Purple Documentation

Release 3.7

sprylab technologies GmbH

Jul 03, 2020

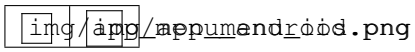
Contents:

CHAPTER 1

App Menu

The app menu is a global app UI component which is used for navigation. It consists of a header area, dynamic entries and a footer area. The configuration of this menu happens in the `app_menu.xml` file that is located in the dynamic resources.

Hint: All values must be properly escaped as required by the XML standard.



1.1 Sections

1.1.1 Navigation Header / Footer

The app menu can be configured to show custom logos at the top and bottom.

There can only be one `navigationHeader` and one `navigationFooter` element. Both are optional. These elements can contain `image` and `search` child elements.

The `image` element has a required `URL` and `height` attribute and can optionally also specify a background color and padding at each side.

The `search` element can be used to insert the search field in the menu. If the `navigationHeader` is used a `search` element must be added if the search should be shown. If it is missing no search will be visible.

Note: If the search is disabled through the app property adding a `search` element will have no effect - no search will be visible.

Example

Structure

```
<?xml version="1.0" encoding="UTF-8"?>
<app_menu>
    <navigationHeader>
        <image URL="top_logo.png"
               height="20.0"
               backgroundColor="#fefefe45"
               paddingLeft="10.0"
               paddingTop="20.0"
               paddingRight="30.0"
               paddingBottom="40.0" />
        <search />
    </navigationHeader>

    <navigation type="app_menu">
        <navigationNode targetURL="beliebige Url">
            <title>Title of this entry</title>
            <iconURL>Icon_for_this_entry.png</iconURL>
        </navigationNode>
    </navigation>

    <navigationFooter>
        <image URL="bottom_logo.png"
               height="20.0"
               backgroundColor="#fefefe45"
               paddingLeft="10.0"
               paddingTop="20.0"
               paddingRight="30.0"
               paddingBottom="40.0" />
    </navigationFooter>
</app_menu>
```

Search

The search field allows searching for issues in the kiosk. It requires internet access as the actual search is done by the Purple Manager. The field will only be visible if the app property `kiosk_search_enabled` is enabled.

Note: This feature is not available on the web platform.

1.1.2 Dynamic Entries

This part of the app menu can be freely customized using `navigationNode` nodes in the `app_menu.xml`. There are some special entries that are being added to the app menu even if they are not declared in the `app_menu.xml` such as the `current issue` or the `settings` entry.

Current Issue

This entry is only visible in single issue apps and navigates to the issue.

Settings

Opens a screen where the user can change app settings, e.g. if usage analysis is allowed or storage settings on Android.

This entry is only visible if there are settings available in the app, e.g. tracking or crash reporting or SD card support (only Android) is enabled, so that the user can opt-out.

If no `navigationNode` with the action url `purple://app/settings/open` is declared in the `app_menu.xml` but there are settings available in the app, then a default settings entry is automatically added at the end of the app menu

Configuration

Each `navigationNode` entry represents an item in the app menu.

The `targetURL` attribute describes the action that will be called when that entry is clicked. This can be either an action url or a web url.

The title of an entry can be set by adding a `title` node inside the `navigationNode`.

Note: Menu entries which are not supported on the web platform will be filtered out and not visible in the app menu in the web newsstand. A warning popup will inform the user about this in the preview version of the web newsstand.

Translations

As of release 3.11 it is now also possible to set translations in the app menu, so that there is no need to make specific folders. This can be done by adding multiple `title` nodes with `locale` attributes as shown in the following example. The resolution strategy is the same as the one for the localization folders in the dynamic resources. For further details on the resolution strategy see *Localization*.

Translated entries

```
<navigationNode targetURL="http://google.com">
    <title>default Title</title>
    <title locale="de">German Title</title>
    <title locale="en">English Title</title>
</navigationNode>
```

Custom-Icons

Each `navigationNode` can have an `activateIconURL` and an `iconURL` which define the icon that is shown next to the entry. These icons are then colored according to its state by the properties `app_menu_icon_active_color` and `app_menu_icon_normal_color`

Icon states

```
<navigationNode targetURL="purple://app/resource/dynamic/faq.html">
    <title>FAQ</title>
    <iconURL>faq_icon.png</iconURL>
```

```
    <activeIconURL>faq_icon_active.png</activeIconURL>
</navigationNode>
```

At last it is also possible to leave icons in different resolutions by adding `@2x` and `@3x` to the filename. The app then selects the best suited resolution at runtime.

Example: Icons and resolutions

Filename	Resolution
faq_icon.png	40 x 40 px
faq_icon@2x.png	80 x 80 px
faq_icon@3x.png	120 x 120 px

Role-Filters

Starting with version 2.6.0 app menu entries can be filtered by user roles. This is done by adding an `access` attribute to the `navigationNode`. A complete list of roles and access expressions can be found in the roles section. If the `access` attribute is not set, then it defaults to the `permitAll` expression.

Example Login and logout based on roles

This app menu will show the login entry only for logged out users and the logout entry only for logged in users.

The other entries are always visible.

```
<?xml version="1.0" encoding="UTF-8"?>
<app_menu>
    <navigation type="app_menu">
        <navigationNode targetURL="purple://kiosk/feed/open">
            <title>Newsfeed</title>
            <iconURL>newsfeed_icon.png</iconURL>
        </navigationNode>
        <navigationNode targetURL="purple://kiosk/open">
            <title>Kiosk</title>
            <iconURL>kiosk_icon.png</iconURL>
        </navigationNode>
        <navigationNode targetURL="purple://kiosk/entitlement/login/open" access=
↪"ROLE_ANONYMOUS">
            <title>Login</title>
            <iconURL>login_icon.png</iconURL>
        </navigationNode>
        <navigationNode targetURL="purple://kiosk/entitlement/logout/perform" access=
↪"AUTHENTICATED">
            <title>Logout</title>
            <iconURL>logout_icon.png</iconURL>
        </navigationNode>
    </navigation>
</app_menu>
```

Example complete app.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<app_menu>
    <navigationHeader>
        <image URL="logo.png" height="100.0" />
        <search />
    </navigationHeader>
    <navigation type="app_menu">
        <navigationNode targetURL="purple://kiosk/feed/open">
            <title>Newsfeed</title>
            <iconURL>menuicons/newsfeed.png</iconURL>
        </navigationNode>
        <navigationNode targetURL="purple://kiosk/open">
            <title>Newsstand</title>
            <iconURL>menuicons/newsstand.png</iconURL>
        </navigationNode>
        <navigationNode targetURL="purple://kiosk/subscriptions/open">
            <title>Subscriptions</title>
            <iconURL>menuicons/subscriptions.png</iconURL>
        </navigationNode>
        <navigationNode targetURL="purple://app/bookmarks/open">
            <title>Bookmarks</title>
            <iconURL>menuicons/bookmarks.png</iconURL>
        </navigationNode>
        <navigationNode targetURL="https://sprylab.com/home.html">
            <title>Website</title>
            <iconURL>menuicons/website.png</iconURL>
        </navigationNode>
        <navigationNode targetURL="purple://app/share_app_or_issue">
            <title>Share</title>
            <iconURL>menuicons/share.png</iconURL>
        </navigationNode>
        <navigationNode targetURL="purple://app/feedback/mail/open">
            <title>Feedback</title>
            <iconURL>menuicons/feedback.png</iconURL>
        </navigationNode>
        <navigationNode targetURL="purple://app/resource/dynamic/info/index.html">
            <title>Info/Contact</title>
            <iconURL>menuicons/legal.png</iconURL>
        </navigationNode>
        <navigationNode targetURL="purple://app/composer/connect/open">
            <title>Composer Connect</title>
            <iconURL>menuicons/chain.png</iconURL>
        </navigationNode>
    </navigation>
    <navigationFooter>
        <image URL="logo.png" height="100.0" />
    </navigationFooter>
</app_menu>
```

Colors

app_menu_background_color

Type: `Color`

This property defines the background color of the app menu.

app_menu_icon_active_color

Type: `Color`

This property defines the color of the icon of the currently selected app menu entry.

app_menu_icon_normal_color

Type: `Color`

This property defines the color of the icon of the app menu entries in its normal state.

app_menu_item_normal_background_color

Type: `Color`

This property defines the background color of an app menu entry.

app_menu_item_normal_text_color

Type: `Color`

This property defines the text color of an app menu entry.

app_menu_item_pressed_background_color

Type: `Color`

This property defines the background color of an app menu entry that is currently being pressed.

app_menu_item_pressed_text_color

Type: `Color`

This property defines the text color of an app menu entry that is currently being pressed.

app_menu_header_background_color

This is only available for Android and Kindle

Type: `Color`

This property defines the background color of the app menu header.

---

app_menu_item_separator_color

This is only available for iOS

Type: `Color`

This property defines the color of the separator between app menu entries.

CHAPTER 2

Deep Links

The deep links can be used inside the app and from outside.

For internal use the link starts with `purple:  //` and for external starts with `purple-<PACKAGE_NAME>://`.

The package name of the app can be found on the app overview page in the Purple DS Manager.

The `<PACKAGE_NAME>` in the deep link has to be used in lower case even though the app's actual package name can contain capital letters.

Example

Internal link `purple://app/info/open`

External link `purple-com.sprylab.purple.apps.developertest://app/info/open`

2.1 App

Open app information

New in version (Android): 2.7.0

New in version (iOS): 2.7.0

Opens the app information view.

URL

purple://app/info/open

Usable Contexts

Context	Usable
App menu	YES
Kiosk promotion area	YES
Storytelling content	YES
Purple webview	YES
Push notification Manager	NO
Push notifications Braze / Pinpoint	YES
In-App Messages Braze	YES

Open app settings

New in version (Android): 2.7.0

New in version (iOS): 3.10.0

Opens the app settings view. In this view it is possible to adjust the settings for the app, e.g. toggle tracking services.

Hint: On iOS this will open the device settings app.

URL

purple://app/settings/open

Usable Contexts

Context	Usable
App menu	YES
Kiosk promotion area	YES
Storytelling content	YES
Purple webview	YES
Push notification Manager	NO
Push notifications Braze / Pinpoint	YES
In-App Messages Braze	YES

Open app menu

New in version (Android): 2.3.3

New in version (iOS): 2.3.3

New in version (Web Kiosk): 3.10.0

Opens the app menu. The menu remains open, if it is already open.

URL

purple://app/menu/open

Usable Contexts

Context	Usable
App menu	YES
Kiosk promotion area	YES
Storytelling content	YES
Purple webview	YES
Push notification Manager	NO
Push notifications Braze / Pinpoint	NO
In-App Messages Braze	NO

Close app menu

New in version (Android): 2.3.3

New in version (iOS): 2.3.3

New in version (Web Kiosk): 3.10.0

Closes the app menu. The menu remains closed, if it is already closed.

URL

purple://app/menu/close

Usable Contexts

Context	Usable
App menu	YES
Kiosk promotion area	YES
Storytelling content	YES
Purple webview	YES
Push notification Manager	NO
Push notifications Braze / Pinpoint	NO
In-App Messages Braze	NO

Toggle app menu

New in version (Android): 2.3.3

New in version (iOS): 2.3.3

New in version (Web Kiosk): 3.10.0

Closes the app menu, if it was opened and opens the app menu if it was closed.

URL

purple://app/menu/toggle

Usable Contexts

Context	Usable
App menu	YES
Kiosk promotion area	YES
Storytelling content	YES
Purple webview	YES
Push notification Manager	NO
Push notifications Braze / Pinpoint	NO
In-App Messages Braze	NO

Open bookmarks view

New in version (Android): 2.7.0

New in version (iOS): 2.7.0

Opens the bookmarks view.

URL

purple://app/bookmarks/open

Usable Contexts

Context	Usable
App menu	YES
Kiosk promotion area	YES
Storytelling content	YES
Purple webview	YES
Push notification Manager	NO
Push notifications Braze / Pinpoint	YES
In-App Messages Braze	YES

Open feedback mail

New in version (Android): 2.7.0

New in version (iOS): 2.7.0

New in version (Web Kiosk): 3.5.0

Opens a pre-filled email for feedback as configured in the *dynamic resources*.

URL

purple://app/feedback/mail/open

Usable Contexts

Context	Usable
App menu	YES
Kiosk promotion area	YES
Storytelling content	YES
Purple webview	YES
Push notification Manager	NO
Push notifications Braze / Pinpoint	YES
In-App Messages Braze	YES

Show dynamic html content

New in version (Android): 2.1.0

New in version (iOS): 2.1.0

New in version (Web Kiosk): 3.10.0

Changed in version: 3.10.0 added :code:‘force_status_bar‘, 3.11.0 added :code:‘app_logo‘, 3.14.0: added :code:‘bounces‘ (iOS only), 5.1: added storefront variant

Opens an html file from *dynamic resources*. The app variant opens the html file on top of the current context while the storefront variant navigates to a kiosk context before opening the html.

URL

purple://app/resource/dynamic/<PATH>

purple://storefront/resource/dynamic/<PATH>

purple://app/resource/dynamic/<PATH>?display_mode= <VALUE> &title_bar= <VALUE> &controls= <VALUE> &force_status_bar= <VALUE> &app_logo= <VALUE>

purple://storefront/resource/dynamic/<PATH>?display_mode= <VALUE> &title_bar= <VALUE> &controls= <VALUE> &force_status_bar= <VALUE> &app_logo= <VALUE>

Parameter	Optional
PATH	NO

Query-Parameter	Optional	Values
display_mode	YES	• `embedded` app menu available (default) • `modal` no app menu available
title_bar	YES	• `true` show title bar (default) • `false` no title bar
controls	YES	• `true` show navigation controls • `false` no navigation controls (default)
force_status_bar	YES	• `true` show status bar • `false` no status bar (default)
app_logo	YES	• `true` show app logo in title bar instead of text • `false` no app logo (default)
bounces	YES	• `true` allow bouncing when scrolling (default) • `false` no bouncing Note: only available for iOS

Hint:

The `force_status_bar` parameter is ignored if the `title_bar` parameter is `true`. The title bar is always presented with a status bar.

The `app_logo` parameter is ignored if the `title_bar` parameter is `false`. The app logo can only be shown if a title bar is shown.

Usable Contexts

Context	Usable
App menu	YES
Kiosk promotion area	YES
Storytelling content	YES
Purple webview	YES
Push notification Manager	NO
Push notifications Braze / Pinpoint	YES
In-App Messages Braze	YES

Share app or issue

New in version (Android): 2.1.0

New in version (iOS): 2.1.0

Changed in version: 2.7.0 iOS Storytelling content is now supported.

Shares the current issue if one is open. Otherwise the app is shared.

URL

purple://app/share_app_or_issue

Usable Contexts

Context	Usable
App menu	YES
Kiosk promotion area	YES
Storytelling content	YES
Purple webview	YES
Push notification Manager	NO
Push notifications Braze / Pinpoint	YES
In-App Messages Braze	YES

Share app or issue or page

New in version (Android): 2.7.0

New in version (iOS): 2.7.0

Shares the current page if an issue is open and the property *Content Sharing* has been enabled in the Purple DS Manager. Otherwise the issue is shared if this is open.

Shares the app if the kiosk is open.

URL

purple://app/share_app_or_issue_or_page

Usable Contexts

Context	Usable
App menu	YES
Kiosk promotion area	YES
Storytelling content	YES
Purple webview	YES
Push notification Manager	NO
Push notifications Braze / Pinpoint	YES
In-App Messages Braze	YES

Open home

New in version (Android): 3.4.0

New in version (iOS): 3.4.0

New in version (Web Kiosk): 3.4.0

Opens the area of the app defined by the `app_initial_screen_url` property such as Kiosk, Channel or some Website in the dynamic resources. Due to Channel and Websites in the dynamic resources not being implemented in web newsstand, the newsstand will always be referred to as *home* in that context.

URL

purple://app/home/open

Usable Contexts

Context	Usable
App menu	YES
Kiosk promotion area	YES
Storytelling content	YES
Purple webview	YES
Push notification Manager	NO
Push notifications Braze / Pinpoint	YES
In-App Messages Braze	YES

Register push service

New in version (iOS): 3.8.0

> **Warning:** This action url is not supported on Android.

Triggers the push registration for the application. This will display a system dialog asking for permission to receive push notifications.

URL

purple://app/push/register

Usable Contexts

Context	Usable
App menu	YES
Kiosk promotion area	YES
Storytelling content	YES
Purple webview	YES
Push notification Manager	NO
Push notifications Braze / Pinpoint	YES
In-App Messages Braze	YES

Open external URL

New in version (Android): 3.9.0

New in version (iOS): 3.9.0

New in version (Web Kiosk): 3.9.0

Open the given URL by handing it over to the underlying OS. The URL has to be URL-encoded.

URL

purple://app/open/external/url/`URLEncodedURL`

Usable Contexts

Context	Usable
App menu	YES
Kiosk promotion area	YES
Storytelling content	YES
Purple webview	YES
Push notification Manager	NO
Push notifications Braze / Pinpoint	YES
In-App Messages Braze	YES

Open HTML onboarding

New in version (Android): 3.10.1

New in version (iOS): 3.10.1

Open HTML onboarding screen which is shown on first app start. This only works if HTML onboarding is enabled.

URL

purple://app/onboarding/app_start/open

Usable Contexts

Context	Usable
App menu	YES
Kiosk promotion area	YES
Storytelling content	YES
Purple webview	YES
Push notification Manager	NO
Push notifications Braze / Pinpoint	YES
In-App Messages Braze	YES

2.2 Kiosk

Open kiosk (newsstand)

New in version (Android): 2.1.0

New in version (iOS): 2.1.0

New in version (Web Kiosk): 3.2.0

Opens the kiosk.

URL

purple://kiosk/open

Usable Contexts

Context	Usable
App menu	YES
Kiosk promotion area	YES
Storytelling content	YES
Purple webview	YES
Push notification Manager	NO
Push notifications Braze / Pinpoint	YES
In-App Messages Braze	YES

Open channel (newsfeed)

New in version (Android): 2.2.0

New in version (iOS): 2.1.0

Opens the channel.

URL

purple://kiosk/feed/open

Usable Contexts

Context	Usable
App menu	YES
Kiosk promotion area	YES
Storytelling content	YES
Purple webview	YES
Push notification Manager	NO
Push notifications Braze / Pinpoint	YES
In-App Messages Braze	YES

Open category chooser

New in version (Android): 2.7.0

New in version (iOS): 2.3.3

New in version (Web Kiosk): 3.2.0

Opens the category chooser view.

URL

purple://kiosk/category/chooser/open

Usable Contexts

Context	Usable
App menu	YES
Kiosk promotion area	YES
Storytelling content	YES
Purple webview	YES
Push notification Manager	NO
Push notifications Braze / Pinpoint	YES
In-App Messages Braze	YES

Show issue preview

New in version (Android): 2.7.0

New in version (iOS): 2.7.0

Opens the preview view of an issue in the kiosk via its ID or its alias.

URL

purple://kiosk/issue/<ISSUE_ID>/preview

purple://kiosk/issue/alias/<ISSUE_ALIAS>/preview

Parameter	Optional
ISSUE_ID	NO
ISSUE_ALIAS	NO

Usable Contexts

Context	Usable
App menu	YES
Kiosk promotion area	YES
Storytelling content	YES
Purple webview	YES
Push notification Manager	NO
Push notifications Braze / Pinpoint	YES
In-App Messages Braze	YES

Entitlement login

New in version (Android): 2.6.0

New in version (iOS): 2.6.0

New in version (Web Kiosk): 3.2.0

Opens the entitlement login view.

URL

purple://kiosk/entitlement/login/open

Usable Contexts

Context	Usable
App menu	YES
Kiosk promotion area	YES
Storytelling content	YES
Purple webview	YES
Push notification Manager	NO
Push notifications Braze / Pinpoint	YES
In-App Messages Braze	YES

Perform entitlement login

New in version (Android): 2.6.0

New in version (iOS): 2.6.0

New in version (Web Kiosk): 3.2.0

Performs the entitlement login using the provided **login_name**, **access_token** and **roles**. If the login was performed for an OAuth entitlement server, the **refresh_token** is used for refreshing the **access_token**. If the login was successful, the **url_encoded_action_url** will be opened.

URL

purple://kiosk/entitlement/login/perform?login_name= `<LOGIN_NAME>` &token= `<ACCESS_TOKEN>` &roles= `<ROLES>` &refresh_token= `<REFRESH_TOKEN>` &success_url= `<URL_ENCODED_ACTION_URL>`

Query-Parameter	Optional
login_name	YES
token	NO
roles	NO
refresh_token	YES
success_url	YES

Usable Contexts

Context	Usable
App menu	YES
Kiosk promotion area	YES
Storytelling content	YES
Purple webview	YES
Push notification Manager	NO
Push notifications Braze / Pinpoint	YES
In-App Messages Braze	YES

Perform entitlement logout

New in version (Android): 2.6.0

New in version (iOS): 2.6.0

New in version (Web Kiosk): 3.2.0

Performs the entitlement logout. If the logout was successful, the **url_encoded_action_url** will be opened.

URL

purple://kiosk/entitlement/logout/perform?success_url= `<URL_ENCODED_ACTION_URL>`

Query-Parameter	Optional
success_url	YES

Usable Contexts

Context	Usable
App menu	YES
Kiosk promotion area	YES
Storytelling content	YES
Purple webview	YES
Push notification Manager	NO
Push notifications Braze / Pinpoint	YES
In-App Messages Braze	YES

Start OAuth login flow

New in version (Android): 3.11.0

New in version (iOS): 3.11.0

Starts the entitlement OAuth login flow. If the login flow was successful completed, the **url_encoded_action_url** will be opened.

URL

purple://kiosk/entitlement/login/oauth/start?success_url= `<URL_ENCODED_ACTION_URL>`

Query-Parameter	Optional
success_url	YES

Usable Contexts

Context	Usable
App menu	YES
Kiosk promotion area	YES
Storytelling content	YES
Purple webview	YES
Push notification Manager	NO
Push notifications Braze / Pinpoint	YES
In-App Messages Braze	YES

Cancel OAuth login flow

New in version (Android): 3.11.0

New in version (iOS): 3.11.0

Cancels the entitlement OAuth login flow.

URL

purple://kiosk/entitlement/login/oauth/cancel

Usable Contexts

Context	Usable
App menu	YES
Kiosk promotion area	YES
Storytelling content	YES
Purple webview	YES
Push notification Manager	NO
Push notifications Braze / Pinpoint	YES
In-App Messages Braze	YES

Open subscription administration view

New in version (Android): 2.7.0

New in version (iOS): 2.7.0

Opens the subscription administration view with all activated subscription variants and depending on configuration including entitlements.

URL

purple://kiosk/subscriptions/open

Usable Contexts

Context	Usable
App menu	YES
Kiosk promotion area	YES
Storytelling content	YES
Purple webview	YES
Push notification Manager	NO
Push notifications Braze / Pinpoint	YES
In-App Messages Braze	YES

Start In-App purchase

New in version (Android): 2.7.0

New in version (iOS): 2.7.0

Starts an In-App purchase of a product (issue/subscription) via its **product_id** property.

URL

purple://kiosk/products/<PRODUCT_ID>/purchase

Parameter	Optional
PRODUCT_ID	NO

Usable Contexts

Context	Usable
App menu	YES
Kiosk promotion area	YES
Storytelling content	NO Android, YES ios
Purple webview	YES
Push notification Manager	NO
Push notifications Braze / Pinpoint	YES
In-App Messages Braze	YES

Restore purchases

New in version (iOS): 2.7.0

> **Warning:** This action url is not supported on Android.

Restores all the purchases of the user.

URL

purple://kiosk/products/restore

Usable Contexts

Context	Usable
App menu	YES
Kiosk promotion area	YES
Storytelling content	YES
Purple webview	YES
Push notification Manager	NO
Push notifications Braze / Pinpoint	YES
In-App Messages Braze	YES

Open issue

New in version (Android): 2.7.0

New in version (iOS): 2.7.0

New in version (Web Kiosk): 3.2.0

New in version (Web Player): 3.0.0

Changed in version: 3.2.0 Add alias, add current page, 3.11.0 added open by index

Opens an issue via its **issue_id**. If the action url is called with the **page_id** the page of the issue will be opened. It is also possible to provide the **issue_alias**, **page_alias** and **page_index** instead of their respective ids.

URL

purple://kiosk/issue/<ISSUE>/open

<ISSUE>	Expected result
<ISSUE_ID>	Opens the issue in the soft-bookmark state.
alias/<ISSUE_ALIAS>	
current (or issue ID or alias of the current issue)	Remains in the current state.

URL

purple://kiosk/issue/<ISSUE>/page/<PAGE>/open

<ISSUE>	<PAGE>	Expected result
<ISSUE_ID>	<PAGE_ID>	Opens an issue via its issue ID on the specified page (page ID, page alias or page index).
	alias/<PAGE_ALIAS>	
	index/<PAGE_INDEX>	
alias/<ISSUE_ALIAS>	<PAGE_ID>	Opens an issue via its alias on the specified page (page ID, page alias or page index).
	alias/<PAGE_ALIAS>	
	index/<PAGE_INDEX>	
current (or issue ID or alias of the current issue)	<PAGE_ID>	Opens the specified page of the current issue.
	alias/<PAGE_ALIAS>	
	index/<PAGE_INDEX>	
<ISSUE_ID>	current	Opens the issue on the first page.
alias/<ISSUE_ALIAS>		
current (or issue ID or alias of the current issue)	current (or page ID, alias or index of the current issue)	Remains in the current state.

Jump to element

This is a special case of *open an issue*. It jumps to an issue, a page or an element. To jump to an element, the **element_id** or the **element_alias** must be specified as the fragment component of the url.

The alignment of the element on the screen can be set with the parameter **align** by stating one of the following alignments: `top_left`, `top_center`, `top_right`, `center_left`, `center_center`, `center_right`, `bottom_left`, `bottom_center`, `bottom_right`

The alignment is a optional query parameter.

URL

purple://kiosk/issue/`<ISSUE>`/open# `<ELEMENT>`

purple://kiosk/issue/`<ISSUE>`/open?align= `<ALIGNMENT>` # `<ELEMENT>`

<ISSUE>	<ELEMENT>	Expected result
`<ISSUE_ID>`	`<ELEMENT_ID>`	Opens the issue on the first page and jumps to the specified element if possible.
alias/`<ISSUE_ALIAS>`	`<ELEMENT_ALIAS>`	
current (or issue ID or alias of the current issue)	`<ELEMENT_ID>`	Remains in the current state.
	`<ELEMENT_ALIAS>`	

URL

purple://kiosk/issue/`<ISSUE>`/page/`<PAGE>`/open# `<ELEMENT>`

purple://kiosk/issue/`<ISSUE>`/page/`<PAGE>`/open?align= `<ALIGNMENT>` # `<ELEMENT>`

<ISSUE>	<PAGE>	<ELEMENT>	Expected result
`<ISSUE_ID>`	`<PAGE_ID>`	`<ELEMENT_ID>`	Opens the issue on the specified page and jumps to the specified element if possible.
alias/`<ISSUE_ALIAS>`	alias/`<PAGE_ALIAS>`	`<ELEMENT_ALIAS>`	
current (or issue ID or alias of the current issue)	`<PAGE_ID>`	`<ELEMENT_ID>`	Opens the specified page of the current issue and jumps to the specified element if possible.
	alias/`<PAGE_ALIAS>`	`<ELEMENT_ALIAS>`	
`<ISSUE_ID>`	current	`<ELEMENT_ID>`	Opens the issue on the first page and jumps to the specified element if possible.
alias/`<ISSUE_ALIAS>`		`<ELEMENT_ALIAS>`	
current (or issue ID or alias of the current issue)	current (or page ID or alias of the current issue)	`<ELEMENT_ID>`	Jumps to the specified element on the current page of the current issue.
		`<ELEMENT_ALIAS>`	

Usable Contexts

Context	Usable
App menu	YES
Kiosk promotion area	YES
Storytelling content	YES
Purple webview	YES
Push notification Manager	NO
Push notifications Braze / Pinpoint	YES
In-App Messages Braze	YES

Open issue via its external identifier

New in version (Android): 2.5.0

New in version (iOS): 2.5.0

New in version (Web Kiosk): 3.2.0

New in version (Web Player): 3.0.0

It is possible to open an issue via its **external_issue_id** property. If the issue cannot be found in the current local kiosk, the optional **fallback_url** can be used to provide an alternative web link for the issue, e.g. the website of an article in the CMS.

URL

purple://kiosk/issue/by_external_id/<EXTERNAL_ISSUE_ID>/open?fallback_url= <URL_ENCODED_URL> &target= <TARGET>

Parameter	Optional
EXTERNAL_ISSUE_ID	NO

Query Parameter	Optional	Description
fallback_url	YES	The url which should be opened if no issue with the external issue ID can be found.
target	YES	The target for the fallback_url. Can be "_blank" for external window, or empty / omitted for inapp browser.

Usable Contexts

Context	Usable
App menu	YES
Kiosk promotion area	YES
Storytelling content	YES
Purple webview	YES
Push notification Manager	NO
Push notifications Braze / Pinpoint	YES
In-App Messages Braze	YES

Open publication

New in version (Android): 3.4.0

New in version (iOS): 3.4.0

New in version (Web Kiosk): 3.4.0

Opens the given publication. A kiosk-publication will be opened in the kiosk and a channel-publication in the channel view. If the target view is disabled, e.g. channel disabled, or if there is no publication with the given **publication_id** then no navigation will be performed.

URL

purple://kiosk/publication/<PUBLICATION_ID>/open

Usable Contexts

Context	Usable
App menu	YES
Kiosk promotion area	YES
Storytelling content	YES
Purple webview	YES
Push notification Manager	NO
Push notifications Braze / Pinpoint	YES
In-App Messages Braze	YES

2.3 Presenter

Navigate between articles

New in version (Android): 3.15.0

New in version (iOS): 3.15.0

Changed in version: 5.2.0 now accepts an issue ID instead of the :code:‘previous‘ and :code:‘next‘ keywords.

Navigates to the given issue within the current article pager. Instead of using an issue ID it is possible to use `previous` and `next` to navigate to the previous or next issue.

URL

purple://presenter/issue/`<ISSUE_ID>|previous|next`

Note: These deep links only work within an article pager that was opened via the JavaScript-API. And only articles within this pager can be accessed.

Usable Contexts

Context	Usable
App menu	NO
Kiosk promotion area	NO
Storytelling content	YES
Purple webview	NO
Push notification Manager	NO
Push notifications Braze / Pinpoint	NO
In-App Messages Braze	NO

2.4 Content

Open table of contents

New in version (Android): 2.7.0

New in version (iOS): 2.7.0

New in version (Web Player): 3.0.0

Opens the table of contents of an issue.

URL

purple://content/toc/open

Usable Contexts

Context	Usable
App menu	NO
Kiosk promotion area	NO
Storytelling content	YES
Purple webview	NO
Push notification Manager	NO
Push notifications Braze / Pinpoint	NO
In-App Messages Braze	NO

Navigate inside an issue via alias

New in version (Android): 2.7.0

New in version (iOS): 2.7.0

New in version (Web Player): 3.0.0

Navigates inside an issue to the corresponding page via **alias**.

URL

purple://content/page/alias/<ALIAS>/open

Parameter	Optional
ALIAS	NO

Usable Contexts

Context	Usable
App menu	NO
Kiosk promotion area	NO
Storytelling content	YES
Purple webview	NO
Push notification Manager	NO
Push notifications Braze / Pinpoint	NO
In-App Messages Braze	NO

Navigate inside an issue via index

New in version (Android): 2.7.0

New in version (iOS): 2.7.0

New in version (Web Player): 3.0.0

Navigates inside an issue to the corresponding page via **index**.

URL

purple://content/page/index/<INDEX>/open

Parameter	Optional
INDEX	NO

Usable Contexts

Context	Usable
App menu	NO
Kiosk promotion area	NO
Storytelling content	YES
Purple webview	NO
Push notification Manager	NO
Push notifications Braze / Pinpoint	NO
In-App Messages Braze	NO

Navigate inside an issue via page ID

New in version (Android): 3.11.0

New in version (iOS): 3.11.0

New in version (Web Player): 3.11.0

Navigates inside an issue to the corresponding page via its **ID**.

URL

purple://content/page/<PAGE_ID>/open

Parameter	Optional
PAGE_ID	NO

Usable Contexts

Context	Usable
App menu	NO
Kiosk promotion area	NO
Storytelling content	YES
Purple webview	NO
Push notification Manager	NO
Push notifications Braze / Pinpoint	NO
In-App Messages Braze	NO

Share content page

New in version (Android): 2.7.0

New in version (iOS): 2.7.0

Shares a content page of an issue via its **alias**.

URL

purple://content/page/alias/<ALIAS>/share

Parameter	Optional
ALIAS	NO

Usable Contexts

Context	Usable
App menu	NO
Kiosk promotion area	NO
Storytelling content	YES
Purple webview	NO
Push notification Manager	NO
Push notifications Braze / Pinpoint	NO
In-App Messages Braze	NO

Add bookmark

This is not available for web

New in version (Android): 3.3.0

New in version (iOS): 3.3.0

Adds a bookmark of the current state of the storytelling content. If there is already a bookmark for the exact same state then no new bookmark will added.

URL

purple://content/bookmark/add

Usable Contexts

Context	Usable
App menu	NO
Kiosk promotion area	NO
Storytelling content	YES
Purple webview	NO
Push notification Manager	NO
Push notifications Braze / Pinpoint	NO
In-App Messages Braze	NO

2.5 Old action urls

Title	Action url	Version Android	Version iOS	Version Web
Open app information	pkapp://action/openAppInformation	• 1.6.0 • 2.1.0 (Braze)	2.1.0	not supported
Open app settings	pkapp://action/openSettings	2.1.0	n/a	not supported
Open bookmarks view	pkapp://action/openBookmarks	2.1.0	2.1.0	not supported
Open composer connect	pkapp://action/openComposerConnect	2.1.0	2.1.0	not supported
Open manager connect	pkapp://action/openPurpleManagerConnect	2.1.0	n/a	not supported
Open feedback mail	pkapp://action/openFeedback	2.1.0	1.7.x	not supported
Show dynamic html content	pkapp://action/showDynamicContent/<PATH>	2.7.0	1.7.x	not supported
Share app or issue or page	purple://app/share_app_or_issue_or_content	2.2.0	2.7.0	not supported
Open kiosk (newsstand)	pkapp://action/openKiosk	• 1.6.x • 2.1.0 (Braze)	• 1.x • 2.1.0 (Braze)	not supported
Open channel (newsfeed)	pkapp://action/openChannelFeed	2.2.0	2.1.0	not supported
Open category chooser	pkapp://action/changeKioskCategory	2.1.0	2.3.3	not supported
Show issue preview	pkapp://action/openIssueDetailByID/<ISSUE_ID>	• 1.6.x • 2.1.0 (Braze)	2.7.0	not supported
Open subscription administration view	pkapp://action/openSubscriptions	• 1.6.x • 2.1.0 (Braze)	• 1.x • 2.1.0 (Braze)	not supported
Start In-App purchase	pkapp://action/purchase/<PRODUCT_ID>	2.1.0	2.1.0	not supported
Restore purchases	pkapp://action/restorePurchases	not supported	1.x	not supported
Open issue	pkapp://action/openIssueByID/<ISSUE_ID>	• 1.6.x • 2.1.0 (Braze)	• 2.7.0 • 2.1.0 (Braze)	not supported
(Open current issue)	pkapp://action/presentCurrentIssue	2.1.0	2.1.0	not supported
(Jump to element)	pkitem://purple/<ISSUE_ID>/<PAGE_ID>?align=<ALIGNMENT> #<ELEMENT_ID>	2.1.0	2.7.0	2.0.0 (inside current issue only)
Open table of contents	pkapp://action/openTOC	1.6.x	1.x	not supported
Navigate inside an issue via alias	pkapp://navigate/alias/<ALIAS>	1.6.x	2.7.0	not supported
Navigate inside an issue via index	pkapp://navigate/index/<INDEX>	1.6.x	2.7.0	not supported
Share content page	pkapp://action/shareContentPage/<ALIAS> pkapp://action/shareContentPage/.	2.2.0	2.1.0	not supported

2.6 Web Player specifics

External links do not work out of the box in Web Player because Safari, Edge, and Internet Explorer do not support the needed function (http://caniuse.com/#feat=registerprotocolhandler).

Due to WebViews being implemented using iframes in Web Player, Action URLs from inside Storytelling WebViews only work if `purpleInterface.js` is included in the embedded page. When the WebView is loaded, all links with Purple Action URLs as href will be given ActionHandlers, so they can be handled by Web Player.

From V 3.0.0 `purpleInterface.js` is included in the Web Player repository. The latest version is delivered via Purple DS | Web Newsstand. It is recommended to include the script from one of the following URLs to assure to always use the latest version: https://kiosk.purplemanager.com/scripts/purpleInterface.js https://kiosk.purplemanager.com/scripts/purpleInterface.min.js

Note: Please be aware that only sites can be displayed which have the `X-FRAME-OPTIONS` header set correctly. Read here for details: https://developer.mozilla.org/en/docs/Web/HTTP/Headers/X-Frame-Options

purpleInterface.js (excerpt)

```
window.purpleInterface = {
callbacks: {},
util: {
    postMessage: function (type, key, value, callback) {

        if (window !== window.parent) {

            // create requestData
            var requestData = {
                type: type,
                key: key
            };
            if (value) {
                requestData.value = value;
            }

            // call postMessage
            window.parent.postMessage(JSON.stringify(requestData), '*');
        }

    }
  }
};

document.addEventListener('DOMContentLoaded', function () {
    window.addEventListener('message', window.purpleInterface.util.receiveMessage);

    window.purpleInterface.util.postMessage('LOAD', 'LOAD', null, function () {
```

```
        if (!window.purple) {
            var links = document.querySelectorAll('a[href^="purple://"], a[ ^="pkapp:/
↪/"], a[href^="pkitem://"]');
            for (var i = 0; i < links.length; i++) {
                links[i].addEventListener('click', function (e) {
                    window.purpleInterface.util.postMessage('ACTION_URL', 1, this.
↪href);
                    e.preventDefault();
                });
            }
        }

    });
});
```

2.7 Standard Protocols

Mailto

New in version (Android): 2.1.0

New in version (iOS): 2.1.0

New in version (Web Player): 3.1.0

Starts the default email client for sending an email.

URL

mailto:<EMAIL_ADDRESS>

mailto:?to= <EMAIL_ADDRESS> &bcc= <EMAIL_ADDRESS> &subject= <SUBJECT> &body= <BODY>

Usable Contexts

Context	Usable
App menu	YES
Kiosk promotion area	YES
Storytelling content	YES
Purple webview	YES
Push notification Manager	NO
Push notifications Braze / Pinpoint	YES
In-App Messages Braze	YES

Tel

New in version (Android): 3.2.0

New in version (iOS): 3.2.0

New in version (Web Player): 3.1.0

Starts the default phone app for a call.

URL

tel:<PHONE_NUMBER>

Usable Contexts

Context	Usable
App menu	YES
Kiosk promotion area	YES
Storytelling content	YES
Purple webview	YES
Push notification Manager	NO
Push notifications Braze / Pinpoint	YES
In-App Messages Braze	YES

CHAPTER 3

Dynamic Resources

3.1 Overview

The dynamic resources are some files that can change some configuration (e.g. *App Menu*, *Channels*, *Tracking*, ...) without submitting an app update. The app checks for updated dynamic resources on every app start and resume.

3.2 Structure

The minimal setup of the dynamic resources requires a `default` folder which contains the configuration files.

Example

```
default/
├── app_menu.xml
├── channel_configs.json
├── sharing.properties
└── ...
```

3.2.1 Localization

It is possible to add translations by adding folders such as `de`, `en` or `de_DE`, `en_US` next to the `default` folder. Depending on the device's preferred languages the app loads the best fitting configuration files from these folders. Starting with Android 7.0 the system supports multiple preferred languages just like iOS. On earlier Android versions only the device's system language is used. The look-up happens in the following way:

For each of the system's preferred language check if there is a folder that

1. matches the exact locale (e.g. `de_DE` or `en_US`)
2. matches the exact language (e.g. use `en` folder for `en_US`)

3. uses the same language but different region (e.g. use `en_UK` folder for `en_US`)

In the case that there is no matching folder, the configuration falls back to the contents of the `default` folder.

Example: Resource resolution

Dynamic resources:

```
/
├── default/
├── de_DE/
└── it_IT/
```

Preferred languages:

1. fr_CH
2. it_CH

Resolution:

1. check exact locale (fr_CH) -> no match
2. check exact language (fr) -> no match
3. check same language but different region (fr) -> no match
4. check exact locale (it_CH) -> no match
5. check exact language (it) -> no match
6. check same language but different region (it) -> it_IT

Example: Different sharing settings

```
/
├── default/
│   ├── app_menu.xml
│   ├── channel_configs.json
│   ├── sharing.properties
│   └── ...
├── en/
│   └── sharing.properties
└── de/
    └── sharing.properties
```

3.2.2 Platform-specific Configuration

Additionally to translations it is also possible to add platform-specific configurations. This is done by adding some platform folders such as `android`, `ios`, `kindle` or `web`. The structure of these folders is the same as mentioned above.

Example: Platform-specific structure

```
/
├── android/
│   ├── default/
│   │   └── app_menu.xml
│   ├── en/
│   │   └── app_menu.xml
│   └── de/
│       └── app_menu.xml
├── ios/
│   ├── default/
│   │   └── app_menu.xml
│   ├── en/
│   │   └── app_menu.xml
│   └── de/
│       └── app_menu.xml
├── web/
│   ├── default/
│   │   └── app_menu.xml
│   ├── en/
│   │   └── app_menu.xml
│   └── de/
│       └── app_menu.xml
├── en/
│   ├── channel_configs.json
│   └── sharing.properties
├── de/
│   ├── channel_configs.json
│   └── sharing.properties
└── default/
    ├── channel_configs.json
    └── sharing.properties
```

Result on Android

```
/
├── default/
│   ├── app_menu.xml (android-default)
│   ├── channel_configs.json (default)
│   └── sharing.properties (default)
├── en/
│   ├── app_menu.xml (android-en)
│   ├── channel_configs.json (default)
│   └── sharing.properties (default)
└── de/
   ├── app_menu.xml (android-de)
   ├── channel_configs.json (default)
   └── sharing.properties (default)
```

Result on iOS

```
/
├── default/
│   ├── app_menu.xml (ios-default)
│   ├── channel_configs.json (default)
│   └── sharing.properties (default)
├── en/
│   ├── app_menu.xml (ios-en)
│   ├── channel_configs.json (default)
│   └── sharing.properties (default)
└── de/
    ├── app_menu.xml (ios-de)
    ├── channel_configs.json (default)
    └── sharing.properties (default)
```

Result on web

```
/
├── default/
│   ├── app_menu.xml (web-default)
│   ├── channel_configs.json (default)
│   └── sharing.properties (default)
├── en/
│   ├── app_menu.xml (web-en)
│   ├── channel_configs.json (default)
│   └── sharing.properties (default)
└── de/
    ├── app_menu.xml (web-de)
    ├── channel_configs.json (default)
    └── sharing.properties (default)
```

If two folders contain the same file, then the more specific file overwrites the more generic one. E.g. some file in `default` gets overwritten by the same file from `android`.

3.3 Configuration

3.3.1 App Menu

The app's side menu can be configured using the `app_menu.xml` file. For a detailed explanation of all its features and configuration see App Menu.

3.3.2 Dynamic Configuration

Starting with PK 5.2.0 it is possible to place a `config.json` in the dynamic resources. This is a simple key-value file which can be used by the app for some dynamic configuration.

Currently only the `onboarding_version` key is used by the app to determine the version of the current onboarding.

3.3.3 Channels

Configuration

The channel view can be set up using the `channel_configs.json`. This file consists of multiple configurations for device types and orientations.

Structure

```
{
    "phone": {
        "portrait": {
            ...
        },
        "landscape": {
            ...
        }
    },
    "tablet": {
        "portrait": {
            ...
        },
        "landscape": {
            ...
        }
    }
}
```

Inside those nodes is where the actual configuration lies. The first part of that node contains the general configuration, such as the ideal teaser size and the visibility of some page indicators.

Example: General configuration

```
"multiColumn": false,
"teaserWidth": 256,
"teaserHeight": 144,
"pageArrowsEnabled": false,
"pageIndicatorsEnabled": true,
"pageIndicatorAlignment": "right",
```

Name	Type	Values	Description
multiColumn	boolean	true, false	defines whether there are multiple columns of teasers
teaserWidth	int		defines the ideal tile width in dp
teaserHeight	int		defines the ideal tile height in dp
pageArrowsEnabled	boolean	true, false	toggles the pages arrows on the side of the screen
pageIndicatorsEnabled	boolean	true, false	toggles the page indicator dots on the top of the screen
pageIndicatorAlignment	String	left, center, right	sets the position of the page indicator dots

The teaserWidth and teaserHeight define the layout of the channel. These values are in density-independent pixels (dp). A list of some common devices and their screen sizes in dp can be found here. If `multiColumn` is `true`, then the layout process tries to fit as many teasers next to each other as possible. While doing so, the teasers are never scaled down, so e.g. on a display with 1024 dp in the width only 4 tiles with a width of 250 dp can be displayed. Once the amount of column is calculated, the tiles are stretched equally to fill the screen while maintaining the aspect ratio defined by the teaserWidth and teaserHeight.

The next part of the configuration defines the teaser types. Currently there are only two types: `topTeaser` which is the first teaser in a channel and `teaser` for all the remaining ones.

Example: Teaser types

```
"types": {
    "topTeaser": {
        "thumbnailKind" : "phone_portrait_top",
        "factorY": 1,
        "spanX": 1,
        "spanY": 1,
        "gradient": {
            "height": 0.6,
            "alphaStart": 1.0,
            "alphaEnd": 0.0
        },
        "title": {
            "fontSize": 20,
            "font": "Roboto-Condensed",
            "maxLines": 3
        },
        "headline": {
            "fontSize": 12,
            "font": "Roboto-Condensed",
            "maxLines": 1
        }
    },
    "teaser": {
        "thumbnailKind" : "phone_portrait_normal",
        "gradient": {
            "height": 0.6,
            "alphaStart": 1.0,
            "alphaEnd": 0.0
        },
        "title": {
            "fontSize": 20,
            "font": "Roboto-Condensed",
            "maxLines": 3
        },
        "headline": {
            "fontSize": 12,
            "font": "Roboto-Condensed",
            "maxLines": 1
        }
    }
}
```

The configuration for both teaser types is mostly the same. The `topTeaser` has three additional attributes. The `factorY` is only evaluated when `multiColumn` is `false`. In this case the first teaser's height is multiplied by this value. For the case that `multiColumn` is `true`, then the top teaser's width is multiplied by the value of `spanX` and its height is multiplied by `spanY`.

Name	Type	Description
factorY	float	(multiColumn=false) multiplier for the height of the top teaser
spanX	int	(multiColumn=true) amount of columns that the top teaser will take
spanY	int	(multiColumn=true) amount of rows that the top teaser will take

The rest of the configuration consists of a `thumbnailKind` which defines the teaser image that is loaded from the Purple Manager, the `gradient` which configures the gradient behind the text and at last the `headline` and `title` nodes which define the configuration for the article title and article description respectively. The `thumbnailKind` is a value that is a composition of the device type (phone or tablet), the device orientation (portrait or landscape) and the teaser type (normal or top). Valid values are e.g. `phone_portrait_top` or `tablet_landscape_normal`.

Name	Type	Description
height	float	the height of the gradient ranged from 0 to 1 where 1 means the full height
alphaStart	float	the transparency value at the bottom of the gradient
alphaEnd	float	the transparency value at the top of the gradient

The upper text is called `headline` and it displays the article's **title**. The lower text is called `title` and contains the article's **description** text. Both of these texts have configurable sizes, fonts and maximum lines which can be set in their respective configuration nodes.

Name	Type	Description
fontSize	int	size of the text
font	String	name of the font
maxLines	int	max amount of lines

Currently colors can only be configured in the Purple Manager.

Example: Complete example configuration

```
{
    "phone": {
        "portrait": {
            "multiColumn": false,
            "teaserWidth": 256,
            "teaserHeight": 144,
            "pageArrowsEnabled": false,
            "pageIndicatorsEnabled": true,
            "pageIndicatorAlignment": "right",
            "types": {
                "topTeaser": {
                    "thumbnailKind" : "phone_portrait_top",
                    "factorY": 1,
                    "spanX": 1,
                    "spanY": 1,
                    "gradient": {
                        "height": 0.6,
                        "alphaStart": 1.0,
                        "alphaEnd": 0.0
                    },
                    "title": {
                        "fontSize": 20,
                        "font": "Roboto-Condensed",
                        "maxLines": 3
                    },
                    "headline": {
                        "fontSize": 12,
                        "font": "Roboto-Condensed",
                        "maxLines": 1
                    }
                },
```

```
                "teaser": {
                    "thumbnailKind" : "phone_portrait_normal",
                    "gradient": {
                        "height": 0.6,
                        "alphaStart": 1.0,
                        "alphaEnd": 0.0
                    },
                    "title": {
                        "fontSize": 20,
                        "font": "Roboto-Condensed",
                        "maxLines": 3
                    },
                    "headline": {
                        "fontSize": 12,
                        "font": "Roboto-Condensed",
                        "maxLines": 1
                    }
                }
            }
        },
        "landscape": {
            "multiColumn": true,
            "teaserWidth": 256,
            "teaserHeight": 144,
            "pageArrowsEnabled": false,
            "pageIndicatorsEnabled": true,
            "pageIndicatorAlignment": "left",
            "types": {
                "topTeaser": {
                    "thumbnailKind" : "phone_landscape_top",
                    "factorY": 1.5,
                    "spanX": 2,
                    "spanY": 2,
                    "gradient": {
                        "height": 0.6,
                        "alphaStart": 1.0,
                        "alphaEnd": 0.0
                    },
                    "title": {
                        "fontSize": 20,
                        "font": "Roboto-Condensed",
                        "maxLines": 3
                    },
                    "headline": {
                        "fontSize": 12,
                        "font": "Roboto-Condensed",
                        "maxLines": 1
                    }
                },
                "teaser": {
                    "thumbnailKind" : "phone_landscape_normal",
                    "gradient": {
                        "height": 0.6,
                        "alphaStart": 1.0,
                        "alphaEnd": 0.0
                    },
                    "title": {
                        "fontSize": 20,
```

```
                        "font": "Roboto-Condensed",
                        "maxLines": 3
                    },
                    "headline": {
                        "fontSize": 12,
                        "font": "Roboto-Condensed",
                        "maxLines": 1
                    }
                }
            }
        }
    },
    "tablet": {
        "portrait": {
            "multiColumn": true,
            "teaserWidth": 256,
            "teaserHeight": 144,
            "pageArrowsEnabled": false,
            "pageIndicatorsEnabled": true,
            "pageIndicatorAlignment": "center",
            "types": {
                "topTeaser": {
                    "thumbnailKind" : "tablet_portrait_top",
                    "factorY": 1.5,
                    "spanX": 2,
                    "spanY": 2,
                    "gradient": {
                        "height": 0.6,
                        "alphaStart": 1.0,
                        "alphaEnd": 0.0
                    },
                    "title": {
                        "fontSize": 28,
                        "font": "Roboto-Condensed",
                        "maxLines": 3
                    },
                    "headline": {
                        "fontSize": 23,
                        "font": "Roboto-Condensed",
                        "maxLines": 1
                    }
                },
                "teaser": {
                    "thumbnailKind" : "tablet_portrait_normal",
                    "gradient": {
                        "height": 0.6,
                        "alphaStart": 1.0,
                        "alphaEnd": 0.0
                    },
                    "title": {
                        "fontSize": 17,
                        "font": "Roboto-Condensed",
                        "maxLines": 3
                    },
                    "headline": {
                        "fontSize": 14,
                        "font": "Roboto-Condensed",
                        "maxLines": 1
```

```
                    }
                }
            }
        },
        "landscape": {
            "multiColumn": true,
            "teaserWidth": 341,
            "teaserHeight": 192,
            "pageArrowsEnabled": true,
            "pageIndicatorsEnabled": false,
            "pageIndicatorAlignment": "center",
            "types": {
                "topTeaser": {
                    "thumbnailKind" : "tablet_landscape_top",
                    "factorY": 1.5,
                    "spanX": 2,
                    "spanY": 2,
                    "gradient": {
                        "height": 0.6,
                        "alphaStart": 1.0,
                        "alphaEnd": 0.0
                    },
                    "title": {
                        "fontSize": 29,
                        "font": "Roboto-Condensed",
                        "maxLines": 3
                    },
                    "headline": {
                        "fontSize": 14,
                        "font": "Roboto-Condensed",
                        "maxLines": 1
                    }
                },
                "teaser": {
                    "thumbnailKind" : "tablet_landscape_normal",
                    "gradient": {
                        "height": 0.6,
                        "alphaStart": 1.0,
                        "alphaEnd": 0.0
                    },
                    "title": {
                        "fontSize": 17,
                        "font": "Roboto-Condensed",
                        "maxLines": 3
                    },
                    "headline": {
                        "fontSize": 11,
                        "font": "Roboto-Condensed",
                        "maxLines": 1
                    }
                }
            }
        }
    }
}
```

Background Image

The default background image for the channel tiles can be changed by adding a `channel_placeholder.png` image to the dynamic resources. Just like other images, it is possible to leave multiple images with different resolutions.

Tutorials

Channel tutorials can be configured in the `onboarding_channel_overview.json` file. It consists of two configurations, one for portrait and landscape each.

Structure

```
{
    "phone": {
        "config": {
                ...
        }
    },
    "tablet": {
        "config": {
                ...
        }
    }
}
```

The first part of the `config` node consists of the style configuration. This is currently only evaluated by iOS. For Android devices colors have to be set in the Purple Manager.

Example: Style configuration

```
"delay": 1.0,
"titleFontSize": 22.0,
"titleFontPostScriptName": "Roboto-Condensed",
"titleTextColor": "#f00000ff",
"titleShadowEnabled": true,
"captionFontSize": 18.0,
"captionFontPostScriptName": "Roboto-Condensed",
"captionTextColor": "#ffffffff",
"captionShadowEnabled": true,
"buttonFontSize": 18.0,
"buttonFontPostScriptName": "Roboto-Condensed",
"buttonTextColor": "#ffffffff",
"buttonBackgroundColor": "#f00000ff",
"focusRingColor": "#f0000000",
```

The next part defines the initial tutorial screen. Its possible to set texts for the title, description, footer and button here. Also there is an option to disable the initial tutorial screen.

Example: Initial tutorial screen configuration

```
"intro":
{
```

```
    "enabled": true,
    "title": "Die neue Channel App",
    "description": "Erleben Sie multimedial aufbereitete Ausgaben und News in einer␣
↪App!",
    "labelFormat": "",
    "buttonTitle": "Los geht's"
},
```

The configuration of the initial screen is followed by the header config. The header is shown at the top of the display for the whole duration of the tutorial. Its also possible to disable the header.

Example: Header configuration

```
"header":
{
    "enabled": false,
    "title": "",
    "subtitle": ""
},
```

At last the configuration for the individual tutorial screens happens. The `views` node is a list that contains configurations for all the screens that are shown during the tutorial. These screens will be displayed in the same order as they are defined in the file. These entries are mostly the same as config for the initial screen with the exception of one additional value. The `name` value defines the item that will be highlighted by the current tutorial screen. Currently only the following values are supported.

Name	Description
left_side_panel_button	the menu button on the top left of the screen
any_channel_article	first teaser on the screen
channel_next_button	the next channel button on the right side of the screen
channel_page_indicator	channel indicator dots on the top of the screen

Example: Views configuration

```
"views": [
     {
     "name": "left_side_panel_button",
     "enabled": true,
     "title": "Seitenmenü",
     "description": "Ihre digitalen Ausgaben können Sie im KIOSK herunterladen.",
     "labelFormat": "Tipp %d von %d",
     "buttonTitle": "Weiter"
     },
     {
     "name": "any_channel_article",
     "enabled": true,
     "title": "Täglich neue Artikel",
     "description": "Täglich aktuelle Artikel für Sie ausgewählt.",
     "buttonTitle": "Weiter",
     "labelFormat": "Tipp %d von %d",
     "scale": 0.7
     },
```

```
    {
    "name": "channel_next_button",
    "enabled": false,
    "title": "Weitere Kategorien",
    "description": "Per Swipe zu vielen weiteren Kategorien und Videos gelangen! ",
    "labelFormat": "Tipp %d von %d",
    "buttonTitle": "Fertig"
    },
    {
    "name": "channel_page_indicator",
    "enabled": true,
    "title": "Weitere Kategorien",
    "description": "Per Swipe zu vielen weiteren Kategorien und Videos gelangen! ",
    "labelFormat": "Tipp %d von %d",
    "buttonTitle": "Fertig"
    }
    ]
}
```

Example: Full example tutorial configuration

```
{
    "phone": {
        "config": {
            "delay": 1.0,
            "titleFontSize": 22.0,
            "titleFontPostScriptName": "Roboto-Condensed",
            "titleTextColor": "#f00000ff",
            "titleShadowEnabled": true,
            "captionFontSize": 18.0,
            "captionFontPostScriptName": "Roboto-Condensed",
            "captionTextColor": "#ffffffff",
            "captionShadowEnabled": true,
            "buttonFontSize": 18.0,
            "buttonFontPostScriptName": "Roboto-Condensed",
            "buttonTextColor": "#ffffffff",
            "buttonBackgroundColor": "#f00000ff",
            "focusRingColor": "#f0000000",
            "intro":
            {
                "enabled": true,
                "title": "Die neue Channel App",
                "description": "Erleben Sie multimedial aufbereitete Ausgaben und␣
↪News in einer App!",
                "labelFormat": "",
                "buttonTitle": "Los geht's"
            },
            "header":
            {
                "enabled": false,
                "title": "",
                "subtitle": ""
            },
            "views": [
                      {
                      "name": "left_side_panel_button",
```

```
                    "enabled": true,
                    "title": "Seitenmenü",
                    "description": "Ihre digitalen Ausgaben können Sie im KIOSK␣
↪herunterladen.",
                    "labelFormat": "Tipp %d von %d",
                    "buttonTitle": "Weiter"
                    },
                    {
                    "name": "any_channel_article",
                    "enabled": true,
                    "title": "Täglich neue Artikel",
                    "description": "Täglich aktuelle Artikel für Sie ausgewählt.",
                    "buttonTitle": "Weiter",
                    "labelFormat": "Tipp %d von %d",
                    "scale": 0.7
                    },
                    {
                    "name": "channel_next_button",
                    "enabled": false,
                    "title": "Weitere Kategorien",
                    "description": "Per Swipe zu vielen weiteren Kategorien und␣
↪Videos gelangen! ",
                    "labelFormat": "Tipp %d von %d",
                    "buttonTitle": "Fertig"
                    },
                    {
                    "name": "channel_page_indicator",
                    "enabled": true,
                    "title": "Weitere Kategorien",
                    "description": "Per Swipe zu vielen weiteren Kategorien und␣
↪Videos gelangen! ",
                    "labelFormat": "Tipp %d von %d",
                    "buttonTitle": "Fertig"
                    }
                    ]
        }
    },
    "tablet": {
        "config": {
            "delay": 1.0,
            "titleFontSize": 22.0,
            "titleFontPostScriptName": "Roboto-Condensed",
            "titleTextColor": "#f00000ff",
            "titleShadowEnabled": true,
            "captionFontSize": 18.0,
            "captionFontPostScriptName": "Roboto-Condensed",
            "captionTextColor": "#ffffffff",
            "captionShadowEnabled": true,
            "buttonFontSize": 18.0,
            "buttonFontPostScriptName": "Roboto-Condensed",
            "buttonTextColor": "#ffffffff",
            "buttonBackgroundColor": "#f00000ff",
            "focusRingColor": "#f0000000",
            "intro":
            {
                "enabled": true,
                "title": "Die neue Channel App",
                "description": "Erleben Sie multimedial aufbereitete Ausgaben und␣
↪News in einer App!",
```

```
                "labelFormat": "",
                "buttonTitle": "Los geht's"
            },
            "header":
            {
                "enabled": false,
                "title": "",
                "subtitle": ""
            },
            "views": [
                        {
                        "name": "left_side_panel_button",
                        "enabled": true,
                        "title": "Seitenmenü",
                        "description": "Ihre digitalen Ausgaben können Sie im KIOSK␣
↪herunterladen.",
                        "labelFormat": "Tipp %d von %d",
                        "buttonTitle": "Weiter"
                        },
                        {
                        "name": "any_channel_article",
                        "enabled": true,
                        "title": "Täglich neue Artikel",
                        "description": "Täglich aktuelle Artikel für Sie ausgewählt.",
                        "buttonTitle": "Weiter",
                        "labelFormat": "Tipp %d von %d",
                        "scale": 0.5
                        },
                        {
                        "name": "channel_next_button",
                        "enabled": true,
                        "title": "Weitere Kategorien",
                        "description": "Per Swipe zu vielen weiteren Kategorien und␣
↪Videos gelangen! ",
                        "labelFormat": "Tipp %d von %d",
                        "buttonTitle": "Fertig"
                        },
                        {
                        "name": "channel_page_indicator",
                        "enabled": false,
                        "title": "Weitere Kategorien",
                        "description": "Per Swipe zu vielen weiteren Kategorien und␣
↪Videos gelangen! ",
                        "labelFormat": "Tipp %d von %d",
                        "buttonTitle": "Fertig"
                        }
                        ]
        }
    }
}
```

3.3.4 Tracking

The tracking is configured by editing `tracking_config.json`. For further details about configuring tracking see: *Tracking*

3.3.5 Sharing

The `sharing.properties` contains the texts that will displayed during the share process. This file will be evaluated by the Purple Manager and not the app itself. The contents of this file will be delivered to the app during the status request which happens on every app start. The purpose of this file is to set an app url and texts that will be printed during app or issue shares.

Example

```
app.url=http://www.example.com
app=Hallo, ich möchte Dir die App empfehlen: {appUrl}
issue=Hallo, folgende Ausgabe möchte ich dir empfehlen: {issueUrl} Hier kannst Du dir␣
↪die App herunterladen: http://www.example.com

app.plaintext=Hallo, ich möchte Dir die App empfehlen
issue.plaintext=Hallo, folgende Ausgabe möchte ich dir empfehlen. Hier kannst Du dir␣
↪die App herunterladen: http://www.example.com
```

Hint: This file must be encoded as ISO 8859-1.

3.3.6 Feedback E-Mail

The feedback e-mail can be configured using the `email_feedback_config.json`, `email_feedback_subject.mustache` and `email_feedback_body.mustache` files. The recipients can be set in the `email_feedback_config.json`, while the other two files define the subject and the body for the e-mail.

Every file has a default configuration. These files can be configured independently. You can override the default configuration by putting your custom configuration file into the root directory (e.g. default/email_feedback_body.mustache). Your custom file must have the same name like the default file.

All templates are encoded in UTF-8.

Placeholder for Templates

Key	Description	Example Value	Localization Key
appName	app name	“Example Test App”	-
appVersionLabel	label for app version	“App Version”	app_version_title
appVersion	app version	“1.0-SNAPSHOT”	-
versionSectionLabel	headline: version section	“Versionen”	version_title
deviceSectionLabel	headline: device section	“Device”	device_title
deviceModelLabel	label for device model	“Device Model”	device_model
deviceModel	device model	“Pixel C”	-
osVersionLabel	label for os version	“OS Version”	os_version
osVersion	os version	Android: “7.0”, iOS: ??	-
manufacturerLabel	label for manufacturer	“Manufacturer”	manufacturer
manufacturer	manufacturer of the device	Android: “Samsung” / “Google” etc, iOS: “Apple”	-
connectionLabel	label for connection	“Connection”	connection
connection	connection info	Android: z.B. “WIFI”, iOS: “OFFLINE”, “WIFI”, “WWAN”	-
deviceIdLabel	label for device-id	“Device Id”	device_id_title
deviceId	device-id	Android: 31c290b15bn6adee, iOS: 8C224FG9-4665-BEFO-834F2C7D7AFF	-
purpleVersionBlock	all relevant Purple versions	“Purple: 3.1.0, Engine: 2.0.0”	-
platform	platform identifier	“Android”, “Kindle” oder “iOS”	-

Example for Files

email_feedback_config.json

```
{
  "recipients": {
    "to": [
      "receiver1@example.com",
      "receiver2@example.com"
    ],
    "cc": [
      "cc1@example.com",
      "cc2@example.com"
    ],
```

```
    "bcc": [
      "bcc1@example.com",
      "bcc2@example.com"
    ]
  }
}
```

email_feedback_subject.mustache

```
{{appName}} - App Feedback ({{platform}})
```

email_feedback_body.mustache

```
--
{{versionSectionLabel}}
{{appVersionLabel}}: {{appVersion}}
{{purpleVersionBlock}}

{{deviceSectionLabel}}
{{deviceModelLabel}}: {{deviceModel}}
{{osVersionLabel}}: {{osVersion}}
{{manufacturerLabel}}: {{manufacturer}}
{{deviceIdLabel}}: {{deviceId}}
{{connectionLabel}}: {{connection}}
```

CHAPTER 4

Tracking

4.1 Overview

Purple DS apps support tracking via several different tracking services. There are three different types of events:

1. Actions
2. Views
3. Purchases

Additionally, if the tracking service supports this, there are several attributes which describe the state of the app / usage by a user.

4.2 Tracking Services

Currently Purple DS apps support several different tracking services.

Warning: Not all tracking services support all event types. See the linked pages below for detailed information about the supported event types of each tracking service.

- *Adjust*
- *Adobe Analytics*
- *Amazon Pinpoint*
- *AT Internet*
- *Braze (Appboy)*
- *Facebook*
- *Firebase Analytics*

- *Flurry*
- *Google Analytics*
- *IVW*

4.2.1 Adjust

New in version 2.1.0.

is a tracking service which supports tracking events (actions and purchases) through tokens generated by Adjust.

Campaign Tracking

Adjust supports campaign tracking for both app installs and deep links (Purple Action-URLs). Please consult the for information on how to create campaigns in Adjust.

Events

Overview

Key-Name in Config	Action Templates	View Templates	Purchase Templates	Attribute Templates
adjust	• `token_android` • `token_ios` • `token_kindle`	unsupported	• `token_android` • `token_ios` • `token_kindle`	unsupported

Actions

Adjust supports tracking of action events. The tokens generated by Adjust can be configured per platform by using the specific template placeholders:

- token_android
- token_ios
- token_kindle

Views

Adjust does not support view events.

Purchases

Adjust supports tracking of purchase events. The tokens generated by Adjust can be configured per platform by using the specific template placeholders:

- token_android
- token_ios
- token_kindle

Additionally the revenue (both price and currency) of the purchase is tracked.

Attributes

Adjust does not support storing attributes.

Event parameters

Adjust does not support sending custom parameters.

Configuration Example

```
{
  "adjust": {
    "eventsEnabledByDefault": false,
    "purchasesEnabledByDefault": false,
    "events": {
      "APP_BOOKMARK_ADDED": {
        "enabled": true,
        "templates": {
          "token_kindle": "<TOKEN>",
          "token_android": "<TOKEN>",
          "token_ios": "<TOKEN>"
        }
      }
    },
    "purchases": {
      "KIOSK_ISSUE_PURCHASED": {
        "enabled": true,
        "templates": {
          "token_kindle": "<TOKEN>",
          "token_android": "<TOKEN>",
          "token_ios": "<TOKEN>"
        }
      }
    }
  }
}
```

4.2.2 Adobe Analytics

New in version 3.4.0.

is a tracking service which supports tracking events (actions, views, purchases).

Events

Overview

Key-Name in Config	Action Templates	View Templates	Purchase Templates	Attribute Templates
adobeanalytics	`action`	`name`	`action`	unsupported

Actions

Adobe Analytics supports tracking of action events. The actual value sent to Adobe is configured through template with the key `action`.

Views

Adobe Analytics supports tracking of view events. The actual value sent to Adobe is configured through template with the key `name`.

Purchases

Adobe Analytics supports tracking of purchase events. The actual value sent to Adobe is configured through template with the key `action`.

Attributes

Adobe Analytics does not support storing attributes.

Event parameters

Adobe Analytics supports sending custom parameters for actions, views and purchases.

Configuration Example

```
{
  "adobeanalytics": {
    "events": {
      "APP_BOOKMARK_ADDED": {
        "templates": {
          "action": "Bookmark added {{CONTENT_NAME}}"
        },
        "parameters": {
          "pageinfo.brand": "purple"
        }
      }
    },
    "views": {
      "KIOSK_CHANNEL_FEED": {
        "templates": {
```

```
          "name": "Kiosk channel {{PUBLICATION_NAME}}"
        },
        "parameters": {
          "pageinfo.brand": "purple"
        }
      }
    },
    "purchases": {
      "KIOSK_ISSUE_PURCHASED": {
        "templates": {
          "action": "Issue purchased {{ISSUE_ID}}"
        },
        "parameters": {
          "pageinfo.brand": "purple",
          "issue.id": "{{ISSUE_ID}}"
        }
      }
    }
  }
}
```

4.2.3 AT Internet

New in version 5.2.

is a tracking service which supports tracking events (actions and views).

Events

Overview

Key-Name in Config	Action Templates	View Templates	Purchase Templates	Attribute Templates
atinternet	• `name` • `chapter1` • `chapter2` • `chapter3` • `level2` • `action`	• `name` • `chapter1` • `chapter2` • `chapter3` • `level2`	unsupported	unsupported

Actions

AT Internet supports tracking of action events. With AT Internet, each action is tracked as a .

For each event, the properties of the Gesture event can be configured via separate templates: `name`, `chapter1`, `chapter2`, `chapter3` and `level2`.

Note: The `level2` template must evaluate to a valid integer value.

Each Gesture event can have a different `action`. This is configured with the corresponding `action` template. The values need to be one of:

- `touch`
- `navigation`
- `download`
- `exit`
- `search`

If no `action` or an invalid value was provided, the event will be sent as a `touch` event.

On the web platform, gestures are always sent as `click` events.

Views

AT Internet supports tracking of view events. With AT Internet, each view is tracked as a .

For each event, the properties of the Screen event can be configured via separate templates: `name`, `chapter1`, `chapter2`, `chapter3` and `level2`.

Note: The `level2` template must evaluate to a valid integer value.

Purchases

AT Internet does not support tracking of purchases.

Attributes

AT Internet does not support storing attributes per user.

Event parameters

AT Internet does support sending custom parameters for actions.

Configuration Example

```
{
  "atinternet": {
    "events": {
      "APP_BOOKMARK_ADDED": {
        "enabled": true,
        "templates": {
          "name": "Bookmark added",
          "chapter1": "{{BOOKMARK_TITLE}}",
          "chapter2": "Chapter 2",
          "chapter3": "Chapter 3",
          "level2": "2",
          "action": "touch"
```

```
        }
      }
    },
    "views": {
      "PRESENTER_PAGE": {
        "enabled": true,
        "templates": {
          "name": "{{CONTENT_NAME}}",
          "chapter1": "{{PAGE_LABEL}}",
          "chapter2": "Chapter 2",
          "chapter3": "Chapter 3",
          "level2": "3"
        }
      }
    },
    "attributes": {
    }
  }
}
```

4.2.4 Braze

New in version 2.1.0.

Warning: Please note: Due to changed conditions for app developers on the part of Braze, we can no longer support this service free of charge. Therefore, we do not provide any general warranty for correct operation when activating this feature. Individual support inquiries are welcome via support@sprylab.com.

(formerly known as Appboy) is a tracking service which supports tracking events (actions / purchases) and storing attributes.

Events

Overview

Key-Name in Config	Action Templates	View Templates	Purchase Templates	Attribute Templates
appboy	`action`	unsupported	unsupported	`name`

Note: Due to compatibility with older versions we keep the previous name as the identifier in the config.

Actions

Braze supports tracking of action events. The actual value sent to Braze is configured through template with the key `action`.

Views

Braze does not support view events.

Purchases

Braze supports tracking of purchase events. Purchase events cannot be configured (besides enabling/disabling the whole event) and always the track the product id, currency code and price.

Attributes

Braze supports storing attributes per user. The name of the attribute can be configured through the `name` template.

Event parameters

Braze does not support sending custom parameters.

Configuration Example

```
{
  "appboy": {
    "eventsEnabledByDefault": false,
    "purchasesEnabledByDefault": false,
    "attributesEnabledByDefault": false,
    "events": {
      "APP_BOOKMARK_ADDED": {
        "enabled": true,
        "templates": {
          "action": "Bookmark added"
        }
      }
    },
    "purchases": {
      "KIOSK_ISSUE_PURCHASED": {
        "enabled": true
      }
    },
    "attributes": {
      "HAS_ACTIVE_SUBSCRIPTION": {
        "enabled": true,
        "templates": {
          "name": "Has an active subscription"
        }
      }
    }
  }
}
```

4.2.5 Facebook

New in version 2.1.0.

Changed in version 3.5.0: Added support for action templates and parameters

is a tracking service which supports tracking events (actions and purchases). It also supports tracking the installation of the app.

General

For setting up the App in Facebook Analytics use the following information:

Android Class Name: `com.sprylab.purple.android.app.purple.splash.SplashActivity`

Andriod Key Hash (Preview Apps): `VnOtQRWs9tYehKQDf9SeALlqsxc=`

For Release Apps you need to create a hash from your release keystore. See the for more information.

Events

Overview

Key-Name in Config	Action Templates	View Templates	Purchase Templates	Attribute Templates
facebook	`action`	unsupported	unsupported	unsupported

Actions

Facebook supports tracking of action events. The actual value sent to Facebook is configured through a template with the key `action`.

Hint: Event names and parameter names length must be under 40 characters and contain only alphanumeric characters, '_', '-' or spaces, and cannot start with a space or hyphen.

For more information please see the .

Views

Facebook does not support view events.

Purchases

Facebook supports tracking of purchase events. Purchase events only send the price and currency for the purchase. The product id can be sent with custom parameters.

Warning: "Log In-App Purchase Events Automatically on iOS" setting should be disabled otherwise in-app purchases logging will be duplicated.

To disable the setting follow the steps below:

1. Go to My Apps.

2. Select your app.
3. Click on the settings tab on the left nav.
4. Find the section labeled iOS.
5. Disable the switch called “Automatically Log In-App Purchase Events on iOS”.

For more information please see the .

Attributes

Facebook does not support storing attributes per user.

Event parameters

Facebook supports sending custom parameters for actions and purchases.

Hint: An event can have up to 25 parameters. This doesn’t just mean for each call, but for all invocations that use that event name.

If you need to remove obsolete parameters - you can deactivate parameters by following the instructions in the Facebook help center.

The length of each parameter value can be no more than 100 characters.

For more information please see the .

Configuration Example

```
{
  "facebook": {
    "events": {
      "APP_BOOKMARK_ADDED": {
        "templates": {
          "action": "Bookmark added {{CONTENT_NAME}}"
        },
        "parameters": {
          "pageinfo.brand": "purple"
        }
      }
    },
    "purchases": {
      "KIOSK_ISSUE_PURCHASED": {
        "templates": {
          "action": "Issue purchased {{ISSUE_ID}}"
        },
        "parameters": {
          "product": "{{PRODUCT_ID}}",
          "issue.id": "{{ISSUE_ID}}"
        }
      }
    }
```

```
    }
}
```

4.2.6 Firebase Analytics

New in version 3.10.0.

is a tracking service which supports tracking events (actions, views and purchases) and storing user attributes.

Events

Overview

Key-Name in Config	Action Templates	View Templates	Purchase Templates	Attribute Templates
firebase_analytics	`action`	`name`	unsupported	`name`

Actions

Firebase Analytics supports tracking of action events. The actual value sent to Firebase Analytics is configured through a template with the key `action`.

Hint: Event names must be under 40 characters and contain only alphanumeric characters or '_' and must start with an alphabetic character.

The length of each parameter value can be no more than 100 characters.

Views

Firebase Analytics supports tracking of view events. The view name send to Firebase Analytics is configured through template with the key `name`.

Purchases

Firebase Analytics supports tracking of purchase events. Purchase events cannot be configured (besides enabling/disabling the whole event) and always track the product id, product name, currency an quantity.

Attributes

Firebase Analytics supports storing attributes per user. The name of the attribute can be configured through the `name` template.

Hint: Attribute names must be under 40 characters and contain only alphanumeric characters or '_' and must start with an alphabetic character.

The length of each parameter value can be no more than 100 characters.

Event parameters

Firebase Analytics supports sending custom parameters for actions.

Hint: An event can have up to 25 parameters. This doesn't just mean for each call, but for all invocations that use that event name.

Configuration Example

```
{
  "firebase_analytics": {
    "events": {
      "APP_BOOKMARK_ADDED": {
        "enabled": true,
        "templates": {
          "action": "Bookmark added"
        },
        "parameters": {
          "name": "{{CONTENT_NAME}}"
        }
      }
    },
    "views": {
      "APP_BOOKMARKS": {
        "enabled": true,
        "templates": {
          "name": "App bookmarks"
        }
      }
    },
    "attributes": {
      "HAS_ACTIVE_SUBSCRIPTION": {
        "enabled": true,
        "templates": {
          "name": "Has an active subscription"
        }
      }
    }
  }
}
```

4.2.7 Flurry

New in version 2.1.0.

is a tracking service which supports tracking events (actions).

Events

Overview

Key-Name in Config	Action Templates	View Templates	Purchase Templates	Attribute Templates
flurry	`action`	unsupported	unsupported	unsupported

Actions

Flurry supports tracking of action events. The actual value sent to Flurry is configured through template with the key `action`.

Views

Flurry does not support view events.

Purchases

Flurry does not support purchase events.

Attributes

Flurry does not support storing attributes per user.

Event parameters

Flurry does not support sending custom parameters.

Configuration Example

```
{
  "flurry": {
    "eventsEnabledByDefault": false,
    "events": {
      "APP_BOOKMARK_ADDED": {
        "enabled": true,
        "templates": {
          "action": "Bookmark added"
        }
      }
    }
  }
}
```

4.2.8 Google Analytics

New in version 2.1.0.

Deprecated since version 5.0: Google Analytics is not available anymore in Apps.

is a tracking service which supports tracking events (actions, views, purchases).

Campaign Tracking

Google Analytics supports campaign tracking for both app installs and deep links (Purple Action-URLs). For information on how to adjust your Play Store links you can consult the . For Purple Action-URLs you just have to append your campaign parameters to the URL.

IP anonymization

Anonymization of IPs has been enabled for App Template 3.9 and above since 30.05.2018.

See the for more information about how the anonymization is done.

Events

Overview

Key-Name in Config	Action Templates	View Templates	Purchase Templates	Attribute Templates
google_analytics	• `category` • `action` • `label`	`name`	unsupported	unsupported

Actions

Google Analytics supports tracking of action events. The actual values sent to Google Analytics is configured through `category`, `action` and `label` templates.

Views

Google Analytics supports tracking of action events. The view name send to Google Analytics is configured through template with the key `name`.

Purchases

Google Analytics supports tracking of purchase events. Purchase events cannot be configured (besides enabling/disabling the whole event) and always track the product id, currency code and price.

Attributes

Google Analytics does not support storing attributes per user.

Event parameters

Google Analytics does not support sending custom parameters.

Configuration Example

```
{
  "google_analytics": {
    "eventsEnabledByDefault": false,
    "viewsEnabledByDefault": false,
    "purchasesEnabledByDefault": false,
    "events": {
      "APP_BOOKMARK_ADDED": {
        "enabled": true,
        "templates": {
          "category": "App",
          "action": "Bookmark added",
          "label": "{{CONTENT_NAME}}"
        }
      }
    },
    "views": {
      "APP_BOOKMARKS": {
        "enabled": true,
        "templates": {
          "name": "App bookmarks"
        }
      }
    },
    "purchases": {
      "KIOSK_ISSUE_PURCHASED": {
        "enabled": true
      }
    }
  }
}
```

4.2.9 IVW

Warning: As of PK 3.15 for iOS and PK 5.0 for Android this tracking service is not supported anymore.

New in version 2.2.0.

is a tracking service which supports tracking actions (views).

Events

Overview

Key-Name in Config	Action Templates	View Templates	Purchase Templates	Attribute Templates
ivw	unsupported	• `category` • `comment`	unsupported	unsupported

Actions

IVW does not support action events.

Views

IVW supports tracking of view events. The actual values sent to IVW is configured through `category` and `comment` templates.

Purchases

IVW does not support purchase events.

Attributes

IVW does not support storing attributes per user.

Event parameters

IVW does not support sending custom parameters.

Configuration Example

```
{
  "ivw": {
    "viewsEnabledByDefault": false,
    "views": {
      "APP_BOOKMARKS": {
        "enabled": true,
        "templates": {
          "category": "<IVW Code>",
          "comment": "Bookmark added"
        }
      }
    }
  }
}
```

4.2.10 Amazon Pinpoint

New in version 3.6.0.

is a tracking service which supports tracking events (actions and purchases) and storing user attributes.

Events

Overview

Key-Name in Config	Action Templates	View Templates	Purchase Templates	Attribute Templates
pinpoint	`action`	unsupported	unsupported	`name`

Actions

Amazon Pinpoint supports tracking of action events. The actual value sent to Amazon Pinpoint is configured through a template with the key `action`.

Note: Event names can only be 50 characters long and will be truncated if they exceed this limit.

Views

Amazon Pinpoint does not support view events.

Purchases

Amazon Pinpoint supports tracking of purchase events. Purchase events cannot be configured (besides enabling/disabling the whole event) and always track the product id, currency code and price.

Attributes

Amazon Pinpoint supports storing attributes per user. The name of the attribute can be configured through the `name` template.

Event parameters

Amazon Pinpoint supports sending custom parameters for actions.

Configuration Example

```
{
  "pinpoint": {
    "events": {
      "APP_BOOKMARK_ADDED": {
        "templates": {
          "action": "Bookmark added {{CONTENT_NAME}}"
        },
        "parameters": {
          "pageinfo.brand": "purple"
        }
      }
    },
    "attributes": {
      "HAS_ACTIVE_SUBSCRIPTION": {
        "enabled": true,
        "templates": {
          "name": "Has an active subscription"
        }
      }
    }
  }
}
```

4.3 Configuration

There are two sets of configurations:

- the default/base configuration, included in the Purple Kit. See https://redmine.purplepublish.com/issues/5094 for the latest version.
- the app configuration, included in the dynamic resources

The app configuration is merged with the "default" configuration. This means that every entry in the app configuration overrides the value in the default configuration.

This allows to have standard configurations for all events which can be overridden for each tracking service.

For this purpose a new JSON file *tracking_config.json* is used in the dynamic resources. This configuration makes it possible to map internal event names to custom event names.

The JSON is structured as follows:

tracking_config.json

```
{
  "default": {
    "eventsEnabledByDefault": false,
    "viewsEnabledByDefault": false,
    "purchasesEnabledByDefault": false,
    "attributesEnabledByDefault": false,
    "events": {
      "<internal_event_key>": {
        "enabled": false,
        "templates": {
          "<template_name>": "<Text to send to tracking service>"
        },
        "parameters": {
```

```
          "paramKey1": "paramValue1"
        }
      }
    },
    "views": {
    },
    "purchases": {
    },
    "attributes": {
    }
  },
  "<tracking_service_name>": {
    "eventsEnabledByDefault": false,
    "viewsEnabledByDefault": false,
    "purchasesEnabledByDefault": false,
    "attributesEnabledByDefault": false,
    "events": {
      "<internal_event_key>": {
        "enabled": false,
        "templates": {
          "<template_name>": "<Text to send to tracking service>"
        }
      }
    },
    "views": {
    },
    "purchases": {
    },
    "attributes": {
    }
  },
}
```

Each event has the following configuration options:

4.3.1 enabled

Each event can be selectively enabled or disabled.

4.3.2 templates

Each event has templates. They are used to configure the tracking services, e.g. Google Analytics has three templates for action events: “category”, “action” and “label”. The names map to the API of the tracking service. Each tracking service supports different templates.

The templates can contain placeholders. They can be referenced in the config by using the following syntax: `{{PLACEHOLDER_NAME}}`, e.g. `{{ISSUE_ID}}`.

A list of all available placeholders can be found *here*.

4.3.3 parameters

Additional parameters (key-value pairs) can be send with each event if the tracking service supports this. Every key will be included in the event. The values can contain all placeholders supported for the event and will be evaluated

(see above) when sending the event to the service.

4.4 Events

A current list of all available events, parameters and default values can be found *here*.

4.4.1 Actions

APP_BOOKMARK_ADDED

New in version (Android): 2.1.0

New in version (iOS): 2.1.0

A bookmark was added by the user.

Template placeholder	Example values
CONTENT_ID	<PUBLICATION_ID>/<ISSUE_ID>
CONTENT_NAME	displayName (Name) of the Issue
BOOKMARK_TITLE	Issues: same as CONTENT_NAME, Articles: displayName (Name) of Publication
BOOKMARK_SECTION	Issues: Section of the ToC, Articles: displayName (Name) of Issue

APP_BOOKMARK_DELETED

New in version (Android): 2.1.0

New in version (iOS): 2.1.0

A bookmark was deleted by the user.

Template placeholder	Example values
CONTENT_ID	<PUBLICATION_ID>/<ISSUE_ID>
CONTENT_NAME	displayName (Name) of the Issue
BOOKMARK_TITLE	Issues: same as CONTENT_NAME, Articles: displayName (Name) of Publication
BOOKMARK_SECTION	Issues: Section of the ToC, Articles: displayName (Name) of Issue

APP_START

New in version (Android): 2.1.0

New in version (iOS): 2.1.0

New in version (Web Kiosk): 3.2.2

The app has been started.

This event has no template placeholders.

APP_STOP

New in version (Android): 2.1.0

New in version (iOS): 2.1.0

New in version (Web Kiosk): 3.2.2

The app has been stopped.

This event has no template placeholders.

APP_FOREGROUND

New in version (Android): 2.1.0

New in version (iOS): 2.1.0

The app has been resumed, e.g. through the recent tasks.

This event has no template placeholders.

APP_BACKGROUND

New in version (Android): 2.1.0

New in version (iOS): 2.1.0

The app has been minimized, e.g. using the home button.

This event has no template placeholders.

APP_SHARED

New in version (Android): 2.1.0

New in version (iOS): 2.5.0

The app has been shared.

This event has no template placeholders.

APP_CONTENT_SHARED

New in version (Android): 2.1.0

New in version (iOS): 2.5.0

The user has shared the currently visible content.

Template placeholder	Example values
CONTENT_ID	The content id of the shared content.
CONTENT_NAME	The name of the shared content, e.g. the issue name.

KIOSK_PROMOTION_OPEN_ACTION

New in version (Android): 2.1.0

New in version (iOS): 2.1.0

An url has been opened from the promotion area in the kiosk.

Template placeholder	Example values
ACTION_URL	The url which has been opened.

KIOSK_PUBLICATION_OPENED

New in version (Android): 2.1.0

New in version (iOS): 2.5.0

New in version (Web Kiosk): 3.2.2

A publication has been selected in the publication filter menu in the kiosk.

Template placeholder	Example values
PUBLICATION_ID	The id of the publication.
PUBLICATION_NAME	The name of the publication.

Additionally you can use all custom properties which are configured for the publication. The name of the property is used as the placeholder key.

KIOSK_CHANNEL_OPENED

New in version (Android): 2.1.0

New in version (iOS): 2.1.0

A channel has been selected in the channel pager.

Template placeholder	Example values
PUBLICATION_ID	The id of the publication.
PUBLICATION_NAME	The name of the publication.

KIOSK_COUPON_ACTIVATED

New in version (Android): 2.1.0

New in version (iOS): 2.1.0

A coupon code has been activated.

Template placeholder	Example values
COUPON_CODE	The coupon code which has been activated.

Warning: This event has been renamed to KIOSK_SUBSCRIPTION_CODE_ACTIVATED in version 2.3. All configurations have to be manually adjusted to use the new event name.

KIOSK_COUPON_DEACTIVATED

New in version (Android): 2.1.0

New in version (iOS): 2.1.0

A coupon code has been deactivated.

Template placeholder	Example values
COUPON_CODE	The coupon code which has been deactivated.

Warning: This event has been renamed to KIOSK_SUBSCRIPTION_CODE_DEACTIVATED in version 2.3. All configurations have to be manually adjusted to use the new event name.

KIOSK_SUBSCRIPTION_CODE_ACTIVATED

New in version (Android): 2.3.0

New in version (iOS): 2.3.0

New in version (Web Kiosk): 3.2.2

A subscription code has been activated.

Template placeholder	Example values
SUBSCRIPTION_CODE	The coupon code which has been activated.

KIOSK_SUBSCRIPTION_CODE_DEACTIVATED

New in version (Android): 2.3.0

New in version (iOS): 2.3.0

New in version (Web Kiosk): 3.2.2

A subscription code has been deactivated.

Template placeholder	Example values
SUBSCRIPTION_CODE	The coupon code which has been deactivated.

KIOSK_ISSUE_DELETED

New in version (Android): 2.1.0

New in version (iOS): 2.1.0

An issue has been deleted.

Template placeholder	Example values
ISSUE_ID	The id of the issue.
ISSUE_NAME	The name of the issue.
PUBLICATION_ID	The id of the publication.
PUBLICATION_NAME	The name of the publication.

Additionally you can use all custom properties which are configured for the publication and issue. The name of the property is used as the placeholder key.

KIOSK_ISSUE_DOWNLOAD_STARTED

New in version (Android): 2.5.0

New in version (iOS): 2.5.0

An issue download has been started.

Template placeholder	Example values
ISSUE_ID	The id of the issue.
ISSUE_NAME	The name of the issue.
PUBLICATION_ID	The id of the publication.
PUBLICATION_NAME	The name of the publication.

Additionally you can use all custom properties which are configured for the publication and issue. The name of the property is used as the placeholder key.

KIOSK_ISSUE_DOWNLOAD_CANCELLED

New in version (Android): 2.5.0

New in version (iOS): 2.5.0

An issue download has been cancelled.

Template placeholder	Example values
ISSUE_ID	The id of the issue.
ISSUE_NAME	The name of the issue.
PUBLICATION_ID	The id of the publication.
PUBLICATION_NAME	The name of the publication.

Additionally you can use all custom properties which are configured for the publication and issue. The name of the property is used as the placeholder key.

KIOSK_ISSUE_DOWNLOAD_FAILED

New in version (Android): 2.5.0

New in version (iOS): 2.5.0

An issue download has failed.

Template placeholder	Example values
ISSUE_ID	The id of the issue.
ISSUE_NAME	The name of the issue.
PUBLICATION_ID	The id of the publication.
PUBLICATION_NAME	The name of the publication.

Additionally you can use all custom properties which are configured for the publication and issue. The name of the property is used as the placeholder key.

KIOSK_ISSUE_DOWNLOADED

New in version (Android): 2.1.0

New in version (iOS): 2.1.0

An issue has been downloaded.

Template placeholder	Example values
ISSUE_ID	The id of the issue.
ISSUE_NAME	The name of the issue.
PUBLICATION_ID	The id of the publication.
PUBLICATION_NAME	The name of the publication.

Additionally you can use all custom properties which are configured for the publication and issue. The name of the property is used as the placeholder key.

KIOSK_ISSUE_PREVIEW_DOWNLOADED

New in version (Android): 2.1.0

New in version (iOS): 2.1.0

A preview issue has been downloaded.

Template placeholder	Example values
ISSUE_ID	The id of the issue.
ISSUE_NAME	The name of the issue.
PUBLICATION_ID	The id of the publication.
PUBLICATION_NAME	The name of the publication.

Additionally you can use all custom properties which are configured for the publication and issue. The name of the property is used as the placeholder key.

KIOSK_ISSUE_OPENED

New in version (Android): 2.1.0

New in version (iOS): 2.1.0

New in version (Web Kiosk): 3.2.2

An issue has been opened.

Template placeholder	Example values
ISSUE_ID	The id of the issue.
ISSUE_NAME	The name of the issue.
PUBLICATION_ID	The id of the publication.
PUBLICATION_NAME	The name of the publication.

Additionally you can use all custom properties which are configured for the publication and issue. The name of the property is used as the placeholder key.

KIOSK_ISSUE_OPEN_FAILED

New in version (Android): 2.5.0

New in version (iOS): ?.?.?

New in version (Web Kiosk): 3.2.2

An issue could not be opened. This can happen if the user tried to open an issue though a deep-link / action-url which is not available in the kiosk.

Template placeholder	Example values
ISSUE_ID	The id of the issue.

KIOSK_ISSUE_PREVIEW_OPENED

New in version (Android): 2.1.0

New in version (iOS): 2.1.0

A preview issue has been opened.

Template placeholder	Example values
ISSUE_ID	The id of the issue.
ISSUE_NAME	The name of the issue.
PUBLICATION_ID	The id of the publication.
PUBLICATION_NAME	The name of the publication.

Additionally you can use all custom properties which are configured for the publication and issue. The name of the property is used as the placeholder key.

KIOSK_ISSUE_PURCHASE_CANCELLED

New in version (Android): 2.1.0

New in version (iOS): 2.1.0

An issue purchase has been cancelled/aborted.

Template placeholder	Example values
PRODUCT_ID	The product id of the issue.
ISSUE_ID	The id of the issue.
ISSUE_NAME	The name of the issue.
PUBLICATION_ID	The id of the publication.
PUBLICATION_NAME	The name of the publication.

Additionally you can use all custom properties which are configured for the publication and issue. The name of the property is used as the placeholder key.

KIOSK_ISSUE_PURCHASED

New in version (Android): 2.1.0

New in version (iOS): 2.1.0

An issue has been purchase.

Template placeholder	Example values
PRODUCT_ID	The product id of the issue.
ISSUE_ID	The id of the issue.
ISSUE_NAME	The name of the issue.
PUBLICATION_ID	The id of the publication.
PUBLICATION_NAME	The name of the publication.

Additionally you can use all custom properties which are configured for the publication and issue. The name of the property is used as the placeholder key.

KIOSK_ISSUE_TOC_OPENED

New in version (Android): 2.1.0

New in version (iOS): 2.1.0

The table of contents of an issue have been opened in the issue preview screen.

Template placeholder	Example values
ISSUE_ID	The id of the issue.
ISSUE_NAME	The name of the issue.
PUBLICATION_ID	The id of the publication.
PUBLICATION_NAME	The name of the publication.

Additionally you can use all custom properties which are configured for the publication and issue. The name of the property is used as the placeholder key.

KIOSK_ISSUE_PREVIEW_TOC_OPENED

New in version (Android): 2.1.0

New in version (iOS): 2.1.0

The table of contents of a preview issue have been opened in the issue preview screen.

Template placeholder	Example values
ISSUE_ID	The id of the issue.
ISSUE_NAME	The name of the issue.
PUBLICATION_ID	The id of the publication.
PUBLICATION_NAME	The name of the publication.

Additionally you can use all custom properties which are configured for the publication and issue. The name of the property is used as the placeholder key.

KIOSK_PURCHASES_RESTORED

New in version (iOS): 2.1.0

The in-app purchases have been manually restored.

This event has no template placeholders.

Warning: This events is not available on Android as purchases are restored automatically after each kiosk sync.

KIOSK_PURCHASE_RESTORATION_FAILED

New in version (iOS): 2.1.0

The in-app purchases could not be restored.

This event has no template placeholders.

Warning: This events is not available on Android as purchases are restored automatically after each kiosk sync.

KIOSK_SUBSCRIPTION_PURCHASE_CANCELLED

New in version (Android): 2.1.0

New in version (iOS): 2.1.0

A subscription purchase has been cancelled/aborted.

Template placeholder	Example values
SUBSCRIPTION_ID	The id of the subscription.
SUBSCRIPTION_NAME	The name of the subscription.
PRODUCT_ID	The product id of the subscription.

Additionally you can use all custom properties which are configured for the subscription. The name of the property is used as the placeholder key.

KIOSK_SUBSCRIPTION_PURCHASED

New in version (Android): 2.1.0

New in version (iOS): 2.1.0

A subscription has been purchased.

Template placeholder	Example values
SUBSCRIPTION_ID	The id of the subscription.
SUBSCRIPTION_NAME	The name of the subscription.
PRODUCT_ID	The product id of the subscription.

Additionally you can use all custom properties which are configured for the subscription. The name of the property is used as the placeholder key.

KIOSK_MENU_ENTITLEMENT_LOGIN_SUCCEEDED

New in version (Android): 2.5.0

New in version (iOS): 2.5.0

New in version (Web Kiosk): 3.2.2

This event will be tracked if the entitlement login was successful.

Template placeholder	Example values
USERNAME	The username used to login.

KIOSK_MENU_ENTITLEMENT_LOGIN_FAILED

New in version (Android): 2.5.0

New in version (iOS): 2.5.0

New in version (Web Kiosk): 3.2.2

This event will be tracked if the entitlement login has failed.

Template placeholder	Example values
USERNAME	The username used to login.

KIOSK_ENTITLEMENT_LINK1_OPENED

New in version (Android): 2.5.0

New in version (iOS): 2.5.0

This event will be tracked if the first entitlement link was opened.

Template placeholder	Example values
LABEL	The displayed text of the link.
URL	The actual url of the link.

KIOSK_ENTITLEMENT_LINK2_OPENED

New in version (Android): 2.5.0

New in version (iOS): 2.5.0

This event will be tracked if the second entitlement link was opened.

Template placeholder	Example values
LABEL	The displayed text of the link.
URL	The actual url of the link.

KIOSK_ENTITLEMENT_LINK3_OPENED

New in version (Android): 2.5.0

New in version (iOS): 2.5.0

This event will be tracked if the third entitlement link was opened.

Template placeholder	Example values
LABEL	The displayed text of the link.
URL	The actual url of the link.

PRESENTER_CONTENT_TOC_OPENED

New in version (Android): 2.1.0

New in version (iOS): 2.1.0

New in version (Web Player): 3.1.0

The user has opened the table of contents of the currently visible content.

Template placeholder	Example values
CONTENT_ID	The content id of the current content.
CONTENT_NAME	The name of the current content, e.g. the issue name.

PRESENTER_CONTENT_PREVIEW_TOC_OPENED

New in version (Android): 2.1.0

New in version (iOS): 2.1.0

New in version (Web Player): 3.1.0

The user has opened the table of contents of the currently visible preview content.

Template placeholder	Example values
CONTENT_ID	The content id of the current content.
CONTENT_NAME	The name of the current content, e.g. the issue name.

PRESENTER_CHANNEL_CONTENT_OPENED

New in version (Android): 2.1.0

New in version (iOS): 2.5.0

The user has opened an article. This event will be triggered after 3 seconds of viewing the content.

Template placeholder	Example values
CONTENT_ID	The content id of the current content.
CONTENT_NAME	The name of the current content, e.g. the issue name.

PRESENTER_ISSUE_CONTENT_OPENED

New in version (Android): 2.1.0

New in version (iOS): 2.5.0

New in version (Web Player): 3.1.0

The user has opened an issue.

Template placeholder	Example values
CONTENT_ID	The content id of the current content.
CONTENT_NAME	The name of the current content, e.g. the issue name.

Additionally you can use all custom properties which are configured for the issue. The name of the property is used as the placeholder key.

PRESENTER_CONTENT_PAGE_OPENED

New in version (Android): 2.1.0

New in version (iOS): 2.5.0

New in version (Web Player): 3.1.0

The user has opened a page in the currently visible content.

Template placeholder	Example values
CONTENT_ID	The content id of the current content.
CONTENT_NAME	The name of the current content, e.g. the issue name.
PAGE_ID	The id of the page.
PAGE_ALIAS	The alias of the page.
PAGE_LABEL	The label of the page.
PAGE_INDEX	The index of the page.
PAGE_NUMBER	The number of the page.
PAGE_TITLE	The title of the page.
PAGE_SECTION	The section of the page.

PRESENTER_CONTENT_OPEN_FAILED

New in version (Android): 2.5.0

New in version (iOS): 2.5.0

New in version (Web Player): 3.1.0

This event will be tracked if an issue can not be opened, e.g. when its contents are corrupt / invalid.

Template placeholder	Example values
CONTENT_ID	The content id of the current content.
CONTENT_NAME	The name of the current content, e.g. the issue name.

PRESENTER_CONTENT_URL_OPENED

New in version (Android): 2.5.0

New in version (iOS): 2.5.0

New in version (Web Player): 3.1.0

This event will be tracked if a link has been opened from within an issue.

Template placeholder	Example values
URL	The opened url.
CONTENT_ID	The content id of the current content.
CONTENT_NAME	The name of the current content, e.g. the issue name.
PAGE_ID	The id of the page.
PAGE_ALIAS	The alias of the page.
PAGE_LABEL	The label of the page.
PAGE_INDEX	The index of the page.
PAGE_NUMBER	The number of the page.
PAGE_TITLE	The title of the page.
PAGE_SECTION	The section of the page.

4.4.2 Views

APP_BOOKMARKS

New in version (Android): 2.1.0

New in version (iOS): 2.1.0

The bookmarks screen is currently visible.

This event has no template placeholders.

APP_SHARING

New in version (iOS): 2.1.0

The bookmarks screen is currently visible.

This event has no template placeholders.

> **Warning:** This event is not available on Android.

APP_MENU

New in version (Android): 2.1.0

New in version (iOS): 2.1.0

The app menu is currently visible.

This event has no template placeholders.

KIOSK

New in version (Android): 2.1.0

New in version (iOS): 2.1.0

New in version (Web Kiosk): 3.2.2

The kiosk is currently visible.

This event has no template placeholders.

KIOSK_CHANNEL_FEED

New in version (Android): 2.1.0

New in version (iOS): 2.1.0

A channel in the channel feed is currently visible.

Template placeholder	Example values
PUBLICATION_ID	The id of the channel.
PUBLICATION_NAME	The name of the channel.

KIOSK_MY_ISSUES

New in version (Android): 2.1.0

New in version (iOS): 2.1.0

The my issues filter in the kiosk filter is active.

This event has no template placeholders.

KIOSK_PUBLICATION_ALL

New in version (Android): 2.1.0

New in version (iOS): 2.1.0

The all issues filter in the kiosk filter is active.

This event has no template placeholders.

KIOSK_PUBLICATION

New in version (Android): 2.1.0

New in version (iOS): 2.1.0

New in version (Web Kiosk): 3.2.2

A specific publication in the kiosk filter is active.

Template placeholder	Example values
`PUBLICATION_ID`	The id of the channel.
`PUBLICATION_NAME`	The name of the channel.

KIOSK_MANAGE_SUBSCRIPTIONS

New in version (Android): 2.1.0

New in version (iOS): 2.1.0

The subscription management screen is visible.

This event has no template placeholders.

KIOSK_ISSUE_PURCHASE

New in version (Android): 2.1.0

New in version (iOS): 2.1.0

The purchase screen of an issue is visible.

Template placeholder	Example values
ISSUE_ID	The id of the issue.
ISSUE_NAME	The name of the issue.

KIOSK_ISSUE_PREVIEW

New in version (Android): 2.1.0

New in version (iOS): 2.1.0

The preview screen of an issue is visible.

Template placeholder	Example values
ISSUE_ID	The id of the issue.
ISSUE_NAME	The name of the issue.

PRESENTER_PAGE

New in version (Android): 2.5.0

New in version (iOS): 2.5.0

New in version (Web Player): 3.1.0

This event tracks the currently visible page in the content. Only real pages can be tracked using this event. Large pages which are scrollable and have the paging enabled are still only tracked as one page.

Template placeholder	Example values
CONTENT_ID	The content id of the current content.
CONTENT_NAME	The name of the current content, e.g. the issue name.
PAGE_ID	The id of the page.
PAGE_ALIAS	The alias of the page.
PAGE_LABEL	The label of the page.
PAGE_INDEX	The index of the page.
PAGE_NUMBER	The number of the page.
PAGE_TITLE	The title of the page.
PAGE_SECTION	The section of the page.

PRESENTER_CONTENT

New in version (Android): 2.2.0

New in version (iOS): 2.1.0

New in version (Web Player): 3.1.0

An issue is visible.

Template placeholder	Example values
CONTENT_ID	The content id of the current content.
CONTENT_NAME	The name of the current content, e.g. the issue name.

Additionally you can use all custom properties which are configured for the publication and issue. The name of the property is used as the placeholder key.

PRESENTER_CONTENT_TOC

New in version (Android): 2.2.0

New in version (iOS): 2.1.0

New in version (Web Player): 3.1.0

The table of contents of an issue is visible.

Template placeholder	Example values
CONTENT_ID	The content id of the current content.
CONTENT_NAME	The name of the current content, e.g. the issue name.

4.4.3 Purchases

KIOSK_ISSUE_PURCHASED

New in version (Android): 2.1.0

New in version (iOS): 2.1.0

An issue has been purchase.

Template placeholder	Example values
PRODUCT_ID	The product id of the issue.
PRICE	The price of the purchase.
CURRENCY_CODE	The currency code of the price.
ISSUE_ID	The id of the issue.
ISSUE_NAME	The name of the issue.
PUBLICATION_ID	The id of the publication.
PUBLICATION_NAME	The name of the publication.

Additionally you can use all custom properties which are configured for the publication and issue. The name of the property is used as the placeholder key.

KIOSK_SUBSCRIPTION_PURCHASED

New in version (Android): 2.1.0

New in version (iOS): 2.1.0

A subscription has been purchased.

Template placeholder	Example values
`PRODUCT_ID`	The product id of the issue.
`PRICE`	The price of the purchase.
`CURRENCY_CODE`	The currency code of the price.
`SUBSCRIPTION_ID`	The id of the subscription.
`SUBSCRIPTION_NAME`	The name of the subscription.

Additionally you can use all custom properties which are configured for the subscription. The name of the property is used as the placeholder key.

4.4.4 Attributes

HAS_ACTIVE_SUBSCRIPTION

New in version (Android): 2.0.0

New in version (iOS): 2.0.0

This attribute has a value of `true` if the user has an active subscription, otherwise it has the value `false`.

This event has no template placeholders.

HAS_ACTIVE_COUPON_CODE

New in version (Android): 2.0.0

New in version (iOS): 2.0.0

This attribute has a value of `true` if the user has an active coupon code, otherwise it has the value `false`.

This event has no template placeholders.

Warning: This attribute has been renamed to `HAS_ACTIVE_SUBSCRIPTION_CODE` in version 2.3. All configurations have to be manually adjusted to use the new event name.

HAS_ACTIVE_SUBSCRIPTION_CODE

New in version (Android): 2.3.0

New in version (iOS): 2.3.0

This attribute has a value of `true` if the user has an active subscription code, otherwise it has the value `false`.

This event has no template placeholders.

HAS_ACTIVE_TRIAL

New in version (Android): 2.0.0

New in version (iOS): 2.0.0

This attribute has a value of `true` if the user has an active trial, otherwise it has the value `false`.

This event has no template placeholders.

HAS_PURCHASED_ISSUE

New in version (Android): 2.0.0

New in version (iOS): 2.0.0

This attribute has a value of `true` if the user has purchased at least one issue, otherwise it has the value `false`.

This event has no template placeholders.

HAS_BOOKMARKS

New in version (Android): 2.0.0

New in version (iOS): 2.0.0

This attribute has a value of `true` if the user has at least one bookmark, otherwise it has the value `false`.

This event has no template placeholders.

HAS_ENTITLEMENT_LOGIN

New in version (Android): 2.5.0

New in version (iOS): 2.5.0

Is `true` if the user has logged in via entitlement. While the user is logged in the attribute value remains `true`. If the user logs out the attribute value is reset to `false`.

This event has no template placeholders.

CHAPTER 5

Webviews

5.1 General

5.1.1 Support for window.print

New in version 3.7.0.

Apps support the `window.print` JavaScript-API. The standard system print dialog is opened if `window.print` is called.

5.1.2 Support for window.open

Apps support the `window.open` JavaScript-API. Starting with 3.11 links opened via this API will be opened in the internal App Browser (with title bar, status bar and navigation, as a "new window") or the same webview, according to the standard browser specifications.

Starting with version 3.10 Action URLs can also be opened via this API on all platforms.

5.1.3 Cookies

Starting with version 3.13.0 cookies are enabled for all webviews in the Android and iOS apps.

5.1.4 Accessing dynamic resources

Starting with version 5.1 it is possible to access files inside the dynamic resources by using the `resource` scheme. These urls must have the following structure:

resource://dynamic/`<path>`

The path should point towards a file inside the dynamic resources. These paths are resolved based on device's preferred languages. The same goes for all relative paths if the html itself was loaded using this scheme. This allows reducing

the total size of the dynamic resources bundle by putting common files in the default folder and localization related files in their respective folders (e.g. de, en, ...). For further information on dynamic resources and how files are resolved based on the device's preferred languages see the corresponding *page*.

5.2 JavaScript-Interfaces

5.2.1 Using the Purple JavaScript-API

Web content can access special JavaScript-APIs to get information about the app / issue and trigger actions.

There are two ways to access the JavaScript-APIs: **automatic** and **manual** inclusion.

Automatic inclusion (deprecated since 3.9)

Changed in version 3.9.0: Deprecated, replaced with manual inclusion

With automatic inclusion you simply need to have a global `onPurpleLoad` function which will get called after the APIs have been made available.

This however only works on Android and iOS and is not supported in the Web Kiosk. This method is therefore considered deprecated since Purple Release 3.9 with the support of manual inclusion.

Manual inclusion

New in version 3.9.0.

To manually include the JS-APIs html consumers can add any source ending on `scripts/purpleInterface.js` or `scripts/purpleInterface.min.js` to their head element. The web view intercepts this request and returns the `purpleInterface.js`. It is recommended to use the following URL to assure the API works on all platforms including web.

Listing 5.1: Example HTML

```
<html>
  <head>
      <script type="text/javascript" src="https://kiosk.purplemanager.com/scripts/
↪purpleInterface.min.js"></script>
  </head>
  <body></body>
</html>
```

Note: The old automatic injection mechanism is used as fallback strategy.

When manually embedding the purpleInterface.js source, the use of the `onPurpleLoad` function is not mandatory. The purple-Object is available instantly after loading the script.

Hint: If the onPurpleLoad function is used anyways, it needs to be defined prior to the above script tag.

Listing 5.2: Example HTML

```
<html>
  <head>
      <script type="text/javascript">
          function onPurpleLoad() {
              // the global "purple" object is now available
          }
      </script>
      <script type="text/javascript" src="https://kiosk.purplemanager.com/scripts/
↪purpleInterface.min.js"></script>
  </head>
  <body></body>
</html>
```

Listing 5.3: Overview of the JavaScript-Interfaces

```
window.purple = {
    app: {
        ...
    },
    metadata: {
        ...
    },
    storefront: {
        ...
    },
    store: {
        ...
    },
    issue: {
        ...
    },
    state: {
        ...
    },
    tracking: {
        ...
    },
    media: {
        ...
    },
    /**
     * Close the current in app browser or article view
     */
    closeView: function() {
        // Implementation
    }
}
```

Due to the asynchronous nature of the javascript bridge all api calls that return values require a callback function as a parameter. This function is then called by the native implementation with the value as its parameter. A common usage would look like this:

Listing 5.4: Sample usage of callback functions

```
function callbackFunctionForSomething(valueOfSomething) {
    console.log(valueOfSomething);
}

window.purple.getSomething(callbackFunctionForSomething);

// or inline:

window.purple.getSomething(function(valueOfSomething) {
    console.log(valueOfSomething)
});
```

closeView

New in version (Android): 2.6.0

New in version (iOS): 2.6.0

Changed in version: 5.2.0 Can also be used to close the article pager now

The `closeView` method can be used to close certain views such as the in app browser, entitlement and article views. It is only possible to close modal views and not embedded ones.

5.2.2 App

This interface is for app-wide information such as the device's connectivity state.

Note: This interface is not available in Web Kiosk.

Listing 5.5: App JavaScript-Interface

```
window.purple = {
    /**
     * @public
     * @static
     * @namespace AppController
     */
    app: {
        /**
         * Adds a listener for connection state changes.
         * The listener will be called with a ConnectionState object.
         * This listener will also be called with the current state right after
         * calling this method.
         */
        addConnectionStateListener: function (listener) {
            // Implementation
        },
        /**
         * Removes a listener for connection state changes.
         */
```

```
        removeConnectionStateListener: function (listener) {
            // Implementation
        },
        /**
         * Adds a listener for lifecycle changes.
         * The listener will be called with a LifecycleEvent object.
         * This listener will also be called with the current state right after
         * calling this method.
         */
        addLifecycleListener: function (listener) {
            // Implementation
        },
        /**
         * Removes a listener for lifecycle changes.
         */
        removeLifecycleListener: function (listener) {
            // Implementation
        },
        /**
         * Close the onboarding screen. If true is passed as the first parameter
         * the onboarding will be shown again on the next app start.
         */
        closeOnboarding: function (showAgain) {
            // Implementation
        }
    }
}
```

addConnectionStateListener

New in version (Android): 3.3.0

New in version (iOS): 3.4.0

The `addConnectionStateListener` method can be used to register a callback function that gets called when the device changes its connection state. The listener will also be called with the current state when this method is called.

This method takes a single parameter: A callback function that gets called with one parameter, a json object.

This json object will consist of a `state` with the value `ONLINE` and `type` of either `TYPE_3G` during mobile connectivity or `TYPE_WLAN` when it is connected to wi-fi.

```
{
  "state": "ONLINE",
  "type": "TYPE_3G|TYPE_WLAN"
}
```

If the device is offline then there will be only a `state` with the value `OFFLINE`.

```
{
  "state": "OFFLINE"
}
```

removeConnectionStateListener

New in version (Android): 3.3.0

New in version (iOS): 3.4.0

This method removes the listener that was added with `addConnectionStateListener` to stop receiving callbacks.

---

addLifecycleListener

New in version (Android): 3.10.2

New in version (iOS): 3.10.2

The `addLifecycleListener` method can be used to register a callback function that gets called when the webview or the device changes its lifecycle state. The listener will also be called with the current state when this method is called.

This method takes a single parameter: A callback function that gets called with one parameter of type object.

This object will consist of a `type` with the following values

- `STARTED` if the webview appears
- `RESUMED` if the webview is visible and gets focus
- `PAUSED` if the webview is visible but loses focus
- `STOPPED` if the webview disappears

When the app comes to foreground or background and the webview is presented, the callback will also be called with the specific `type`.

```
{
  "type": "STARTED|RESUMED|PAUSED|STOPPED"
}
```

---

removeLifecycleListener

New in version (Android): 3.10.2

New in version (iOS): 3.10.2

This method removes the listener that was added with `addLifecycleListener` to stop receiving callbacks.

---

closeOnboarding

New in version (Android): 3.10.0

New in version (iOS): 3.10.0

Close the onboarding screen. If true is passed as the first parameter the onboarding will be shown again on the next app start.

This API method is only available on the onboarding screen.

See the onboarding documentation for more information about this feature.

5.2.3 App-Browser

This interface can be used to retrieve information about the configuration of the current webview, e.g. if it's displayed modally or embedded, has titlebar and controls.

Listing 5.6: App-Browser JavaScript-Interface

```
window.purple = {
   appBrowser: {
      /**
       * Get the display mode for the current webview.
       */
      getDisplayMode: function (callback) {
          // Implementation
      },
      /**
       * Get the titlebar configuration for the current webview.
       */
      isTitleBarEnabled: function (callback) {
          // Implementation
      },
      /**
       * Get the controls configuration for the current webview.
       */
      isControlsEnabled: function (callback) {
          // Implementation
      },
      /**
       * Get the statusbar configuration for the current webview.
       */
      isStatusBarEnabled: function (callback) {
          // Implementation
      }
   }
}
```

getDisplayMode

New in version (Android): 3.5.0

New in version (iOS): 3.5.0

New in version (Web Kiosk): 3.7.0

New in version (Web Player): 3.5.0

Get the display mode for the current webview.

This method has one parameter, a callback function which will get the display mode value in as a single string parameter.

Values can be `embedded` or `modal`.

---

isTitleBarEnabled

New in version (Android): 3.5.0

New in version (iOS): 3.5.0

New in version (Web Kiosk): 3.7.0

New in version (Web Player): 3.5.0

Get the titlebar configuration for the current webview.

This method has one parameter, a callback function which will get a boolean value in as a single parameter.

---

isControlsEnabled

New in version (Android): 3.5.0

New in version (iOS): 3.5.0

New in version (Web Kiosk): 3.7.0

New in version (Web Player): 3.5.0

Get the controls configuration for the current webview.

This method has one parameter, a callback function which will get a boolean value in as a single parameter.

---

isStatusBarEnabled

New in version (Android): 3.10.0

New in version (iOS): 3.10.0

New in version (Web Kiosk): 3.10.0

New in version (Web Player): 3.10.0

Get the statusbar configuration for the current webview.

This method has one parameter, a callback function which will get a boolean value in as a single parameter.

5.2.4 Metadata

Metadata / information about the app and issue can be accessed through this javascript interface.

Listing 5.7: Metadata JavaScript-Interface

```
window.purple = {
    /**
     * @public
     * @static
     * @namespace MetaDataController
     */
    metadata: {
        /**
         * Get metadata values by key. A callback function with a single parameter is␣
↪required that will be called
         * with the value for the given key.
         *
         * @param {string} key          the metadata key
         * @param {Function} callback   the callback for the value
         */
        getMetadata: function (key, callback) {
            // Implementation
        }
    }
}
```

---

getMetadata

New in version (Android): 2.4.0

New in version (iOS): 2.6.0

New in version (Web Kiosk): 3.7.0

New in version (Web Player): 2.6.0

This method returns the value for a given key. The value will be provided via the *callback* method.

The following keys are available:

---

app_id

New in version (Android): 2.4.0

New in version (iOS): 2.4.0

New in version (Web Kiosk): 3.7.0

New in version (Web Player): 2.6.0

New in version (Composer): 3.1.0

Changed in version: 2.6.0 now globally available in all web views

The id of the app in the Purple Manager.

On macOS this always returns “Preview Publication”.

Available contexts

- Entitlement HTML Login
- Dynamic HTML-Content
- In-App-Browser
- Storytelling Content

---

app_version

New in version (Android): 2.6.0

New in version (iOS): 2.6.0

New in version (Composer): 3.1.0

The version of the app.

Available contexts

- Entitlement HTML Login
- Dynamic HTML-Content
- In-App-Browser
- Storytelling Content

---

preview_app

New in version (Android): 2.6.0

New in version (iOS): 2.6.0

New in version (Web Kiosk): 3.7.0

New in version (Web Player): 2.6.0

New in version (Composer): 3.1.0

Boolean value indicating if the app is a preview or release app.

On macOS this always returns `true`.

Available contexts

- Entitlement HTML Login
- Dynamic HTML-Content
- In-App-Browser
- Storytelling Content

---

device_id

New in version (Android): 2.6.0

New in version (iOS): 2.6.0

The unique id of the device. Used for communication with the Purple Manager.

Available contexts

- Entitlement HTML Login
- Dynamic HTML-Content
- In-App-Browser
- Storytelling Content

---

device_model

New in version (Android): 2.6.0

New in version (iOS): 2.6.0

New in version (Composer): 3.1.0

The model of the device.

Available contexts

- Entitlement HTML Login
- Dynamic HTML-Content
- In-App-Browser
- Storytelling Content

---

device_os

New in version (Android): 2.6.0

New in version (iOS): 2.6.0

New in version (Web Kiosk): 3.7.0

New in version (Web Player): 2.6.0

New in version (Composer): 3.1.0

The os version of the device.

Available contexts

- Entitlement HTML Login
- Dynamic HTML-Content
- In-App-Browser
- Storytelling Content

platform

New in version (Android): 2.6.0

New in version (iOS): 2.6.0

New in version (Web Kiosk): 3.7.0

New in version (Web Player): 2.6.0

New in version (Composer): 3.1.0

The platform (`android`, `kindle`, `ios`, `web` or ``macOS`) on which this app is running on.

Available contexts

- Entitlement HTML Login
- Dynamic HTML-Content
- In-App-Browser
- Storytelling Content

locale

New in version (Android): 2.6.0

New in version (iOS): 2.6.0

New in version (Web Kiosk): 3.7.0

New in version (Web Player): 2.6.0

New in version (Composer): 3.1.0

The current locale of the system.

Available contexts

- Entitlement HTML Login
- Dynamic HTML-Content
- In-App-Browser
- Storytelling Content

manager_base_url

New in version (Android): 2.6.0

New in version (iOS): 2.6.0

New in version (Web Kiosk): 3.7.0

New in version (Web Player): 2.6.0

The base url for communicating with the delivery service of the Purple Manager.

Available contexts

- Entitlement HTML Login
- Dynamic HTML-Content
- In-App-Browser
- Storytelling Content

---

push_registration_token

New in version (Android): 3.0.0

New in version (iOS): 3.0.0

The push registration token. This can be used to send pushes to the app.

Available contexts

- Entitlement HTML Login
- Dynamic HTML-Content
- In-App-Browser
- Storytelling Content

---

entitlement_login

New in version (Android): 2.4.0

New in version (iOS): 2.4.0

New in version (Web Kiosk): 3.7.0

New in version (Web Player): 2.6.0

Changed in version: 2.6.0 now globally available in all web views

The entitlement username of the user if he is logged in.

Available contexts

- Dynamic HTML-Content
- In-App-Browser
- Storytelling Content

entitlement_token

New in version (Android): 2.4.0

New in version (iOS): 2.4.0

New in version (Web Kiosk): 3.7.0

New in version (Web Player): 2.6.0

Changed in version: 2.6.0 now globally available in all web views

The entitlement access token of the user if he is logged in.

Available contexts

- Entitlement HTML Login
- Dynamic HTML-Content
- In-App-Browser
- Storytelling Content

entitlement_forced_login_enabled

New in version (Android): 2.6.0

New in version (iOS): 2.6.0

The configuration parameter for forced entitlement login on app start.

Available contexts

- Entitlement HTML Login
- Dynamic HTML-Content
- In-App-Browser
- Storytelling Content

entitlement_login_mode

New in version (Android): 2.6.0

New in version (iOS): 2.6.0

Can be either `login` or `relogin` indicating the current mode / reason why the login screen has been opened. `relogin` will be returned if the server responded with an error that the access token is invalid during subscription validation.

Available contexts

- Html-Entitlement Login

entitlement_mode

New in version (Android): 3.11.0

New in version (iOS): 3.11.0

Can be either `entitlement`, `oauth` or `none` indicating the mode of the first entitlement server that is configured for the app.

Available contexts

- Entitlement HTML Login
- Dynamic HTML-Content
- In-App-Browser
- Storytelling Content

entitlement_refresh_token

New in version (Android): 3.11.0

New in version (iOS): 3.11.0

The refresh token that is being used to request a new access token for the entitlement api calls. This value is only available if oauth is being used as entitlement server.

Available contexts

- Entitlement HTML Login
- Dynamic HTML-Content
- In-App-Browser
- Storytelling Content

issue_id

New in version (Android): 2.4.0

New in version (iOS): 2.4.0

New in version (Web Player): 2.6.0

New in version (Composer): 3.1.0

Changed in version: Android/iOS 3.17: Property now available in dynamic HTML and In-App-Browser (see below)

The id of the currently viewed issue.

Available contexts

- Storytelling Content
- Dynamic HTML-Content (when opened via an ST-Action)
- In-App-Browser (when opened via an ST-Action)

---

issue_name

New in version (Android): 2.4.0

New in version (iOS): 2.4.0

New in version (Web Player): 2.6.0

New in version (Composer): 3.1.0

Changed in version: Android/iOS 3.17: Property now available in dynamic HTML and In-App-Browser (see below)

The name of the currently viewed issue.

In the Composer Preview on macOS this always returns “Preview Publication”.

Available contexts

- Storytelling Content
- Dynamic HTML-Content (when opened via an ST-Action)
- In-App-Browser (when opened via an ST-Action)

---

issue_alias

New in version (Android): 3.17.0

New in version (iOS): 3.17.0

The alias of the currently viewed issue.

Available contexts

- Storytelling Content
- Dynamic HTML-Content (when opened via an ST-Action)
- In-App-Browser (when opened via an ST-Action)

---

publication_id

New in version (Android): 2.4.0

New in version (iOS): 2.4.0

New in version (Web Player): 2.6.0

New in version (Composer): 3.1.0

Changed in version: Android/iOS 3.17: Property now available in dynamic HTML and In-App-Browser (see below)

The id of the publication of the currently viewed issue.

In the Composer Preview on macOS this always returns “Preview Publication”.

Available contexts

- Storytelling Content
- Dynamic HTML-Content (when opened via an ST-Action)
- In-App-Browser (when opened via an ST-Action)

---

publication_name

New in version (Android): 2.4.0

New in version (iOS): 2.4.0

New in version (Web Player): 2.6.0

New in version (Composer): 3.1.0

Changed in version: Android/iOS 3.17: Property now available in dynamic HTML and In-App-Browser (see below)

The name of the publication of the currently viewed issue.

On macOS this always returns “Preview Publication”.

Available contexts

- Storytelling Content
- Dynamic HTML-Content (when opened via an ST-Action)
- In-App-Browser (when opened via an ST-Action)

---

page_id

New in version (Android): 2.4.0

New in version (iOS): 2.4.0

New in version (Web Player): 2.6.0

New in version (Composer): 3.1.0

The id of the currently viewed content page.

Available contexts

- Storytelling Content

---

page_alias

New in version (Android): 2.4.0

New in version (iOS): 2.4.0

New in version (Composer): 3.1.0

The alias of the currently viewed content page.

Available contexts

- Storytelling Content

---

page_title

New in version (Android): 2.4.0

New in version (iOS): 2.4.0

New in version (Web Player): 2.6.0

New in version (Composer): 3.1.0

The title of the currently viewed content page.

Available contexts

- Storytelling Content

page_index

New in version (Android): 2.4.0

New in version (iOS): 2.4.0

New in version (Web Player): 2.6.0

New in version (Composer): 3.1.0

The index of the currently viewed content page.

Available contexts

- Storytelling Content

page_filename

New in version (Android): 2.4.0

New in version (iOS): 2.4.0

New in version (Web Player): 2.6.0

New in version (Composer): 3.1.0

The filename of the current page (not the webview page, but the storytelling content page)

Available contexts

- Storytelling Content

onboarding_mode

New in version (Android): 3.10.1

New in version (iOS): 3.10.1

The mode for the onboarding screen. Can be `appstart` when opened during app start or `manual` when opened via action url.

Available contexts

- HTML onboarding screen

5.2.5 Storefront

Web content can access storefront data through a javascript interface.

Note: This interface is not available in Composer Native Preview.

Listing 5.8: Storefront JavaScript-Interface

```
window.purple = {
    /**
     * @public
     * @static
     * @namespace StorefrontController
     */
    storefront: {
        /**
         * Get subscriptions.
         *
         * @param {Function} callback     the callback for the subscriptions
         */
        getSubscriptions: function (callback) {
            // Implementation
        },
        /**
         * Gets a list of all publications. Callback will be called with a
         * JSONArray of Publication objects.
         */
        getPublications: function (callback) {
            // Implementation
        },
        /**
         * Gets a list of all issues for the publication with the given
         * publicationId. Callback will be called with a JSONArray of Issue
         * objects.
         */
        getIssues: function (publicationId, callback) {
            // Implementation
        },
        /**
         * Gets a list of all issue states for the given issueIds. Callback
         * will be called with a JSONArray of IssueState objects without a
         * progress value.
         */
        getIssueStates: function (issueIds, callback) {
            // Implementation
        },
        /**
         * Gets the issue object for the given issueId. The callback function
         * will be called with the corresponding Issue object. If the issueId
         * is the id of a preview issue, then the corresponding parent issue
         * will be returned.
         */
        getIssueById: function (issueId, callback) {
            // Implementation
        },
        /**
         * Starts the download of the issue with the given issueId.
```

```
         * This can also be a preview issue.
         */
        startDownload: function (issueId) {
            // Implementation
        },
        /**
         * Pauses the download of the issue with the given issueId.
         * This can also be a preview issue.
         */
        pauseDownload: function (issueId) {
            // Implementation
        },
        /**
         * Deletes the content of the issue with the given issueId.
         * This includes the preview content and temporary downloaded data.
         * The callback will be called with the current IssueState object.
         */
        deleteIssue: function (issueId, callback) {
            // Implementation
        },
        /**
         * Adds a listener for issue state changes.
         * The listener will be called with an IssueState object with a
         * progress value.
         */
        addIssueStateListener: function (listener) {
            // Implementation
        },
        /**
         * Removes a listener for issue state changes.
         */
        removeIssueStateListener: function (listener) {
            // Implementation
        },
        /**
         * Loads the new storefront.
         * The callback will be called with the StorefrontUpdateResult object.
         */
        updateStorefront: function (callback) {
            // Implementation
        },
        /**
         * Adds a listener for newsstand and newsfeed changes.
         */
        addUpdateListener: function (listener) {
            // Implementation
        },
        /**
         * Removes a listener for newsstand and newsfeed changes.
         */
        removeUpdateListener: function (listener) {
            // Implementation
        }
        /**
         * Returns the base url for the files of the given issueId.
         */
        getIssueBaseUrl: function (issueId, callback) {
            // Implementation
```

```
        },
        /**
         * Returns the pages information for the given issueId.
         */
        getIssuePages: function (issueId, callback) {
            // Implementation
        },
        /**
         * Returns the toc information for the given issueId.
         */
        getIssueToc: function (issueId, callback) {
            // Implementation
        },
        /**
         * Open a list of articles in a pager.
         */
        openArticles: function (articleIds, initialArticleId, callback) {
            // Implementation
        },
        /**
         * Get a list of all categories. The callback will be called with a
         * JSONArray of Category objects.
         */
        getCategories: function (callback) {
            // Implementation
        }
    }
}
```

getSubscriptions

New in version (Android): 3.0.0

New in version (iOS): 3.0.0

Changed in version: 3.4.0 moved from :code:‘kiosk‘ to :code:‘storefront‘ API. The method in the :code:‘kiosk‘ is still available but deprecated.

Subscriptions can be accessed through the `getSubscriptions` method. It takes one parameter, a callback function, which is called with an array of subscriptions.

Listing 5.9: Subscription model

```
{
  "name": "One Month Subscription",
  "productId": "com.sprylab.onemonth",
  "duration": "one_month",
  "hidden": false,
  "unlocksAllContentDuringPeriod": true,
  "index": 1,
  "publicationIds": ["aabbcc", "1233456"],
  "formattedPrice": "13.37€",
```

```
  "price": 13.37,
  "currency": "EUR",
  "properties": [
      {
          "name": "testproperty",
          "value": "testvalue"
      }
  ],
  "state": "NONE|PURCHASING|VALIDATING|PURCHASED"
}
```

---

getPublications

New in version (Android): 3.4.0

New in version (iOS): 3.4.0

New in version (Web Kiosk): 3.7.0

This method lists all publications in the storefront. It takes one parameter, a callback function, which is called with an array of publications.

Listing 5.10: Publications model

```
{
  "publicationId": "aabbcc",
  "displayName": "Publication A",
  "displayDescription": "Fancy publication description",
  "properties": [
    {
      "name": "testproperty",
      "value": "testvalue"
    }
  ],
  "index": 0,
  "type": "KIOSK|CHANNEL",
  "thumbnails": {
    "default": "url",
    "kind1": "url"
  }
}
```

---

getIssues

New in version (Android): 3.4.0

New in version (iOS): 3.4.0

New in version (Web Kiosk): 3.7.0

Changed in version: Android/iOS 3.15: The issue alias is now also available, 4.0: :code:‘LOCKED‘ issues are not returned anymore, 5.1: Added :code:‘publicationId‘, 5.2: Added :code:‘externalIssueId‘

This method needs to be called to obtain a list of issues for a specific publication. It takes two parameters: A `publicationId` and a callback function which then will be called with an array of issues. The `publicationId` can be acquired from a publication model that is obtained through the getPublications method.

Listing 5.11: Issue model

```
{
  "issueId": "aabbcc",
  "publicationId": "ddeeff",
  "alias": "alias2",
  "displayName": "Issue A",
  "displayDescription": "Fancy issue description",
  "properties": [
    {
      "name": "testproperty",
      "value": "testvalue"
    }
  ],
  "index": 0,
  "pubDate": 123,
  "contentLength": 1337,
  "numberOfPages": 42,
  "comingSoon": true,
  "previewIssue": {
    "id": "ddeeff",
    "contentLength": 1337,
    "numberOfPages": 42
  },
  "productId": "com.sprylab.issue1",
  "thumbnails": {
    "default": "url",
    "kind1": "url"
  },
  "tags": ["tag1", "tag2", "tag3"],
  "categories": ["categoryId1", "categoryId2"]
}
```

The `pubDate` is a UNIX timestamp in ms. If the issue has a preview issue then `previewIssue` is a jsonObject with the preview issue's id, its content size and the number of pages otherwise it is `null`. The `productId` can be used to purchase the issue through the `store` api. It can be `null` if the issue is not purchasable through in app payments.

---

getIssueStates

New in version (Android): 3.4.0

New in version (iOS): 3.4.0

New in version (Web Kiosk): 3.7.0

Changed in version: 4.0: removed :code:‘LOCKED‘ and :code:‘UPDATE‘ states, added updateAvailable to the state model and states for progressive loading, 5.0: removed :code:‘INSTALLING‘ state

To obtain the state of specific issues this method can be used. It takes two parameters. The first is an array of issue ids and the second a callback function which returns an array of issue state objects for the requested issue ids.

Listing 5.12: Issue state model

```
{
  "issueId": "aabbcc",
  "state": "<STATE>",
  "updateAvailable": <true|false>
}
```

Certain states are only available depending whether progressive loading is enabled or not. Note that issues that are hidden through entitlement are not returned by getIssues anymore until they are unlocked. The following table describes the possible issue states.

State	Description	PL	PK
LOCKED	locked through entitlement -> not visible for the user	off	<4.0
COMING_SOON	coming soon enabled -> visible, but not downloadable	any	>=3.4.0
PURCHASABLE	not purchased	any	>=3.4.0
AVAILABLE	download possible -> purchased, or no paid content	any	>=3.4.0
INDEXING	minimal required	on	>=4.0
PARTIALLY_INSTALLED_DOWNLOADING	ready to be displayed, still loading and not finished yet	on	>=4.0
PARTIALLY_INSTALLED_PAUSED	ready to be displayed, not loading and not finished yet	on	>=4.0
DOWNLOAD_PAUSED	download started and paused	off	>=3.4.0
DOWNLOADING	downloading (and extracting on >= PK 5.0)	off	>=3.4.0
INSTALLING	extracting, post processing	off	<5.0
INSTALLED	downloaded, extracted, available for reading	any	>=3.4.0
UPDATE	downloaded and new version available (=local version does not match remote version)	off	<4.0

---

getIssueById

New in version (Android): 5.1

New in version (iOS): 5.1

This method can be used to request information for a single issue with a given issueId. The callback function will be called with the corresponding Issue Object. See getIssues for the structure of such an object. When the passed issueId

is the id of a preview issue then the corresponding parent issue is returned instead.

---

startDownload

New in version (Android): 3.4.0

New in version (iOS): 3.4.0

With this method it is possible to start the download of an issue with the given `issueId`.

---

pauseDownload

New in version (Android): 3.4.0

New in version (iOS): 3.4.0

With this method it is possible to pause the download of the issue with the given `issueId`.

---

deleteIssue

New in version (Android): 3.4.0

New in version (iOS): 3.4.0

This method allows the deletion of an issue. It takes two parameters. The first is the `issueId` of the issue that will be removed and the other is a callback function that will be called with an issue state object after the delete process completed.

---

addIssueStateListener

New in version (Android): 3.4.0

New in version (iOS): 3.4.0

Changed in version: 5.0 changed behavior of the progress in the :code:‘DOWNLOADING‘ state

The `addIssueStateListener` method allows the registration of a listener that will be called each time when the state of an issue changes. This method takes a single parameter which is a callback function. It will be called with issue state objects which contain an additional `progress` value. This `progress` can be a value between 0 and 100.

For a description of the states see the table in `getIssueStates`.

Listing 5.13: Issue state model (with progress, for possible states see getIssueStates)

```
{
  "issueId": "aabbcc",
  "state": "<STATE>",
  "updateAvailable": <true|false>,
  "progress": 0
}
```

---

removeIssueStateListener

New in version (Android): 3.4.0

New in version (iOS): 3.4.0

This method removes a listener that was set with `addIssueStateListener` to stop receiving issue state updates.

---

updateStorefront

New in version (Android): 3.4.0

New in version (iOS): 3.4.0

New in version (Web Kiosk): 3.7.0

This method starts a synchronization of the storefront with the Purple Manager. The result of this process will then be called on the given callback function.

For a successful synchronization it will be a simple json object:

```
{
  "success": true
}
```

For failures it will be a json object which may contain an error code:

```
{
  "success": false,
  "error_code": "[OFFLINE|UNKNOWN]"
}
```

---

addUpdateListener

New in version (Android): 3.11.0

New in version (iOS): 3.11.0

Changed in version: Android/iOS 3.15: Callback now has a parameter (see below)

The `addUpdateListener` method allows the registration of a listener that will be called each time the kiosk has been changed due to a sync or login change. This method takes a single parameter which is a callback function which

itself takes one parameter.

This parameter will be a JSON object with the following format:

```
{
   type: "PARTIAL|FULL|LOGIN|LOGOUT",
   success: true
}
```

Or for errors:

```
{
   type: "PARTIAL|FULL|LOGIN|LOGOUT",
   success: false,
   error_code: "OFFLINE|UNKNOWN"
}
```

---

removeUpdateListener

New in version (Android): 3.11.0

New in version (iOS): 3.11.0

This method removes a listener that was previously added with `addUpdateListener` to stop receiving kiosk updates.

---

getIssueBaseUrl

New in version (Android): 3.13.0

New in version (iOS): 3.13.0

This method can be used to retrieve the base url for the given issueId. If the issue is not downloaded yet, the callback is called with `null`. With this base url it is possible to request files of an issue. Requests to the `pages.xml` and `TOC.xml` will return `404 Not Found`. To request the contents of these files, the `storefront.getIssuePages()` and `storefront.getIssueToc()` methods should be used.

---

getIssuePages

New in version (Android): 3.13.0

New in version (iOS): 3.13.0

This method works the same as issue.getPages() except that it takes an issueId as parameter and returns the data for that issue. See *issue.getPages()* for details regarding the page model.

---

getIssueToc

New in version (Android): 3.13.0

New in version (iOS): 3.13.0

This method works the same as issue.getToc() except that it takes an issueId as parameter and returns the data for that issue. See *issue.getToc()* for details regarding the toc model.

---

openArticles

New in version (Android): 3.14.0

New in version (iOS): 3.14.0

Opens a list of articles by their issue ID in a native pager. Invalid issues, e.g. paid issues, non-channel issues, will be filtered. If no issues remain to be opened, the callback will be called with the *error_code NO_ISSUES_FOUND*.

The second parameter may be an issue ID from the list of issue IDs which will be the initially opened issue. If the issue ID is invalid, the first valid issue will be shown.

When the articles have been successfully opened the callback function will be called with a simple json object:

```
{
  "success": true
}
```

For failures it will be a json object which may contain an error code:

```
{
  "success": false,
  "error_code": "[NO_ISSUES_FOUND|UNKNOWN]"
}
```

---

getCategories

New in version (Android): 3.15.0

New in version (iOS): 3.15.0

Gets a list of all categories. The callback will be called with a JSONArray like the following example:

```
[
  {
    "id": "categoryId1",
    "name": "Category name",
    "thumbnailURL": "http://",
    "properties": { "name": "value" },
    "categories": [
      {
        "id": "categoryId2",
        "name": "Category name 2",
        "thumbnailURL": "http://",
        "properties": { "name": "value" },
        "categories": []
      }
    ]
```

```
  },
  {
    "id": "categoryId3",
    "name": "Category name",
    "thumbnailURL": "http://",
    "properties": { "name": "value" },
    "categories": []
  }
]
```

5.2.6 Store

It is also possible to start purchases and manage the subscription codes through a javascript interface.

Note: This interface is not available in Web Kiosk and Composer Native Preview.

Listing 5.14: Store JavaScript-Interface

```
window.purple = {
    /**
     * @public
     * @static
     * @namespace StoreController
     */
    store: {
        /**
         * Purchase a product
         *
         * @param {string} productId
         * @param {Function} callback
         */
        purchase: function (productId, callback) {
            // Implementation
        },
        /**
         * Subscribe to a subscription
         *
         * @param {string} productId
         * @param {Function} callback
         */
        subscribe: function (productId, callback) {
            // Implementation
        },
        /**
         * Restores purchases on device. Only available on iOS
         * (Android will call callback immediately).
         *
         * @param {Function} callback
         */
        restorePurchases: function (callback) {
            // Implementation
        },
        /**
         * Set a function as a listener which should be called when
```

```
         * the purchase state of a subscription product did change.
         * @param {Function} listener
         */
        setPurchaseStateListener: function (listener) {
            // Implementation
        },
        /**
         * Get the current subscription codes
         *
         * @param {Function} callback
         */
        getSubscriptionCodes: function (callback) {
            // Implementation
        },
        /**
         * Add subscription codes
         *
         * @param {String[]} codes
         * @param {Function} callback
         */
        addSubscriptionCodes: function (codes, callback) {
            // Implementation
        },
        /**
         * Remove the subscription codes
         *
         * @param {String[]} codes
         * @param {Function} callback
         */
        removeSubscriptionCodes: function (codes, callback) {
            // Implementation
        },
        /**
         * Gets the price information for the given productIds.
         * The callback will be called with a JSONArray of ProductInfo objects.
         */
        getPrices: function (productIds, callback) {
            // Implementation
        }
    }
}
```

purchase

New in version (Android): 3.0.0

New in version (iOS): 3.0.0

The `purchase` method can be used to purchase a single product, e.g. an issue.

It takes two parameters: the product id and a callback function.

The callback function gets called with one parameter.

For successful purchases it will be a simple json object:

```
{
  "success": true
}
```

For failures it will be a json object which may contain an error code:

```
{
  "success": false,
  "error_code": "CANCELLED"
}
```

---

subscribe

New in version (Android): 3.0.0

New in version (iOS): 3.0.0

The `subscribe` method can be used to purchase a subscription.

It takes two parameters: the product id and a callback function.

The callback function gets called with one parameter.

For successful purchases it will be a simple json object:

```
{
  "success": true
}
```

For failures it will be a json object which may contain an error code:

```
{
  "success": false,
  "error_code": "CANCELLED"
}
```

---

restorePurchases

New in version (Android): 3.0.0

New in version (iOS): 3.0.0

Previously purchased products and subscriptions can be restored using the `restorePurchases` method.

It takes one parameter, a callback function, which gets called when the restore has finished.

The callback function gets called with one parameter: a success or failure result object. See purchase / subscribe for details about this object.

Note: This call is only available on iOS. It will do nothing on Android and call the callback function immediately with a success-object.

---

setPurchaseStateListener

New in version (Android): 3.0.0

New in version (iOS): 3.0.0

Set a function as a listener which will be called when the purchase state of a subscription product changed.

Note: This call is only available on iOS. It will do nothing on Android.

getSubscriptionCodes

New in version (Android): 3.0.0

New in version (iOS): 3.0.0

Get the current subscription codes.

It takes one parameter, a callback function, which gets called with the subscription codes in a string array as the only parameter.

addSubscriptionCodes

New in version (Android): 3.0.0

New in version (iOS): 3.0.0

Add and activate (multiple) subscription codes.

It takes two parameters: the codes as a string array and a callback function, which gets called with a success or failure result object. See purchase / subscribe for details about this object.

removeSubscriptionCodes

New in version (Android): 3.0.0

New in version (iOS): 3.0.0

Remove and deactivate (multiple) subscription codes.

It takes two parameters: the codes as a string array and a callback function, which gets called with a success or failure result object. See purchase / subscribe for details about this object.

getPrices

New in version (Android): 3.3.0

New in version (iOS): 3.4.0

This method requests the price information for given product ids. It takes two parameters: The first is an array of product ids and the second a callback method that will be called with an array of product info objects.

```
{
  "productId": "some.product.id",
  "formattedPrice": "13.37€",
  "price": 13.37,
  "currency": "EUR"
}
```

5.2.7 Issue

This API can be used to retrieve information (e.g. pages and toc from the `pages.xml` and `TOC.xml`) of the current issue.

Listing 5.15: Issue JavaScript-Interface

```
window.purple = {
    issue: {
        getPages: function (callback) {
            // Implementation
        },
        getToc: function (callback) {
            // Implementation
        }
    }
}
```

getPages

New in version (Android): 3.3.0

New in version (iOS): 3.3.0

New in version (Web Player): 3.2.0

New in version (Composer): 3.1.0

Changed in version: 3.13 :code:‘targetURL‘ has been added to the response. :code:‘targetURL‘ and :code:‘thumbnailURL‘ do not have a scheme such as :code:‘pkmedia://‘ anymore and are now relative to the content root.

The `getPages` method can be used to retrieve all pages.

It takes one parameter: a callback function.

The callback function gets called with one parameter: an array of page model objects.

Listing 5.16: Page model

```
{
  "id": "",
  "pageIndex": 1,
  "pageNumber": 1,
  "pageLabel": "Seite 1",
  "title": "Seite 1",
  "shortTitle": "Seite 1",
  "alias": "Seite 1",
  "showPurchaseSuggestion": true,
  "placeholder": false,
  "excludeFromPaging": true,
  "targetURL": "page-id.stxml",
  "thumbnailURL": "thumbs/thumb-page123.jpg",
  "sharingEnabled": true,
  "sharingText": "Text",
  "sharingURL": "http://example.com",
  "customData": "tag1,tag2"
}
```

getToc

New in version (Android): 3.4.0

New in version (iOS): 3.4.0

New in version (Web Player): 3.3.0

New in version (Composer): TODO

Changed in version: 3.13 :code:‘thumbnailURL‘ does not have a scheme such as :code:‘pkmedia://‘ anymore and is now relative to the content root.

The `getToc` method can be used to retrieve all toc pages.

It takes one parameter: a callback function.

The callback function gets called with one parameter: an array of toc page model objects.

Listing 5.17: Toc page model

```
{
  "pageId": "<page-id>",
  "pageAlias": "Page Alias",
  "title": "Title 123",
  "section": "Section",
  "shortTitle": "Old Content Short-Title",
  "teaser": "Old Content Only",
  "thumbnailURL": "thumbs/thumb-page123.jpg"
}
```

5.2.8 State

This API can be used to store custom state for usage in different webviews. It can only store string values.

Listing 5.18: State JavaScript-Interface

```
window.purple = {
    state: {
        setState: function (key, value) {
            // Implementation
        },
        getState: function (key, callback) {
            // Implementation
        }
    }
}
```

Note: On macOS the state is stored on a per document bases inside the users defaults. This means that the state data is not part of the Purple-Project files.

setState

New in version (Android): 3.3.0

New in version (iOS): 3.3.0

New in version (Web Kiosk): 3.7.0

New in version (Web Player): 3.1.4

New in version (Composer): 3.1.0

The `setState` method can be used to store a string value for a string key.

It takes two parameters: the key and a value.

If the value is `null` the key gets deleted.

Hint: The key is case-sensitive.

getState

New in version (Android): 3.3.0

New in version (iOS): 3.3.0

New in version (Web Kiosk): 3.7.0

New in version (Web Player): 3.1.4

New in version (Composer): 3.1.0

The `getState` method can be used to retrieve a string value for a string key.

It takes two parameters: the key and a callback function.

The callback function gets called with two parameters: the key and the value. If there is no value for the given key, the value will be `null`.

Hint: The key is case-sensitive.

5.2.9 Tracking

This API can be used to track custom events in web views (e.g. purchase over an external entitlement or custom view like read mode). There are three different types of events:

1. Actions
2. Views
3. Purchases

The events are forwarded to the app's enabled *tracking services*. The same *configuration mechanism* like in the apps is used.

Note: User Attributes are currently not supported.

Note: All the keys in the optionalParams object will be automatically converted to upper case.

Listing 5.19: Tracking JavaScript-Interface

```
window.purple = {
    tracking: {
    /**
    * Track an action, e.g. a tap on a button.
    *
    * @param {string} key the event name
    * @param {Object} [optionalParams] can be sent with each event if the tracking␣
↪service supports this.
    * Every key will be included in the event. The values can contain all␣
↪placeholders supported for the event
    * and will be evaluated (see app tracking) when sending the event to the service.
    * @param {Function} [callback] is called when the event has been tracked
    */
    trackAction: function (key, optionalParams, callback) {
        },
    /**
    * Track a view, e.g. a currently shown screen.
    *
    * @param {string} key the key of the screen event
    * @param {Object} [optionalParams] can be sent with each event if the tracking␣
↪service supports this.
    * Every key will be included in the event. The values can contain all␣
↪placeholders supported for the event
    * and will be evaluated (see app tracking) when sending the event to the service.
    * @param {Function} [callback] is called when the event has been tracked
```

```
    */
    trackView: function (key, optionalParams, callback) {
        },
    /**
    * Track a purchase.
    *
    * @param {string} key the purchase event key
    * @param {string} productId
    * @param {number} price
    * @param {string} currencyCode
    * @param {string} transactionId
    * @param {Object} [optionalParams] can be sent with each event if the tracking service supports this.
    * Every key will be included in the event. The values can contain all placeholders supported for the event
    * and will be evaluated (see app tracking) when sending the event to the service.
    * @param {Function} [callback] is called when the event has been tracked
    */
    trackPurchase: function (key, productId, price, currencyCode, transactionId, optionalParams, callback) {
        }
    }
}
```

Note: The `optionalParams` (key-value pairs) can be sent with each event if the tracking service supports this. Every key will be included in the event. The values can contain all placeholders supported for the event and will be evaluated (see *app tracking*) when sending the event to the service.

---

trackAction

New in version (Android): 3.10.3

New in version (iOS): 3.10.5

New in version (Web Kiosk): 3.11.0

New in version (Web Player): 3.11.0

Changed in version: 3.14.0 added callback

The `trackAction` method can be used to track a custom action event.

It takes two parameters: key and optionalParams. Starting with 3.14.0 the method takes a callback function which is called when the event has been tracked.

---

trackView

New in version (Android): 3.10.3

New in version (iOS): 3.10.5

New in version (Web Kiosk): 3.11.0

New in version (Web Player): 3.11.0

Changed in version: 3.14.0 added callback

The `trackView` method can be used to track a custom view event.

It takes two parameters: key and optionalParams. Starting with 3.14.0 the method takes a callback function which is called when the event has been tracked.

---

trackPurchase

New in version (Android): 3.10.3

New in version (iOS): 3.10.5

New in version (Web Kiosk): 3.11.0

New in version (Web Player): 3.11.0

Changed in version: 3.14.0 added callback

The `trackPurchase` method can be used to track a custom purchase event.

It takes six parameters: key, productId, price, currencyCode, transactionId and optionalParams. Starting with 3.14.0 the method takes a callback function which is called when the event has been tracked.

5.2.10 Media

This API can be used to play remote audio streams and files. The audio will continue playing even if the user puts the app to the background.

---

background_mode_audio_enabled

New in version (iOS): 3.11.0

Type: `boolean`

Default: `false`

This property enables the background audio mode for iOS apps. This allows the playback to continue when the app is moved to the background.

Note that enabling this property also means that other audio applications get interrupted.

Listing 5.20: Media JavaScript-Interface

```
window.purple = {
  media: {
    startAudio: function (displayName, url) {
        // Implementation
    },
    pauseAudio: function () {
        // Implementation
    },
    resumeAudio: function () {
        // Implementation
    },
    stopAudio: function () {
```

```
        // Implementation
    },
    seekTo: function(time) {
        // Implementation
    },
    setPlaybackRate: function (rate) {
        // Implementation
    },
    getPlaybackRate: function (callback) {
        // Implementation
    },
    addStatusListener: function (statusListener) {
        // Implementation
    },
    removeStatusListener: function (statusListener) {
        // Implementation
    },
    addProgressListener: function (progressListener) {
       // Implementation
    },
    removeProgressListener: function (progressListener) {
        // Implementation
    }
  }
};
```

---

startAudio

New in version (Android): 3.11.0

New in version (iOS): 3.11.0

Starts a remote audio stream or file. This method takes two parameters: a display name used for notification and player UI and the URL to play.

---

pauseAudio

New in version (Android): 3.11.0

New in version (iOS): 3.11.0

Pauses the current audio playback.

---

resumeAudio

New in version (Android): 3.11.0

New in version (iOS): 3.11.0

Resumes the current audio playback.

---

stopAudio

New in version (Android): 3.11.0

New in version (iOS): 3.11.0

Stops the current audio playback.

seekTo

New in version (Android): 3.11.0

New in version (iOS): 3.11.0

Seeks to the given time in the current audio playback. This method takes the desired time in milliseconds.

setPlaybackRate

New in version (Android): 4.0

New in version (iOS): 4.0

Sets the playback rate to the specified value. Only values between 0.5 and 2.0 are allowed. Values outside these bounds will be clipped to their respective limits. The playback rate gets reset to the default rate of 1.0 on app restarts.

getPlaybackRate

New in version (Android): 4.0

New in version (iOS): 4.0

Requests the current playback rate which will be the single parameter of the callback function.

addStatusListener

New in version (Android): 3.11.0

New in version (iOS): 3.11.0

Adds a listener for playback state changes.

removeStatusListener

New in version (Android): 3.11.0

New in version (iOS): 3.11.0

Removes a listener for playback state changes.

addProgressListener

New in version (Android): 3.11.0

New in version (iOS): 3.11.0

Adds a listener for playback progress changes.

removeProgressListener

New in version (Android): 3.11.0

New in version (iOS): 3.11.0

Removes listener for playback progress changes.

5.2.11 Web Player specifics

Due to WebViews being implemented using iframes in Web Player and Web Newsstand, JavaScript injection works different in this context.

The global purple object can be made available by including `purpleInterface.js` in the embedded page. An interface for the Purple Object is re-implemented in the child page. When a function of the purple object is called, a HTML5 PostMessage will be sent to the parent window (Web Player / Web Newsstand). This will invoke the actual call on the Purple Object. The Web Player / Web Newsstand will then respond with another HTML 5 PostMessage which the child window (iframe) will process.

From V 3.0.0 `purpleInterface.js` is included in the Web Player repository. The latest version is delivered via Purple DS | Web Newsstand. It is recommended to include the script from one of the following URLs to assure to always use the latest version:

https://kiosk.purplemanager.com/scripts/purpleInterface.js

https://kiosk.purplemanager.com/scripts/purpleInterface.min.js

Note: Please be aware that only sites can be displayed which have the `X-FRAME-OPTIONS` header set correctly. Read here for details: https://developer.mozilla.org/en/docs/Web/HTTP/Headers/X-Frame-Options

purpleInterface.js (excerpt)

```
window.purpleInterface = {
callbacks: {},
util: {
    receiveMessage: function (event) {
        try {
            // get response data
            var responseData = JSON.parse(event.data);
            var value = responseData.value;
```

```
            var callbackId = responseData.callbackId;
            var key = responseData.key;

            if (callbackId) {
                // call callback function from callback map
                if(key) {
                    window.purpleInterface.callbacks[callbackId](key, value);
                } else {
                    window.purpleInterface.callbacks[callbackId](value);
                }
                // delete callback
                window.purpleInterface.callbacks[callbackId] = null;
            } else if(key === 'RELOAD'){
                window.document.location.reload();
            } else if(key === 'HISTORY_BACK') {
                window.history.back();
            } else if(key === 'HISTORY_FORWARD') {
                window.history.forward();
            } else if(key === 'DOCUMENT_TITLE') {
                window.purpleInterface.util.postMessage('DOCUMENT_TITLE', 'DOCUMENT_
↪TITLE', document.title);
            }

        } catch (e) {
        }
    },
    postMessage: function (type, key, value, callback) {

        if (window !== window.parent) {

            // create requestData
            var requestData = {
                type: type,
                key: key
            };
            if (value) {
                requestData.value = value;
            }
            if (callback) {
                // create id = index in callback array
                var callbackId = window.purpleInterface.util.generateUUID();
                requestData.callbackId = callbackId;
                // add callback to callback array
                window.purpleInterface.callbacks[callbackId] = callback;
            }

            // call postMessage
            window.parent.postMessage(JSON.stringify(requestData), '*');
        }

    },
    generateUUID: function () {
        var d = new Date().getTime();
        return 'xxxxxxxx-xxxx-4xxx-yxxx-xxxxxxxxxxxx'.replace(/[xy]/g, function (c) {
            var r = (d + Math.random() * 16) % 16 | 0;
            d = Math.floor(d / 16);
            return (c === 'x' ? r : (r & 0x3 | 0x8)).toString(16);
        });
```

```
    }
  }
};

document.addEventListener('DOMContentLoaded', function () {
    window.addEventListener('message', window.purpleInterface.util.receiveMessage);

    window.purpleInterface.util.postMessage('LOAD', 'LOAD', null, function () {

        if (!window.purple) {

            window.purple = {};

            var links = document.querySelectorAll('a[href^="purple://"], a[ ^="pkapp:/
↪/"], a[href^="pkitem://"]');
            for (var i = 0; i < links.length; i++) {
                links[i].addEventListener('click', function (e) {
                    window.purpleInterface.util.postMessage('ACTION_URL', 1, this.
↪href);
                    e.preventDefault();
                });
            }

            // purple object
            window.purple.metadata = {
                getMetadata: function (key, callback) {
                    window.purpleInterface.util.postMessage('META', key, null,␣
↪callback);
                }
            };

            window.purple.state = {
                setState: function (key, value) {
                    window.purpleInterface.util.postMessage('STATE', key, value,␣
↪null);
                },
                getState: function (key, callback) {
                    window.purpleInterface.util.postMessage('STATE', key, null,␣
↪callback);
                }

            };

            window.purple.issue = {
                getPages: function (callback) {
                    window.purpleInterface.util.postMessage('PAGES', 'PAGES', null,␣
↪callback);
                },
                getToc: function (callback) {
                    window.purpleInterface.util.postMessage('TOC', 'TOC', null,␣
↪callback);
                }
            };

            window.purple.closeView = function() {
                window.purpleInterface.util.postMessage('CLOSE_VIEW', 'CLOSE_VIEW');
            };
```

```
            if ('onPurpleLoad' in window && typeof onPurpleLoad === 'function') {
                onPurpleLoad();
            }

            var search = window.location.search;
            if (document.referrer){
                if (search) {
                    search += '&' + document.referrer.split('?')[1];
                } else {
                    search = '?' + document.referrer.split('?')[1];
                }
            }
            history.replaceState({}, document.title, window.location.origin + window.
↪location.pathname + search);
        }

    });
});
```

CHAPTER 6

Entitlement server connection

6.1 Overview

Entitlement allows content to be shown to authorized users of apps and web newsstands only. Purple DS can utilize entitlement to verify app and newsstand requests with an external entitlement server.

Fig. 6.1: Quick overview: How Purple DS utilizes external entitlement servers.

6.2 Workflow

To achieve this, an entitlement server must be provided and some configuration must be carried out in Purple DS | Manager.

Fig. 6.2: 4 simple steps to implement entitlement

6.2.1 Provide an entitlement server

On client side, an entitlement server must be provided. Purple DS may connect to several types of entitlement servers, including but not limited to the following:

- *Standard Entitlement (V1)* A very simple entitlement server definition.
- *Standard Entitlement (V2)* An enhancement of the previous to have more control.
- *Adobe DPS Entitlement V1* Connect to entitlement servers following the Adobe DPS Entitlement standard, used by several publishers.

6.2.2 Issue configuration

Hide issues

By default, all issues can be read by all users of an app or web newsstand. This is usually not desired when an entitlement server is used. There are two stages of privacy to achieve.

1. An issue can be marked as "Paid content" in Purple DS | Manager. In this case, the issues thumbnail and title are visible for the unauthorized user, as well as some preview content, but the user must log in to the entitlement server in order to read the full content.

Note: There may be other options to unlock such content, e.g. app store payment or coupon redemption, depending on the configuration.

---

2. An issue can be marked as "Content protected via Entitlement" in Purple DS | Manager. In this case, the issue is not to be seen for unauthorized users.

Common issue identifiers

The entitlement server provides information about entitled issue IDs for the logged in user. This obviously requires a common understanding of the entitlement server and Purple DS about the meaning of the exchanged issue IDs.

There are three possible implementations to achieve this:

1. The most common way is to configure the clients issue ID at the appropriate issue in the Purple DS | Manager. Purple DS | Delivery detects issues, where any of the delivered issue IDs is configured, and allows access.

Note: It is possible to configure the same client issue ID at multiple issues in the Purple DS | Manager to allow access to multiple issues at once.

---

2. The other way round, Purple DS issue IDs can be configured at the client side, and the clients entitlement server delivers them. Purple DS | Delivery recognizes it's own issue IDs and allows access to the issues.
3. For some use cases it is sufficient, to allow access to all app content. In this case, the clients entitlement server delivers a single pseudo issue ID, which Purple DS | Delivery interprets as "allow access to all issues".

6.2.3 Entitlement server configuration

To use entitlement to unlock app content, the clients entitlement server must be configured in Purple DS | Manager. There is a special tab in the apps's tab bar for the entitlement settings. Add a server, configure the URL and optionally the cache duration (depending of the entitlement server type) and save the configured server.

App configuration

Entitlement must be switched on for the app. Several additional parameters can be set, like force login at app start and others. This is done at the same tab as the server configuration. After this, the app can be built as usual.

Web newsstand configuration

To use entitlement in a web newsstand, it is to be configured in the basic settings of the app, provided a web newsstand is configured at all. This setting can be changed at any time.

6.3 Available entitlement interfaces

6.3.1 Standard Entitlement (V1)

A Standard Entitlement Server in Version 1 provides an HTTP REST interface with 3 methods to login, retrieve a list of issues and logout. It is configured with the URL and cache validity.

API Description

1. Login

By logging in, users obtain a session token that is required for all subsequent API calls.

Request

POST `/user/login`

Request header	Value
Content-Type	application/x-www-form-urlencoded

Parameters (Request body)

Name	Description
username	urlencoded username
password	urlencoded password

Sample request

```
curl '<Endpoint-Base-URL>/user/login' -H 'content-type: application/x-www-form-
↪urlencoded' -d 'username=testuser%40example.com&password=1234'
```

Response

In case of success, the answer is a string representing the session token that is to be used for all further API calls.

HTTP Status code	Response body
200 OK	the authentication token

Sample response body

```
2ef9f161-46e1-90a1-7ed6-658256124b4c
```

Error response

HTTP Status code	Response body	Description
403 Forbidden	WRONG_CREDENTIALS	username and/or password incorrect
403 Forbidden	USER_DEACTIVATED	user is deactivated

Sample error response body

```
WRONG_CREDENTIALS
```

2. List issues

With the help of the session token, the system asks for a list of unique identifiers of issues. The implementation can deliver either Purple DS issue IDs or external issue numbers, which are configured as "Issue No." in the Purple DS | Manager.

The special case of returning an array of exactly one empty string (`[""]` in JSON) is interpreted as "all issues" and can be used to grant access to all app content, if this is the intended use case.

Request

GET `/issues/list`

Parameters (URL)

Name	Description
token	the token received from login

Sample request

```
curl '<Endpoint-Base-URL>/issues/list?token=2ef9f161-46e1-90a1-7ed6-658256124b4c'
```

Response

On success, a JSON String Array containing the issue id's or issue numbers of the issues available for the user is returned.

Response header	Value
Content-Type	application/json;charset=UTF-8

HTTP Status code	Response body
200 OK	JSON-Encoded Array of Strings

Sample response body

```
["842a954728n7490118s0b8329ff","147b876348z9371540994872649dr",
↪"143a938211b058372659d737163ab"]
```

3. Logout

A call to logout should invalidate the token. Further calls to the API with the token are expected to fail.

Request

POST `/user/logout`

Request header	Value
Content-Type	application/x-www-form-urlencoded

Parameters (Request body)

Name	Description
token	the authentication token

Sample request

```
curl '<Endpoint-Base-URL>/user/logout' -H 'content-type: application/x-www-form-
↪urlencoded' -d 'token=2ef9f161-46e1-90a1-7ed6-658256124b4c'
```

Response

No response body is expected to be returned by this call.

HTTP Status code	Description
200 OK	the user is logged out

Configuration

The following parameters can be set when this entitlement type is selected in the **Purple DS | Manager** for an app:

Parameter	Description	Example
URL	Server URL of the entitlement REST interface	https://example.com/rest/api
Cache Validity in Minutes	Number of Minutes a retrieved issue list should be cached	1

6.3.2 Standard Entitlement (V2)

A Standard Entitlement Server in Version 2 provides an HTTP REST interface with 5 methods to login, renew the session token, retrieve a list of issues, verify access to a single issue and logout. It is configured with the URL.

API Description

1. Login

By logging in, users obtain a session token that is required for all subsequent API calls.

Request

POST `/user/login`

Request header	Value
Content-Type	application/x-www-form-urlencoded

Parameters (Request body)

Name	Description
username	urlencoded username
password	urlencoded password
appId	the bundle identifier / package name of the app or a configured value
deviceId	the unique device id of the calling device

Sample request

```
curl '<Endpoint-Base-URL>/user/login' -H 'content-type: application/x-www-form-
↪urlencoded'\
 -d 'username=testuser%40example.com&password=1234&appId=com.domain.myapp&
↪deviceId=6789-1234-1234-123-123456'
```

Response

In case of success, the answer is a string representing the session token that is to be used for all further API calls.

HTTP Status code	Response body
200 OK	the authentication token

Sample response body

```
2ef9f161-46e1-90a1-7ed6-658256124b4c
```

Error response

HTTP Status code	Response body	Description
403 Forbidden	WRONG_CREDENTIALS	username and/or password incorrect
403 Forbidden	USER_DEACTIVATED	user is deactivated

Sample error response body

```
WRONG_CREDENTIALS
```

2. Token renewal

By renewing the token, users obtain an updated session token required for all subsequent API calls.

Request

POST `/token/renew`

Request header	Value
Content-Type	application/x-www-form-urlencoded

Parameters (Request body)

Name	Description
token	the token retrieved by login or a prior renew
appId	the bundle identifier / package name of the app or a configured value
deviceId	the unique device id of the calling device

Sample request

```
curl '<Endpoint-Base-URL>/token/renew' -H 'content-type: application/x-www-form-
↪urlencoded'\
 -d 'token=2ef9f161-46e1-90a1-7ed6-658256124b4c&appId=com.domain.myapp&deviceId=6789-
↪1234-1234-123-123456'
```

Response

In case of success, the answer is a string representing the new session token that is to be used for all further API calls. The prior session token is no longer used by the app.

HTTP Status code	Response body
200 OK	the authentication token

Sample response body

```
3ef9f161-46e1-90n1-7ed6-658256hl4bff
```

Error response

HTTP Status code	Description
401 Unauthorized	the transmitted token was incorrect

3. List issues

With the help of the session token, the system asks for a list of unique identifiers of issues. The implementation can deliver either Purple DS issue IDs or external issue numbers, which are configured as "Issue No." in the Purple DS | Manager.

The special case of returning an array of exactly one empty string (`[""]` in JSON) is interpreted as "all issues without an Issue No." and can be used to grant access to all app content, if this is the intended use case.

Request

GET `/issues/list`

Parameters (URL)

Name	Description
token	the token retrieved by login or renew
appId	the bundle identifier / package name of the app or a configured value
deviceId	the unique device id of the calling device

Sample request

```
curl '<Endpoint-Base-URL>/issues/list?token=2ef9f161-46e1-90a1-7ed6-658256124b4c&
↪appId=com.domain.myapp&deviceId=6789-1234-1234-123-123456'
```

Response

On success, a JSON String Array containing the issue id's or issue numbers of the issues available for the user is returned.

Response header	Value
Content-Type	application/json;charset=UTF-8

HTTP Status code	Response body
200 OK	JSON-Encoded Array of Strings

Sample response body

```
["842a954728n7490118s0b8329ff","147b876348z9371540994872649dr",
↪"143a938211b058372659d737163ab"]
```

4. Verify access to an issue

After retrieving a list of issue identifiers, the app presents the identified issues to the user. Right before the app opens a specific issue, it verifies again, that the user has still access to that issue.

This verification is done with the external issue number, which is configured as "Issue No." in the **Purple DS | Manager** and if this is not successful with the Purple DS issue ID as well.

In the aforementioned special case of granting access to all app content by returning an array of exactly one empty string (`[""]` in JSON) this call should simply return an HTTP status 200 OK on any request.

Request

GET `/issue/verify`

Parameters (URL)

Name	Description
token	the token retrieved by login or renew
issueId	the identifier of an issue
appId	the bundle identifier / package name of the app or a configured value
deviceId	the unique device id of the calling device

Sample request

```
curl '<Endpoint-Base-URL>/issue/verify?token=2ef9f161-46e1-90a1-7ed6-658256124b4c&
↪appId=com.domain.myapp&deviceId=6789-1234-1234-123-123456'
```

Response

No response body is expected to be returned by this call. The HTTP status code is interpreted as follows:

HTTP Status code	Description
200 OK	the issue is accessible
401 Unauthorized	the token is invalid, the issue is not accessible
403 Forbidden	the issue is not accessible

5. Logout

A call to logout should invalidate the token. Further calls to the API with the token are expected to fail.

Request

POST `/user/logout`

Request header	Value
Content-Type	application/x-www-form-urlencoded

Parameters (Request body)

Name	Description
token	the token retrieved by login or renew
appId	the bundle identifier / package name of the app or a configured value
deviceId	the unique device id of the calling device

Sample request

```
curl '<Endpoint-Base-URL>/user/logout' -H 'content-type: application/x-www-form-
↪urlencoded' \
 -d 'token=2ef9f161-46e1-90a1-7ed6-658256124b4c&appId=com.domain.myapp&deviceId=6789-
↪1234-1234-123-123456'
```

Response

No response body is expected to be returned by this call.

HTTP Status code	Description
200 OK	the user is logged out

Configuration

The following parameters can be set when this entitlement type is selected in the **Purple DS | Manager** for an app:

Parameter	Description	Example
URL	Server URL of the entitlement REST interface	https://example.com/rest/api
appId	The appId to be used to identify the app at the entitlement server. Default is the package name.	my-app

6.3.3 Adobe DPS Entitlement V1

The Adobe DPS Entitlement description is spread over multiple documents and tutorials to be found here: http://www.adobe.com/devnet/digitalpublishingsuite/entitlement.html and elsewhere.

Here is a brief aggregation of the information used when implementing the appropriate plugin.

API Description

All calls described in the following are designed to return an XML response. Though not all server implementations set the appropriate response content type, therefore our (universal client) implementation doesn't rely on that. Furthermore, the XML response has an attribute named `httpResponseCode` which is usually set to an appropriate HTTP status code, while the actual HTTP status code of the response is `200 Ok`. Our implementation accepts both the actual HTTP status code and the `httpResponseCode` attribute from the XML response and treats them synonymously.

1. Sign in with credentials

By signing in, users obtain an authentication token that is required for all subsequent API calls.

Request

GET `/SignInWithCredentials`

Parameters (URL)

Name	Description
emailAddress	urlencoded username
password	urlencoded password
appId	the bundle identifier / package name of the app or a configured value
uuid	the unique device id of the calling device

Sample request

```
curl '<Endpoint-Base-URL>/SignInWithCredentials?emailAddress=testuser%40example.com&
↪password=1234&appId=com.package.app&uuid=3944-1345-1145656-5645'
```

Response

In case of success, the answer is an XML object with an `httpResponseCode` of 200 and an `authToken`.

HTTP Status code	Response body
200 OK	XML object, see sample below

Sample response body

```
<result httpResponseCode="200">
  <authToken>VFgrV1lKd09pL2s2NnlIKzE5R</authToken>
</result>
```

Error response

HTTP Status code	Response body
200 OK	XML object, see sample below
401 Unauthorized	XML object, see sample below

Sample error response body

```
<result httpResponseCode="401" errorCode=""/>
```

2. Renew authentication token

Using this call, an authentication token retrieved by SignInWithCredentials can be renewed. The result is either the same or a new token which should be used instead in following API calls.

Request

GET `/renewAuthToken`

Parameters (URL)

Name	Description
authToken	an authentication token retrieved by SignInWithCredentials or a prior call to renewAuthToken
appId	the bundle identifier / package name of the app or a configured value

Sample request

```
curl '<Endpoint-Base-URL>/renewAuthToken?authToken=VFgrV1lKd09pL2s2NnlIKzE5R&
↪appId=com.package.app'
```

Response

In case of success, the answer is an XML object with an `httpResponseCode` of 200 and a new `authToken`.

HTTP Status code	Response body
200 OK	XML object, see sample below

Sample response body

```
<result httpResponseCode="200">
  <authToken>VFgrV1lKd09pL2s2NnlIKzE5R</authToken>
</result>
```

Error response

HTTP Status code	Response body
200 OK	XML object, see sample below
401 Unauthorized	XML object, see sample below

Sample error response body

```
<result httpResponseCode="401" errorCode=""/>
```

3. Entitlements

This calls delivers identifiers of the issues, the user is entitled to see. While Purple DS issue ID's would be handled correctly, normally external issue identifiers are used, which are configured as "Issue No." in the Purple DS | Manager.

Request

GET `/entitlements`

Parameters (URL)

Name	Description
authToken	an authentication token retrieved by SignInWithCredentials or a prior call to renewAuthToken
appId	the bundle identifier / package name of the app or a configured value

Sample request

```
curl '<Endpoint-Base-URL>/entitlements?authToken=VFgrV1lKd09pL2s2NnlIKzE5R&appId=com.
↪package.app'
```

Response

In case of success, the answer is an XML object with an `httpResponseCode` of 200 and a list of `productId` elements wrapped in an `entitlements` element.

HTTP Status code	Response body
200 OK	XML object, see sample below

Sample response body

```
<result httpResponseCode="200">
  <entitlements>
    <productId>com.package.app.07.2017</productId>
        <productId>com.package.app.09.2017</productId>
        <productId>com.package.app.thanksgiving.special</productId>
        <productId>com.package.app.11.2017</productId>
  <entitlements>
</result>
```

Error response

HTTP Status code	Response body
200 OK	XML object, see sample below
401 Unauthorized	XML object, see sample below

Sample error response body

```
<result httpResponseCode="401" errorCode=""/>
```

4. Verify entitlement

After retrieving a list of issue identifiers, the app presents the identified issues to the user. Right before the app opens a specific issue, it verifies again, that the user has still access to that issue.

Request

GET `/verifyEntitlement`

Parameters (URL)

Name	Description
authToken	an authentication token retrieved by SignInWithCredentials or a prior call to renewAuthToken
productId	the issue identifier as retrieved by the call to entitlements
appId	the bundle identifier / package name of the app or a configured value

Sample request

```
curl '<Endpoint-Base-URL>/verifyEntitlement?authToken=VFgrV1lKd09pL2s2NnlIKzE5R&
↪productId=com.package.app.07.2017&appId=com.package.app'
```

Response

In case of success, the answer is an XML object with an `httpResponseCode` of 200 and a boolean value in an `entitled` element.

HTTP Status code	Response body
200 OK	XML object, see sample below

Sample response body

```
<result httpResponseCode="200">
  <entitled>true<entitled>
</result>
```

Error response

HTTP Status code	Response body
200 OK	XML object, see sample below
401 Unauthorized	XML object, see sample below

Sample error response body

```
<result httpResponseCode="401" errorCode=""/>
```

Configuration

The following parameters can be set when this entitlement type is selected in the **Purple DS | Manager** for an app:

Parameter	Description	Example
URL	Server URL of the entitlement REST interface	https://example.com/rest/api
Cache Validity in Minutes	Number of Minutes a retrieved issue list should be cached	1
appId	The appId to be used to identify the app at the entitlement server. Default is the package name.	my-app

CHAPTER 7

Manager public REST interface

Before starting with specific APIs we give an overview of the basic data model and the complete workflow from login to upload of packages.

7.1 Data model

Fig. 7.1: Publication and issue data model

7.2 Workflow

At first you have to *Login* to the Purple DS | Manager. That generates a `sessionID` used for all other API calls.

The second step is to retrieve an ID of a publication for which an issue will be edited. Therefore a *Publication list* containing all publications visible to the logged in user may be requested. From the result the publication name may be used to identify a specific publication.

Using the publication ID, either a new issue is created by *Create issue* or the list of all issues of that publication is requested by *Issue list*. Again the issue name may be used to identify a specific issue and the ID is used for further operations.

Now in a similar way an issue version is created by *Create issue version* or requested from the list of versions of this issue to be retrieved by *Issue version list*.

Hint: As a matter of fact when creating an issue, a first issue version is created automatically. Therefore, the latter is most probably what you would do.

Using the version ID, new packages for this issue version can be uploaded. The packages contain the actual issue data. A package is uploaded to an issue version by *Upload package* and then the issue version is published by *Change issue version status*.

Don't forget to *Logout* to invalidate the `sessionID` when you are done.

The following picture shows the complete workflow needed for publishing a new issue for an existing publication including the propagation of the necessary ID parameters from one request to another.

Fig. 7.2: Simple workflow to publish a new issue

For more sophisticated use cases, one can upload multiple packages to an issue version and *Associate package with platforms or device classes*. In order to have all parameter values for these constraints, the global *Configuration* - containing the lists of platforms and device types - has to be queried once.

Furthermore, additional assets like videos or music can be added by *Upload package*, preview or "coming soon" versions can be created and many more. All of this is laid out in detail in the following text.

7.3 API Description

This chapter describes all APIs to fulfill the package management workflow explained before.

7.3.1 Basic requirements for all API calls

Base URL

Base URL for all API calls is

Target	URL
Production system	https://purplemanager.com/purple-manager-backend
Staging system	https://staging.purplemanager.com/purple-manager-backend

Security header

Each request to the backend must have a request header named `xr` and an additional query parameter (even if it's a POST request) named `xr`, both containing the same random number (integer).

7.3.2 Configuration

For later usage during upload of packages the global configuration needs to be queried. It returns all platforms and device classes which need to be passed using their identifiers.

[GET] /configuration

Request Parameters

none

Result

Result is a JSON object with the following relevant entries

Field	Type	Description
platforms	Array	All known platforms as JSON objects containing an id and name field
deviceClasses	Array	All known device classes as JSON objects containing an id and name field and an optional field "subclasses" containing sub-device classes.

7.3.3 Login

After successful login a session token is returned, which is used for each subsequent API call.

[POST] /auth/login

Request Parameters

Parameter name	Type	Description
email	String	Email address of a valid Purple DS | Manager user
password	String	Password of a valid Purple DS | Manager user

Result

Result is a JSON object from which only the property `sessionID` is used for further API usage.

Field	Type	Description
sessionID	String	Session token used for subsequent API calls

7.3.4 Logout

Logout invalidates the current session.

[POST] /auth/logout

Request Parameters

Parameter name	Type	Description
sessionID	String	Session token that is invalidated

Result

No result data.

7.3.5 Publication list

A publication list is associated with one or more accounts (teams) the logged in user is associated with. Using this list a publication may be selected by name (or other criteria).

[GET] /publication/list

Request Parameters

Parameter name	Type	Description
sessionID	String	Valid session token created by *Login*.

Result

The result is a JSON array containing all publications accessible for the logged in user. Each publication is a JSON object containing the following relevant properties:

Field	Type	Description
id	String	Publication id used for subsequent API calls
name	String	Name of the publication

7.3.6 Create issue

In case the issue for which the package shall be uploaded does not exist yet, it may be created using this API call.

[POST] /issue

Request Parameters

Parameter name	Type	Description
sessionID	String	Valid session token created by *Login*.
publicationId	String	Id of the publication returned by *Publication list*.
id	String	Set to `-1`
name	String	Name of the new issue.
description	String	A description of the new issue (optional).
published	Date	Date of publication of the new issue (may be in the past or future). Format: `yyyy-MM-dd`
file	File	issue thumbnail file (jpg or png)

Result

ID of the new issue

7.3.7 Issue list

Querying an issue list of a publication helps to select an issue by name.

[GET] /publication/listissues

Request Parameters

Parameter name	Type	Description
sessionID	String	Valid session token created by *Login*.
publicationId	String	Id of a publication returned by *Publication list*.

Result

The result is a JSON array containing all issue of the given publication. Each issue is a JSON object with the following relevant properties:

Field	Type	Description
id	String	Issue id used for subsequent API calls
name	String	Name of the issue

7.3.8 Change issue "coming soon" status

In order to change the issue "coming soon" status there are activating and deactivating API calls. These calls are distinguished by their path.

Purpose	Call
Activate "coming soon"	[POST] /issue/comingsoon
Deactivate "coming soon"	[POST] /issue/notcomingsoon

Note: Setting `comingSoon` for an issue resets any versions `active` status and vice versa. See *Change issue version status*.

Request Parameters

Parameter name	Type	Description
sessionID	String	Valid session token created by *Login*.
id	String	Id of an issue for which the status is to be changed. It may be retrieved by a call to *Issue list* or by *Create issue*.

Result

No result data.

7.3.9 Create issue version

Shall a package be uploaded into a new version of an issue, it has to be created beforehand. Every issue has a first version (number 1) created automatically when the issue is created.

Besides the "full version" of an issue there may be "customer preview versions" used to present part of an issue to the customer for free. For these "customer preview versions" the first version has to be created by this API call explicitly.

[POST] /version

Request Parameters

Parameter name	Type	Description
sessionID	String	Valid session token created by *Login*.
issueId	String	Id of an issue returned by *Issue list* or by *Create issue*.
customerPreview	boolean	Optional, if true a new "customer preview version" is created, otherwise a full version is created.

Result

The result is the id of the new issue version.

7.3.10 Issue version list

Having the list of all versions of an issue, one may pick a version by its version number or the status (published, preview or "coming soon") and use the id of this version in subsequent calls.

[GET] /issue/listversions

Request Parameters

Parameter name	Type	Description
sessionID	String	Valid session token created by *Login*.
issueId	String	Id of an issue returned by *Issue list* or by *Create issue*.
customerPreview	boolean	Optional, if true "customer preview versions" are returned, otherwise the list of full versions is returned.

Result

The result is a JSON array containing all versions of the given issue. Each version is a JSON object with the following relevant properties:

Field	Type	Description
id	String	Version id that may be used in subsequent API calls
number	Integer	Version number
active	Boolean	Flag indicating whether this version is published in release app.
preview	Boolean	Flag indicating whether this version is published in preview app.

7.3.11 Change issue version status

In order to change issue version status there are activating and deactivating API calls for each of the status flags: `active` (published to release and preview app) and `preview` (published to preview app). These calls are distinguished by their path.

Note: For backward compatibility reasons the issue status `comingSoon` can be set through a version as well.

Boundary conditions

1. In order to Change issue version status to be published into the release app, there needs to be a bundle uploaded, see *Upload package*.
2. If a version is published to release or preview app, the same status of another version is automatically deleted, so there is always exactly one issue version published to release or preview app.
3. A version published to release app cannot be modified afterwards, a new version has to be created instead.
4. Setting `comingSoon` for an issue resets any versions `active` status and vice versa.

Purpose	Call	
Publish to release app	[POST] /version/activate	
Unpublish from release app	[POST] /version/deactivate	
Publish to preview app	[POST] /version/preview	
Unpublish from preview app	[POST] /version/notpreview	
Activate “coming soon”	[POST] /version/comingsoon	deprecated, use /issue/comingsoon instead
Deactivate “coming soon”	[POST] /version/notcomingsoon	deprecated, use /issue/notcomingsoon instead

Request Parameters

Parameter name	Type	Description
sessionID	String	Valid session token created by *Login*.
versionId	String	Id of an issue version for which the status is to be changed. It may be retrieved by a call to *Issue version list* or by *Create issue version*.

Result

No result data.

7.3.12 Upload package

Using this call a Purple DS archive (.pkar) is uploaded to an issue version. It may also be used to upload large asset files (e.g. videos) that are nor bundled into the Purple DS archive.

Note: The version must not been flagged as "published to a release app", a.k.a. `active`.

[POST] /package/upload

Request Parameters

Parameter name	Type	Description
sessionID	String	Valid session token created by *Login*.
versionId	String	Id of an issue version for which the status is to be changed. It may be retrieved by a call to *Issue version list* or by *Create issue version*.
file	File	File to be uploaded

Result

Result is the id of the uploaded package.

7.3.13 Associate package with platforms or device classes

After uploading a file it is not yet associated with a platform or device class. This has to be done by a subsequent API call. If a packages is not associated with platforms and/or device classes it is valid for all platforms and/or device classes.

Note: The version must not been flagged as published to a release app.

[POST] /package/editRelations

Request Parameters

These parameters need to be passed within the URL:

Parameter name	Type	Description
sessionID	String	Valid session token created by *Login*.
packageId	String	Id of a package uploaded beforehand using *Upload package*.

Body

For defining associations a request body containing a JSON object needs to be sent with the request. It has the following properties:

Field	Type	Description
t	String	Always `par`
deviceClasses	Array	An array of JSON object containing the device classes.
platforms	Array	An array of JSON objects containing the platforms
tags	String	A comma separated string of tags, may be an empty string.

Device classes object

Objects within the deviceClasses array containing the following properties:

Field	Type	Description
t	String	Must contain `pdc`
deviceClassId	String	Id of the device class that shall be selected or unselected for the package. Use the *Configuration* call to query available device classes once.
deviceClassName	String	Name of the device class that shall be selected or unselected for the package. Use the *Configuration* call to query available device classes once.
selected	Boolean	Indicates the change to the device class association, i.e. selects or unselects the device class for the package.

Platform object

Objects within the platforms array containing the following properties:

Field	Type	Description
t	String	Must contain `pap`
platformId	String	Id of the platform that shall be selected or unselected for the package. Use the *Configuration* call to query available platforms once.
platformName	String	Name of the platform that shall be selected or unselected for the package. Use the *Configuration* call to query available platforms once.
selected	Boolean	Indicates the change to the platform association, i.e. selects or unselects the platform for the package.

Result

No result data.

7.3.14 Deleting a package

Using this call a package is deleted.

[POST] /package/delete

Request Parameters

Parameter name	Type	Description
sessionID	String	Valid session token created by *Login*.
packageId	String	Id of a package uploaded beforehand using *Upload package*.

Result

No result data.

7.3.15 Delete all packages of an issue version

Using this call all packages of an issue version are deleted.

[POST] /package/deleteAll

Request Parameters

Parameter name	Type	Description
sessionID	String	Valid session token created by *Login*.
versionId	String	Id of an issue version for which the status is to be changed. It may be retrieved by a call to *Issue version list* or by *Create issue version*.
packageType	String	Indicates which type of files shall be deleted. Possible values: `content_bundle` for Purple DS archives and `asset` for all other assets.

Result

No result data.

7.4 Dynamic Resources

An app uses various resources which can be changed at runtime. They are packaged into a zip file and uploaded to the Purple DS | Manager. The required structure of the zipped content is described at *Dynamic Resources*. Once uploaded, the resource file becomes valid immediately and is downloaded by the app at the next start or resume.

7.4.1 Upload dynamic resources for an app

A new version of the dynamic resource file can be uploaded by a multipart HTTP POST request (content-type: multipart/form-data).

[POST] /app/uploadresources

Request Parameters

Parameter name	Type	Description
sessionID	String	Valid session token created by *Login*.
appId	String	Id of an app.
preview	Boolean	`true` if the preview resources should be changed, otherwise the live resources will be changed.
file	File	The dynamic resources zip file.

Result

On successful execution, the appId is returned together with an HTTP status code of 200. Otherwise, a meaningful HTTP status code and response string are returned.

7.4.2 Download dynamic resources for an app

The currently uploaded dynamic resource file can be downloaded by a GET request. If no dynamic resources have been uploaded yet, the app will use default resources and these will be returned by this request.

[GET] /app/downloadresources

Request Parameters

Parameter name	Type	Description
sessionID	String	Valid session token created by *Login*.
appId	String	Id of an app.
preview	Boolean	`true` if the preview resources should be returned, otherwise the live resources will be returned.

Result

On successful execution, the file is returned together with an HTTP status code of 200. Otherwise, a meaningful HTTP status code is returned.

CHAPTER 8

Search

Calls to search are used to retrieve the search results from issue content of an App.

8.1 URL

[POST] https://purplemanager.com/delivery/search

8.2 Request Parameters

Name	Type	Description	Required
appId	String	The id of the requesting app	YES
platform	String (android/ios/kindle)	Platform of the device	YES
preview	Boolean (true/false)	Is the requesting app a test (true) or production (false) app	NO, defaults to false
locale	String	Language locale like de_DE or en_US	YES
model	String	Model identifier like iPhone 4S, some kind of model number (iOS only)	NO
smallestScreenWidthDp	Integer	Density-Independent Pixels of the smallest screen width (Android only)	NO

Note: The paramters `model` and `smallestScreenWidthDp` are used to filter the results by device class (phone / tablet). If they are omitted all results will be returned even if the containing issue is not shown on the requesting device.

8.3 Request Body

Name	Type	Description	Required
issueIds	List<String>	when content search is active limit search to issues with given ids	NO, defaults to no limit
publicationIds	List<String>	when content search is active limit search to issues from publications with given ids	NO, defaults to no limit
phrase	String	the search phrase	YES
fuzzy	Boolean (true/false)	use a fuzzy search	NO, defaults to false
sortPages	Boolean (true/false)	sort the search hits by pages of the issues	NO, defaults to false
findAll	Boolean (true/false)	find all words of the search phrase (any word otherwise)	NO, defaults to false

```
{
  "issueIds": ["xxx"],
  "publicationIds": ["xxx"],
  "phrase": "xxx",
  "fuzzy": true,
  "sortPages": true,
  "findAll": true
}
```

8.4 Request Headers

Name	Type	Content	Required
Authorization	String	`Token <authToken>` results from entitled issues are only returned if the authToken grants access to these issues	NO
Content-Type	String	`application/json`	YES

8.5 Response Codes

Http Status Code	Reason
200 - OK	

8.6 Response Headers

Name	Type	Content
Content-Type	String	`application/json;charset=UTF-8`
Content-Length	int	Length of json

8.7 Response Body

A JSON Response is returned.


```
{
  "numberOfIssueHits": 1,
  "numberOfPageHits": 2,
  "issues": [
      {
          "issueId": "xxx",
          "publicationId": "xxx",
          "pages": [
              {
                  "pageIndex": 0,
                  "pageNumber": 1,
                  "pageLabel": "1",
                  "pageTitle": "",
                  "excerpt": "... <strong>xxx/strong> ..."
              },
              {
                  "pageIndex": 5,
                  "pageNumber": 6,
                  "pageLabel": "10",
                  "pageTitle": "",
                  "excerpt": "... <strong>xxx/strong> ..."
              }
          ]
      }
  ]
}
```