Adobe Digital Publishing Custom HTML Store API Reference 20 0

Custom HTML Store API Reference

This JavaScript API provides the tools you need to add scripts to a custom store-front page that communicates with the Digital Publishing Store server. You can make this store-front available to viewer designers through Adobe Viewer Builder, so that they can embed it in a viewer. A custom store-front can allow viewer users to:

- Purchase subsciptions that entitle them to view and download items
- Log in with their credentials
- Discover what items are available
- View, purchase, and download items from the store

Use the links on the left to get detailed information for each of the JavaScript API functions. For many of the calls, you must provide callback functions to handle the result of the operation; details of the callback syntax are also provided.

The API makes use of these terms.

Terminology

device	An electronic device that can be used to view and download folios offered by a digital publishing store.
folio	A digitally-published item offered for purchase and download by a digital publishing store. Identified by a unique identifier (such as an ISBN), and also by a product identifier unique within a particular store.
folio state	A server state constant associated with each folio that tells you whether the folio can be purchased, downloaded, viewed, or updated, and whether such an operation is aleady in process for it. See Folio States.
receipt	A record of a transaction with the store server. Each store defines its own receipt format.
store	In this context, a server that offers digitally published items for sale and download, to users who are authorized through purchasing a *subscription*.
subscription	A user's purchased entitlement to download published items from a *store*. Subscriptions have a term of validity. A user must authenticate according to store requirements, and have a currently valid subscription in order to view and download folios.

Archive folio

Archive a folio with the given product ID.

Since SDK: 20

adobe.dps.store.archiveFolio

Parameter	Type	Description
folioProductId	*string*	A string containing the product id of the folio to archive.
callback	*function*	Function that will be called with this signature: {"result":status} where status is one of "succeeded", "failed"

archiveFolioFunction

Parameter	Type	Description
dictionary	*folioResult*	A dictionary containing the result status.
"result"	*folioStatus*	The folio-buying status.

API Snippet

archiveFolio

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-
transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
<title>viewFolio API</title>
<script type="text/javascript">
function archiveFolio()
{
  if (window.adobedpscontextloaded)
  {
    var productId = document.getElementById("archivefolio").value;
    // call into the archiveFolio API
    adobe.dps.store.archiveFolio(productId, function(returnedData)
     {
      var status = returnedData['result'];
      document.write("<p>Archive Folio returned with a " + status + " result</p>");
     }
    );
  }
}
</script>
</head>

<body>
<p>Archive Folio</p>
<input type="text" id="archivefolio" value="folioProductId"/>
<button type="button" onclick="archiveFolio()">Archive Now</button>
</body>
</html>
```

Buy folio

Purchase a folio with the given product ID.

Since SDK: 20

adobe.dps.store.buyFolio

Parameter	Type	Description
folioProductId	*string*	A string containing the product id of the folio to buy.
callback	*function*	Function that will be called with this signature: {"result":status} where status is one of "succeeded", "failed", "cancelled", "refunded"
withAnalytics	*boolean*	An optional boolean indicating whether you want the Viewer app to track analytics for this event. If you track analytics via JavaScript in your own store, pass in false. Default value: true

buyFolioFunction

Parameter	Type	Description
dictionary	*folioResult*	A dictionary containing the result status.
"result"	*folioStatus*	The folio-buying status.

API Snippet

buyFolio

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-
transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
<title>Buy Folio API</title>
<script type="text/javascript">
function buyFolio()
{
  var folioID = document.getElementById("buy").value;
  // call into the buyFolio API
  adobe.dps.store.buyFolio(folioID, function(returnedData)
    {
      var status = returnedData['result'];
      document.write("<p>Buy Folio returned with a " + status + " result</p>");
    }
  );
}
</script>
</head>

<body>
<p>Buy MyFolio</p>
<button type="button" id="buy" value="12345" onclick="buyFolio()">Buy Now</button>
</body>
</html>
```

Buy subscription

Purchase a subscription with the given product ID.

Since SDK: 20

adobe.dps.store.buySubscription

Parameter	Type	Description
subscriptionProductId	*string*	A string containing the product id of the subscription to buy.
callback	*buySubscriptionFunction*	Function that will be called with this signature: {"result":status} where status is one of "succeeded", "failed", "cancelled", "refunded", "notsupported"
withAnalytics	*boolean*	An optional boolean indicating whether you want the Viewer app to track analytics for this event. If you track analytics via JavaScript in your own store, pass in false. Default value: true

buySubscriptionFunction

Parameter	Type	Description
dictionary	*buySubscriptionResult*	A dictionary containing the result status.
"result"	*buySubscriptionStatus*	The subscription buying status.

API Snippet

buySubscription

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-
transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
<title>Buy Subscription API</title>
<script type="text/javascript">
function buySubscription()
{
  var folioID = document.getElementById("buy").value;
  // call into the buySubscription API
  adobe.dps.store.buySubscription(subscriptionID, function(returnedData)
    {
      var status = returnedData['result'];
      document.write("<p>Buy Subscription returned with a " + status + " result</p>");
    }
  );
}
</script>
</head>

<body>
<p>Buy MySubscription</p>
<button type="button" id="buy" value="12345" onclick="buySubscription()">Buy Now</button>
</body>
</html>
```

Dismiss custom dialog

This method must be called within the custom dialog in order to dismiss the custom dialog.

Since SDK: 20

adobe.dps.store.dismissCustomDialog

API Snippet

dismissCustomDialog

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-
transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
<title>dismissCustomDialog API</title>
<script type="text/javascript">
function dismissCustomDialog()
{
  if (window.adobedpscontextloaded)
  {
    // call into the presentCustomDialog API
    adobe.dps.store.dismissCustomDialog();
  }
}
</script>
</head>

<body>
<button type="button" onclick="dismissCustomDialog()">Dismiss Custom Dialog</button>
</body>
</html>
```

Dismiss welcome screen

Dismiss the welcome screen. This method must be called within the custom welcome screen webview.

Since SDK: 20

adobe.dps.store.dismissWelcomeScreen

API Snippet

dismissWelcomeScreen

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-
transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
<title>dismissWelcomeScreen API</title>
<script type="text/javascript">
window.onadobedpscontextloaded = loadPage;
function loadPage()
{
  // call into the dismissWelcomeScreen API
  if (window.adobedpscontextloaded)
  {
    adobe.dps.store.dismissWelcomeScreen();
  }
}
</script>
</head>

<body>
</body>
</html>
```

Display welcome screen

Display a custom welcome screen webview. The welcome screen is a transparent full screen webview that loads a url configured in the viewer. You dismiss this custom webview by calling the dismissWelcomeScreen method from within the invoked webview.

Since SDK: 20

adobe.dps.store.displayWelcomeScreen

API Snippet

displayWelcomeScreen

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-
transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
<title>displayWelcomeScreen API</title>
<script type="text/javascript">
window.onadobedpscontextloaded = loadPage;
function loadPage()
{
  // call into the displayWelcomeScreen API
  if (window.adobedpscontextloaded)
  {
    adobe.dps.store.displayWelcomeScreen();
  }
}
</script>
</head>

<body>
</body>
</html>
```

Get device id

Get a unique identifier string for the device the viewer is running on.

Since SDK: 20

adobe.dps.store.getDeviceId

Parameter	Type	Description
callback	*getDeviceIdFunction*	Function that will be called with this signature: deviceId where deviceId is the string containing the unique device ID

getDeviceIdFunction

Parameter	Type	Description
deviceId	*string*	The unique device identifier.

API Snippet

getDeviceId

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-
transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
<title>Get Device ID API</title>
<script type="text/javascript">
function getDeviceID()
{
  var deviceID = document.getElementById("device_id").value;
  // call into the getDeviceID API
  adobe.dps.store.getDeviceId(deviceID, function(returnedDeviceId)
    {
      document.write("<p>Device ID: " + returnedDeviceId + "</p>");
    }
  );
}
</script>
</head>

<body>
<p>Get Device ID</p>
<input type="text" id="device_id" width="100" />
<button type="button" id="getDeviceId" onclick="getDeviceID()">Get Device ID</button>
</body>
</html>
```

Get entitled products

Get list of product Ids that are entitled via third-party entitlements.

Since SDK: 20

adobe.dps.store.getEntitledProducts

Parameter	Type	Description
callback	*getEntitledProductsFunction*	Function that will be receive an array of Store Product Ids.

getEntitledProductsFunction

Parameter	Type	Description
products	*productArray*	Array of entitled product ids.
productId	*string*	A list of product IDs.

API Snippet

getEntitledProducts

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-
transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
<title>getEntitledProducts API</title>
<script type="text/javascript">
window.onadobedpscontextloaded = loadPage;
function loadPage()
{
  // call into the getEntitledProducts API
  if (window.adobedpscontextloaded)
  {
    adobe.dps.store.getEntitledProducts(my_callback);
  }
}
</script>
</head>

<body>
</body>
</html>
```

Get folio data

Get library state information for the folios in the library.

Since SDK: 20

adobe.dps.store.getFolioData

Parameter	Type	Description
callback	*getFolioDataFunction*	A developer-supplied function to handle the list of folios. Returned state per folio has the following values: 100, purchasable >= 200<1000 entitlied, downloadable, installable. >= 1000 viewable, update available. NOTE: If no current or cached information is available, the function will return an empty list

getFolioDataFunction

Parameter	Type	Description
folios	*folioArray*	A list containing the library state information for the folios in the library.
folio	*folio*	A list of folios.
productId	*string*	The store product identifier.
id	*string*	The unique identifier of the folio.
state	*folioState*	A FolioState constant.
price	*string*	A localized string representing the price of the folio.
title	*string*	Title extracted from the folio metadata.
description	*string*	Description extracted from the folio metadata.
broker	*string*	Authority that will entitle the folio.
thirdPartyEntitled	*boolean*	Indicates whether this folio is entitled by the entitlement server.
folioNumber	*string*	Metadata field to identify the folio number.
publicationDate	*uint*	Indicates the date the folio was published returned as unix time (uint).
targetDimensions	*string*	Dimensions of the folio passed back as a string in the form "1024x768".

API Snippet

getFolioData

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-
transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
<title>Get Folio Data API</title>
<script type="text/javascript">
function getFolioData()
{
  // call into the getFolioData API
  adobe.dps.store.getFolioData(function(returnedFolios)
    {
      var results = "<ul>";
      for (folio in returnedFolios)
      {
                var folioDate = new Date();
                folioDate.setTime( returnedFolios[folio].publicationDate );

        results += "<li>Folio: " + returnedFolios[folio].title + "</li><ul>" +
           "<li>Id: " + returnedFolios[folio].id + "</li>" +
           "<li>ProductId: " + returnedFolios[folio].productId + "</li>" +
           "<li>Price: " + returnedFolios[folio].price + "</li>" +
           "<li>State: " + returnedFolios[folio].state + "</li>" +
           "<li>Broker: " + returnedFolios[folio].broker + "</li>" +
           "<li>thirdPartyEntitled: " + returnedFolios[folio].thirdPartyEntitled + "</li>" +
           "<li>folioNumber: " + returnedFolios[folio].folioNumber + "</li>" +
```

```
            "<li>publicationDate: " + folioDate + "</li>" +
            "<li>targetDimensions: " + returnedFolios[folio].targetDimensions + "</li>" +
            "<li>Description: " + returnedFolios[folio].description + "</li>" +
            "</ul>";
      }
      results += "</ul>";
      document.write(results);
    }
  );
}
</script>
</head>

<body>
<p>Get Folio Data</p>
<button type="button" id="getFolioData" onclick="getFolioData()">Get Folio Data</button>
</body>
</html>
```

Get preview image

Request the path for a local preview image. the image will be downloaded if it is not already cached.

Since SDK: 20

adobe.dps.store.getPreviewImage

Parameter	Type	Description
productId	*string*	A string containing the product id of the subscription to buy.
portrait	*boolean*	An boolean indicating whether you want a portrait or landscape preview image
width	*number*	The request width of the preview (Note: This property may be ignored)
height	*number*	The request height of the preview (Note: This property may be ignored)
callback	*getPreviewImageFunction*	Function that will be called with this signature: {"result"=status, "path"=path} where status is one of "succeeded", "failed" and path is the path to the preview image

getPreviewImageFunction

Parameter	Type	Description
"result"	*string*	status is one of "succeeded" or "failed".
"path"	*string*	The local location of the desired preview image.

API Snippet

getPreviewImage

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-
transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
<title>getPreviewImage API</title>
<script type="text/javascript">
function getPreviewImage()
{
  if (window.adobedpscontextloaded)
  {
    var productId = document.getElementById("productId").value;
    var portrait = document.getElementById("portrait").checked;
    var width = document.getElementById("width").value;
    var height = document.getElementById("height").value;
    // call into the getPreviewImage API
    adobe.dps.store.getPreviewImage(productId, portrait, width, height, function(data)
      {
        document.write("<p>Results: " +
           data['result'] + " and path: " +
           data['path'] + '</p>');
      }
    );
  }
}
</script>
</head>

<body>
<p>Get Preview Image</P>
<br>ProductId: <input type="text" id="productId" value="productId"/>
<br>Portrait: <input type="checkbox" id="portrait" checked/>
<br>Width: <input type="text" id="width" value="300"/>
<br>Height: <input type="text" id="height" value="500"/>
<button type="button" onclick="getPreviewImage()">Get Preview Image</button>
</body>
</html>
```

Get receipt data

Get receipts for all purchased folios and purchased subscriptions.

Since SDK: 20

adobe.dps.store.getReceiptData

Parameter	Type	Description
callback	*getReceiptDataFunction*	Function that will be receive a dictionary of Store Product Ids -> receipt pairs in the form of {productId:receipt}. The folio product Ids can be retreived by the getFolioData() call. The receipts should be in the form the respective store server expects to receive. For subscription receipts, we only pass the receipt associated with the subscription that was most recently purchased.

getReceiptDataFunction

Parameter	Type	Description
dictionary	*getReceiptDataResult*	A dictionary containing the receipt information, as defined for a specific service.
productId	*receipt*	The receipt data for the given product.

API Snippet

getReceiptData

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-
transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
<title><%= wadl_file.awsi.name %> API</title>
<script type="text/javascript">
window.onadobedpscontextloaded = loadPage;
function loadPage()
{
  // call into the getReceiptData API
  if (window.adobedpscontextloaded)
  {
    adobe.dps.store.getReceiptData(function(dictionary)
      {
        alert("Product " + myProductID_1 + ": receipt " + dictionary[myProductID_1]);
        alert("Product " + myProductID_2 + ": receipt " + dictionary[myProductID_2]);
      }
    );
  }
}
</script>
</head>

<body>
</body>
</html>
```

Get subscription data

Get library state information for the subscriptions available in the library.

Since SDK: 20

adobe.dps.store.getSubscriptionData

Parameter	Type	Description
callback	*getSubscriptionDataFunction*	Function to receive the list of subscriptions. each subscription has these properties; productId, title, duration, price, owned, active where productId is the store product identifier title is the formatted title of the subscription duration is the formatted duration of the subscription (eg. 3 Months, 1 Week etc) price is a localized string representing the price of the folio owned is a boolean signifying that this subscription has been owned by the user active is a boolean signifying that this subscription is currently active NOTE: If no current or cached information is available, the function will return an empty list

getSubscriptionDataFunction

Parameter	Type	Description
subscriptions	*subscriptionArray*	A list containing the library state information for the subscriptions available in the library.
subscription	*subscription*	A list of subscriptions.
productId	*string*	The store product identifier.
title	*string*	The formatted title of the subscription.
duration	*string*	The formatted duration of the subscription (such as "3 Months", "1 Week").
price	*string*	A localized string representing the price of the folio.
owned	*boolean*	True if this subscription has been owned by the user.
active	*boolean*	True if this subscription is currently active.

API Snippet

getSubscriptionData

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-
transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
<title><%= wadl_file.awsi.name %> API</title>
<script type="text/javascript">
window.onadobedpscontextloaded = loadPage;
function loadPage()
{
  // call into the getSubscriptionData API
  if (window.onadobedpscontextloaded)
  {
    adobe.dps.store.getSubscriptionData(function(data)
      {
        for (subscription in data)
        {
          if (subscription.active)
            alert('Active Subscription: ' + subscription.title);
        }
      }
    );
  }
}
</script>
</head>

<body>
```

```
</body>
</html>
```

Get user info

Get auth token and user name. Either value may be empty.

Since SDK: 20

adobe.dps.store.getUserInfo

Parameter	Type	Description
callback	*getUserInfoFunction*	Function that will be called with the user info: {"authToken":currentAuthToken, "userName": currentUsername}

getUserInfoFunction

Parameter	Type	Description
dictionary	*getUserInfoResult*	A dictionary containing the returned user information.
"authToken"	*string*	The authentication token for the user.
"userName"	*string*	The user's log-in name.

API Snippet

getUserInfo

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-
transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
<title>getUserInfo API</title>
<script type="text/javascript">
function getUserInfo()
{
  // call into the getUserInfo API
  if (window.adobedpscontextloaded)
  {
    adobe.dps.store.getUserInfo(function(data)
      {
        document.write("<p>Auth Token: " +
          data['authToken'] + " and Username: " +
          data['userName'] + '</p>');
      }
    );
  }
}
</script>
</head>

<body>
<button type="button" onclick="getUserInfo()">Get User Info</button>
</body>
</html>
```

Get window orientation

Returns the orientation registered by the Viewer since the orientation provided by window.location does not work on all Android devices.

Since SDK: 20

adobe.dps.store.getWindowOrientation

Parameter	Type	Description
callback	*getWindowOrientationFunction*	Function that will receive the proper orientation in a JSON object.

getWindowOrientationFunction

API Snippet

getWindowOrientation

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-
transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
<title>getWindowOrientation API</title>
<script type="text/javascript">
window.onadobedpscontextloaded = loadPage;
function loadPage()
{
  // call into the getWindowOrientation API
  if (window.adobedpscontextloaded)
  {
    adobe.dps.store.getWindowOrientation(my_callback);
  }
}
</script>
</head>

<body>
</body>
</html>
```

Go to library

Call to send the user to the library

Since SDK: 20

adobe.dps.store.goToLibrary

API Snippet

goToLibrary

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-
transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
<title>goToLibrary API</title>
<script type="text/javascript">
function goToLibrary()
{
  if (window.adobedpscontextloaded)
  {
    // call into the goToLibrary API
    adobe.dps.store.goToLibrary();
  }
}
</script>
</head>

<body>
<button type="button" onclick="goToLibrary()">Go To Library</button>
</body>
</html>
```

Present custom dialog

Display a custom subscription popup dialog. The dialog is a transaprent full screen webview that loads a url configured in the viewer. You dismiss this custom dialog by calling the dismissCustomDialog method from within the invoked webview.

Since SDK: 20

adobe.dps.store.presentCustomDialog

API Snippet

presentCustomDialog

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-
transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
<title>presentCustomDialog API</title>
<script type="text/javascript">
function presentCustomDialog()
{
  if (window.adobedpscontextloaded)
  {
    // call into the presentCustomDialog API
    adobe.dps.store.presentCustomDialog();
  }
}
</script>
</head>

<body>
<button type="button" onclick="presentCustomDialog()">Present Custom Dialog</button>
</body>
</html>
```

Refresh entitlements

Initiate a refresh of entitlements in the viewer. Notice that there is no callback for this function. On error, will fail silently

Since SDK: 20

adobe.dps.store.refreshEntitlements

API Snippet

refreshEntitlements

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-
transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
<title>refreshEntitlements API</title>
<script type="text/javascript">
window.onadobedpscontextloaded = loadPage;
function loadPage()
{
  // call into the refreshEntitlements API
  if (window.adobedpscontextloaded)
  {
    adobe.dps.store.refreshEntitlements();
  }
}
</script>
</head>

<body>
</body>
</html>
```

Register library update complete handler

Assign a function to be called when a library update has completed. The library update could result from calling updateLibrary from the JS API. The handler assignment replaces any previously registered handlers for library update notifications.

Since SDK: 20

adobe.dps.store.registerLibraryUpdateCompleteHandler

Parameter	Type	Description
callback	*registerLibraryUpdateCompleteHandlerFunction*	Register the handler that will be called when a libraryUpdate has occurred.

registerLibraryUpdateCompleteHandlerFunction

API Snippet

registerLibraryUpdateCompleteHandler

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-
transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
<title>registerLibraryUpdateCompleteHandler API</title>
<script type="text/javascript">

function registerUpdateLibraryHandler()
{
  if (window.adobedpscontextloaded)
  {
    // call into the updateLibrary API
    adobe.dps.store.registerLibraryUpdateCompleteHandler(libraryUpdateResponder);
  }
}

function unregisterUpdateLibraryHandler()
{
  if (window.adobedpscontextloaded)
  {
    // call into the updateLibrary API
    adobe.dps.store.unregisterLibraryUpdateCompleteHandler(libraryUpdateResponder);
  }
}

function libraryUpdateResponder()
{
        alert("update library complete");
}

function updateLibrary()
{
  if (window.adobedpscontextloaded)
  {
    // call into the updateLibrary API
    adobe.dps.store.updateLibrary();
  }
}
</script>
</head>

<body>
<br>Register an update library complete handler<button type="button" onclick=
"registerUpdateLibraryHandler()">Register Handler</button>
<br>Unregister an update library complete handler<button type="button" onclick=
"unregisterUpdateLibraryHandler()">Unregister Handler</button>
<br><button type="button" onclick="updateLibrary()">Update Library</button>
```

```
</body>
</html>
```

Restore purchases

Restore purchases.

Since SDK: 20

adobe.dps.store.restorePurchases

API Snippet

restorePurchases

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-
transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
<title>restorePurchases API</title>
<script type="text/javascript">
window.onadobedpscontextloaded = loadPage;
function loadPage()
{
  // call into the restorePurchases API
  if (window.adobedpscontextloaded)
  {
    adobe.dps.store.restorePurchases();
  }
}
</script>
</head>

<body>
</body>
</html>
```

Set user info

Assigns the auth token and user name for the viewer to persist. Only the authToken value is used to determine entitlement. The username is additional data availbale to the JS API and has no relation to user login credentials. To log a user out call this function wiht an empty authToken value. Notice that there is no callback for this function invalid values for either authToken or userName will make this function fail silently.

Since SDK: 20

adobe.dps.store.setUserInfo

Parameter	Type	Description
authToken	*string*	String containing the auth token or undefined in order to sign out.
userName	*string*	String containing the user name used to sign in or undefined to clear the persisted user name.

API Snippet

setUserInfo

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-
transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
<title>setUserInfo API</title>
<script type="text/javascript">
function setUserInfo()
{
  if (window.adobedpscontextloaded)
  {
    var authToken = document.getElementById("authToken").value;
    var userName = document.getElementById("userName").value;
    // call into the setUserInfo API
    adobe.dps.store.setUserInfo(authToken,userName);
  }
}
</script>
</head>

<body>
<p>Set User Info</p>
<input type="text" id="authToken" value="authToken"/>
<input type="text" id="userName" value="userName"/>
<button type="button" onclick="setUserInfo()">Set User Info</button>
</body>
</html>
```

Unregister library update complete handler

Removes the assigned handler so it will no longer be called when a library update completes.

Since SDK: 20

adobe.dps.store.unregisterLibraryUpdateCompleteHandler

API Snippet

unregisterLibraryUpdateCompleteHandler

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-
transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
<title>unregisterLibraryUpdateCompleteHandler API</title>
<script type="text/javascript">

function registerUpdateLibraryHandler()
{
  if (window.adobedpscontextloaded)
  {
    // call into the updateLibrary API
    adobe.dps.store.registerLibraryUpdateCompleteHandler(libraryUpdateResponder);
  }
}

function unregisterUpdateLibraryHandler()
{
  if (window.adobedpscontextloaded)
  {
    // call into the updateLibrary API
    adobe.dps.store.unregisterLibraryUpdateCompleteHandler(libraryUpdateResponder);
  }
}

function libraryUpdateResponder()
{
        alert("update library complete");
}

function updateLibrary()
{
  if (window.adobedpscontextloaded)
  {
    // call into the updateLibrary API
    adobe.dps.store.updateLibrary();
  }
}
</script>
</head>

<body>
<br>Register an update library complete handler<button type="button" onclick=
"registerUpdateLibraryHandler()">Register Handler</button>
<br>Unregister an update library complete handler<button type="button" onclick=
"unregisterUpdateLibraryHandler()">Unregister Handler</button>
<br><button type="button" onclick="updateLibrary()">Update Library</button>
</body>
</html>
```

Update library

Initiate library update. The response will not be sent through a callback function but through the libraryUpdateComplete since a library update can be initiated by different actions and the interface should be notified when this happens.

Since SDK: 20

adobe.dps.store.updateLibrary

API Snippet

updateLibrary

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-
transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
<title>updateLibrary API</title>
<script type="text/javascript">
function goToLibrary()
{
  if (window.adobedpscontextloaded)
  {
    // call into the updateLibrary API
    adobe.dps.store.updateLibrary();
  }
}
</script>
</head>

<body>
<button type="button" onclick="updateLibrary()">Update Library</button>
</body>
</html>
```

View folio

Download or view the folio with the given product ID. If the folio is already on the device the viewer will open it for viewing otherwise the viewer will start the download of the folio. In either case the store web view will get closed immediately. Notice that there is no callback for this function. If the view/download fails the error is handled by the viewer in whatever way is appropriate.

Since SDK: 20

adobe.dps.store.viewFolio

Parameter	Type	Description
folioProductId	*string*	A string containing the product id of the folio to view or download.

API Snippet

viewFolio

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-
transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
<title>viewFolio API</title>
<script type="text/javascript">
function viewFolio()
{
  if (window.adobedpscontextloaded)
  {
    var productId = document.getElementById("viewfolio").value;
    // call into the viewFolio API
    adobe.dps.store.viewFolio(productId);
  }
}
</script>
</head>

<body>
<p>View Folio</p>
<input type="text" id="viewfolio" value="folioProductId"/>
<button type="button" onclick="viewFolio()">View Now</button>
</body>
</html>
```

Folio States

Constants that describe the library state of a folio that an API function is attempting to operate on. The state tells you whether the folio can be purchased, downloaded, viewed, or updated, and whether such an operation is aleady in process for it.

Name	Value	Description