, set this parameter followed by the language code	subtitleLanguage=de
subtitlesAlign	To have the subtitle text aligned other than the default left, set center or right.	left/center/right
slideLang	Usually when two audio streams are streamed live as in the example above, it is required to display two sets of presentation slides in a webcast. Slides with the original Language and a second set with the translated text. In this case, the two presentations need to be merged (first set of slide with original slides followed by the translated slides) before importing them into the Simplex Creator/Pro. Please get sure the number of slides and order are the same for both languages. During the live event, only the original slides need to be pushed live. By setting the parameter slideLang=0 the original slides are displayed. While setting slideLang=1 the translated slides will be displayed. Here are the steps for projects configuring: Ondemand project (supported both Flash & HTML): ● Publish a project with video and slides (SIM14313588300_1 (2), (3) - numbers near SIM generated for each project ● Add slides (.swf & .png & .jpg) with another language via Manager to the project (they should have the same names ● but continue previous slides count (SIM14313588300_4 (5), (6)). Every slide should have its translation slide) ● Open the player with enabled slideLang parameter & you will see slides #4, 5, 6 instead of #1, 2, 3. Live project (supported by Flash only): ● Create a Live projects with slides ● If this parameter is enabled and you have 10 slides in one presentation then clicking on the first slide will ● automatically display 6th slide, second - 7th and so on. So the half of all slides is used for slides with translation.	true/false or 1/0
enableHLS	On iOS devices our HTML player is by default using HLS (HTTP Live Streaming – the videos are encoded in different qualities and segmented. During playback the delivery of the segments is adapting automatically to the bandwidth available). To use this technology as well for the Flash Player, you need to enable it by setting this parameter to true.	true/false or 1/0
firstIndex	On videos or webcast with chapter markers (cue points or slides) this parameter allows to load and playback the content, from a specific chapter While firstIndex=0 is the first chapter, the following chapters can be addressed by counting up from zero.	example: firstIndex=3
openIndex	In the complete view of the player, when displaying a video or webcast with chapters/attachments/chat/polls, the index panel is by default open. While when in embed mode, for reason of limited space, the index panel is by default closed. With this parameter panel will act to your requirements, when the video/webcast is loaded.	true/false or 1/0
googleAnalyticsID	Adding this GET parameter with your Google Analytics Account ID will have the player, track the viewers statistics to your account. In addition to the standard Google Analytics tracking like Location, viewers OS and Browser the player will send tracking information for the video watched, user interaction (play, pause, seeking) and view duration (25%,50%,75%,complete). Project: Test -> RealTime -> Overview	googleAnalyticsID=UA-XXXXXX-X

customerLogo	Tells the player to use customer logo instead of default one. Logo can be added to authors shared folder and will be displaying in the player if this parameter will be enabled.	name of the image with extension (default name: player_logo.jpg)
sendEvents	By enabling of this GET parameter, the player will send postMessages cross-domain to the origin page, that has the player embedded via iFrame. Work only together with simplexinteraction.js added to the players folder.	true/false or 1/0
hideHelp	In case a project has slides help button becomes visible in the controls bar. To hide it use this hideHelp parameter.	true/false or 1/0
useJS	This parameter is used to make a force start of HTML part of the player. So if this parameter enabled Flash part of the player will be ignored.	true/false
pip	Parameter to change start position of small video / slide window that appears in Embed and Fullscreen modes. Supports 4 value, one for every player's corner. Top-left position is used by default if no parameter is set or its value is invalid.	TL (top-left), TR (top-right), BL (bottom-left), BR (bottom-right)
LSInstID	Live Stream Instance ID. Used to define a Live stream which should be played. Takes value which was used in streaming tool to modify stream name.	LSInstID=1, detailed description can be found here
rightStationIDpath	Parameter sets the path to right station ID (image that is displayed over the video block in specified position). Supported file types: jpg, png, gif	rightStationID=http://name.com/image.png
leftStationIDpath	Parameter sets the path to left station ID (image that is displayed over the video block in specified position). Supported file types: jpg, png, gif	leftStationID=http://name.com/image.png
rightStationIDmarginX, rightStationIDmarginY	Parameters set position of the right station ID. Coordinates start position is at the left top corner of the player. Parameters use pixels value for positioning.	rightStationIDmarginX=50, rightStationIDmarginY=50
leftStationIDmarginX, leftStationIDmarginY	Parameters set position of the left station ID. Coordinates start position is at the left top corner of the player. Parameters use pixels value for positioning.	leftStationIDmarginX=50, leftStationIDmarginY=50
stateBgImage	Parameter allows to set background image for the player in PREVIEW, LIVE or POSTLIVE states (only for Embed mode). It can be scaled or takes central position of the video block.	stateBgImage=http://name.com/image.png
stateBgColor	Sets background color of the customized screen in PREVIEW, LIVE or POSTLIVE states (only for Embed mode). Uses HEX value without #.	stateBgColor=E8E8E8
stateFontColor	Sets font color of customized messages that can be defined for customized screen in PREVIEW, LIVE or POSTLIVE states (only for Embed mode). Uses HEX value without #.	stateFontColor=E8E8E8
stateFontSize	Sets font size of customized messages that can be defined for customized screen in PREVIEW, LIVE or POSTLIVE states (only for Embed mode). Parameters use value in pixels for positioning	stateFontSize=16
stateImageScaling	Allows to scale background image of customized screen. If stateImageScaling is set to true the stateBgImage will scale with the player. If stateImageScaling set to false the image will float horizontally and vertically centered over the stateBgColor.	true/false or 1/0

userName	This parameter is used to define user's name for project chat & preview screen. It is available only in case of selected authentication methods "Individual login key" & "Individual login key & password".	userName=name
profileID	Specifies profile ID to use for player.sid file loading which is required for channel view customization. In case parameter is not set, player uses customer's default Creator profile & its player.sid.	profileID=18 (any existed customer owned profile id)
enableSearch	Parameter is used to enable search UI displaying in channels mode. It contains search input, search button & clear button. By default these UI elements are disabled.	true/false or 1/0
channelRating	Allows to display or hide project rating stars in channel view. Disabled by default.	true/false or 1/0
enableRatingChange	Parameter allows to enable/disable activity of project rating start in a channel view. By default this functionality is disabled so a user can only see rating value of the project by can't vote.	true/false or 1/0
pagingAmount	Defines amount of visible pages of channel paging control. It is better to use odd numbers to make control's pages labels centered. Default pages amount is 9. Parameter influences on all channel views in case user passed several channels into the player.	pagingAmount=13
activeChannelPage	Sets active channel page that is opened on player start. In case a user sets active page that is exceed total amount of channel pages, player opens the last available page. In case user sets invalid value the first channel page wil be opened. Default page is the first channel page (activeChannelPage=0). Parameter influences only on the first channel opened after player start, next channel will be opened on the default first page.	activeChannelPage=3
channelSearch	Allows to set search string to filter channel projects on player start. Parameter influences only on the first channel opened after player start, next channel will be opened without search filtering.	channelSearch=text
videosPerPage	Defines amount of projects displayed per page in channel mode. Default amount is 20 (specified in player template).	videosPerPage=10
ratingStarsColor	Allows to set non-active rating star color. HEX colors without # symbol allowed only. In case the parameter is not defined player uses FFF (white) as default color. Parameter influences on both channel & single player modes.	ratingStarsColor=00FF00
ratingStarsActiveColor	Defines active (voted or hovered) rating stars color. HEX colors without # symbol allowed only. In case the parameter is not defined player tries to use themeColor parameter (ratingStarsActiveColor parameter has priority over themeColor parameter to allow to set different colors for stars & other controls). In case no themeColor parameter defined player will use default themeColor value (19AFFE). Parameter influences on both channel & single player modes.	ratingStarsActiveColor=00 00FF
subtitleLines	Allows to set custom amount of visible subtitle lines. Default amount is 3 lines.	subtitleLines=4
seekbarBackground	Sets custom background image for player seek bar. Uses absolute path to an image. Can be applied only for played seek bar area (data-pdk-seekbar-current element). In case parameter is not defined player uses default or custom themeColor value.	seekbarBackground=http //name.com/image.png

channelSortingType	Defines type of channel projects sorting. The following types are supported: Last published (ofdd), Past published (ofda), Last created (ocdd), Past created (ocda), Most viewed (omvd), Less viewed (omva), Last updated (oudd), Past updated (ouda), Project title descending (optd), Project title ascending (opta). Default sorting is Last published (ofdd).	channelSortingType=omvd
displayChannelSorting	Allows to show or hide channel projects sorting drop-down menu. By default the menu is hidden.	true/false or 1/0
displayMuteButton	Allows to toggle visibility of video mute button. By default mute button is visible.	true/false or 1/0
muteVideo	Allows to mute video on player start. All volume controls are influenced. By default video is unmuted on player start.	true/false or 1/0
useWebSockets	Allow to enable data transferring through Web Sockets connection.	true/false or 1/0
hideTime	Hides current video time & duration timers displayed on control bar. By default both timers are available on control bar for ondemand projects.	true/false or 1/0
showProjectViews	Enables project views (amount of users watched a project) displaying in channel mode. Hidden by default.	true/false or 1/0
timeBeforeAdSkip	Allows to set amount of time passed before ad skip control appearance. By default it is 10 seconds.	timeBeforeAdSkip=5
forceSockJsFallback	Allows to simulate disabling of WebSocket transport.	true/false or 1/0
useHive	Forces the Player to use Hive. False by default.	true/false or 0/1

API Methods

Simplex.create()

This method allows to create a new player app instance dynamically. It returns an instance of AngularJS $injector related to the newly created app. It requires a minimal configuration set to be passed in case there is no configuration specified in SimplexConfig object. There are several ways to call .create() method. Given that the following variables are defined

```
var node = document.getElementById('pdk-player');
var config = {
       cID: 100,
       aID: 200,
       pID: 300,
       serverUrl: https://server.simplex.tv/
};
var name = 'my-player-name';
```

Simplex.create() can be called following ways:

1) Creating a new player instance on a selected DOM node. In this case settings from SimplexConfig object will be used for configuration. player name will be assigned automatically,

```
var player = Simplex.create(node);
```

2) Creating a new player instance on a selected DOM node assigning a name, specified as a second argument.

```
var player = Simplex.create(node, name);
```

3) Creating a new player instance on a selected DOM node using second argument as config.

```
var player = Simplex.create(node, config);
```

4) Creating new player instance on a selected DOM node using both predefined app instance name and config.

```
var player = Simplex.create(node, name, config);
```

5) Creating a new player instance on an existing DOM node with specified name attribute (if it exists and is a part of DOM at the moment) with optional config object, passed as a second argument

```
var player = Simplex.create(name, config);
```

Simplex.destroy()

In case the player instance is no longer needed we should destroy it. This also can be done in several slightly different ways:

1) Destroys all player app instances, both dynamically and statically created. Additionally accepts optional boolean argument **restore**which allows to restore initial player DOM node state (with all its child nodes) from cache.

```
Simplex.destroy(restore);
```

2) Destroys player app instance with certain name. Additionally accepts optional boolean argument **restore** which allows to restore initial player DOM node state (with all its child nodes) from cache.

```
Simplex.destroy(name, restore);
```

3) Destroys player app using AngularJS application $injector instance, returned by Simplex.create() method. Additionally accepts optional boolean argument **restore** which allows to restore initial player DOM node state (with all its child nodes) from cache.

```
Simplex.destroy(player, restore);
```

Simplex["playerName"].changeProject(configObject)

```
Simplex.changeProject({
cID: 100,
aID: 200,
pID: 300,
serverUrl: https://server.simplex.tv/
});
```

Used to change current project. Can be used in channel mode to load selected project.

Simplex["playerName"].togglePlayback()

Used to change video playback state (pause to play & back)

Simplex["playerName"].play()

Used to start video playback

Simplex["playerName"].pause()

Used to stop video playback

PDK Events tracking

Simplex["playerName"].on(eventName, callback);

Used to subscribe to native video events

Simplex["playerName"].off(eventName);

Used to unsubscribe from attached video events

SimplexReady Event

```
document.addEventListener('SimplexReady', function() {
// Execute all related PDK code here
})
```

It is used to check if the PDK API is ready to use. Use the described event to start your code execution.

It will fix an issue with not defined Simplex object.

Players instantiation on arbitrary DOM nodes

To instantiate multiple independent player apps on arbitrary DOM nodes simply mark the DOM node or nodes with **data-pdk-player** attribute, like this:

```
<body data-pdk-player></body>
```

New player app instances will be automatically created on marked nodes.

data-pdk-player attribute also accepts a string value. If this value is passed, it would be used for namespacing SimplexConfig configuration object in order to make independent player app configuration possible. So, if you're going to instantiate several player apps with different configurations and use single SimplesConfig object to do that, you may do the following:

```
<div data-pdk-player="playerName"></div>
<div data-pdk-player="anotherName"></div>
```

Also there's one more way to pass config parameters to the player app directly from HTML code:

data-pdk-config attribute. Do the following:

```
<div data-pdk-player="playerName" data-pdk-config="autostart=true&rating=false"></div>
<div data-pdk-player="anotherName" data-pdk-config="autostartLive=false&muteVideo=true"></div>
```