



Introduction to ContextHub in AEM 6.4

Artur Kudlacz | developer

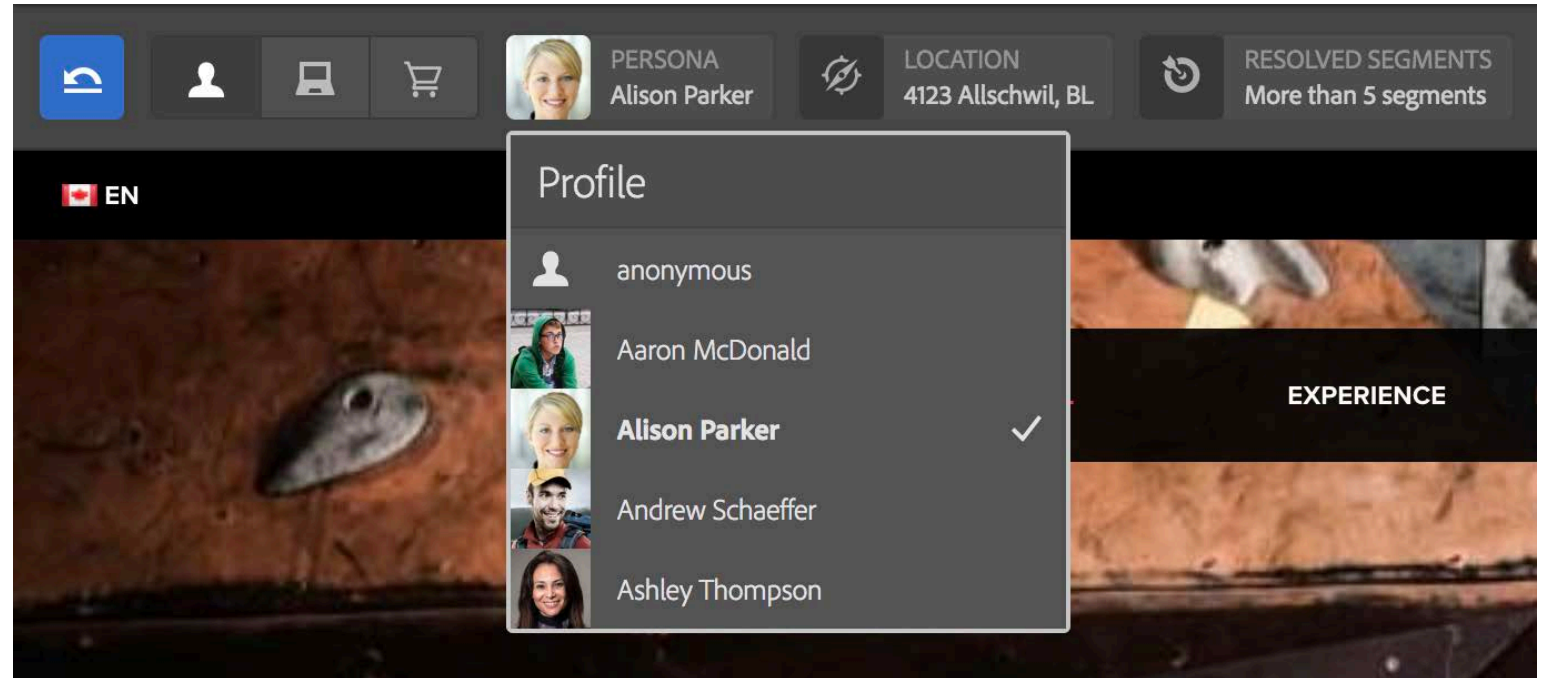
ClientContext

- represents a dynamically assembled collection of user data related to **current page** and **visitor**
- data collections are called **stores**
- data, browser, persona, location **simulation** ability
- **segmentation** module allows to build conditions (segments) based on data collected in the stores
- **targeting** engine allows to create content variant – segment pairs used to show best matching content
- comes with ui config and segment editor
- generic stores, eventing, public API
- **static** client library
- **classic ui**, obsolete, performance issues

The screenshot displays a user profile for Alison Parker. The profile includes a profile picture, email address (aparker@geometrix.info), name (ALISON PARKER), gender (female), and birth date (27 Feb 1992, age 19). A note indicates that the email address is used as the user ID. The profile also shows a creation date (8 Sep 2010 3:13 PM) and product size information (No productSize, No productSizeFootwear). Below the profile is a section for 'Alison Parker's friends and followers (social graph)' with a row of profile pictures. The browser information section shows Firefox 10 or higher, Mac OS X, and various browser settings like 'No keywords', '2560x1440', 'Desktop', and 'Firefox'. The location section shows a map of Ireland with a pin in Dublin, and text indicating 'IE IRELAND County Dublin'. At the bottom, there are tags for 'biking hiking running' and 'summer geometrix-outdoors left'.

ContextHub

- **touch ui**
- **dynamic** client library
- **modes** and **modules** to categorize data in ui
- **event throttling** / optimization
- extensible
- **feature parity** with ClientContext
- delivered in reduced, minified and compressed set of resources (kernel, ui and styles)
- built-in **diagnostics** page and **logging**
- modularized and well commented code



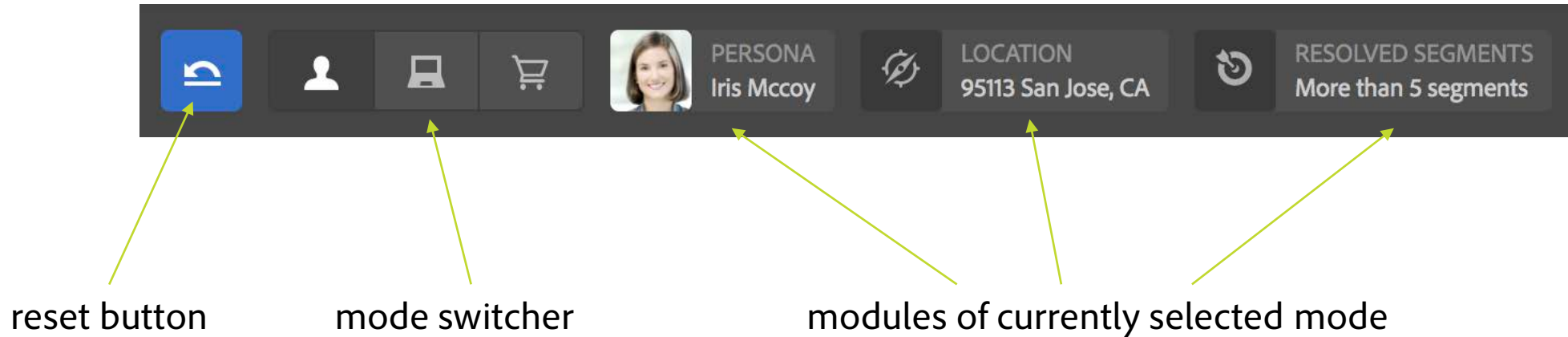


Demo

#AdobeRemix
Lauro Samblás

Mode

- allows to group modules displayed in ContextHub ui
- comes with a **name** and **icon**
- can be **disabled** (and not displayed anymore)
- **mode switcher** appears if more than one mode is defined in ContextHub configuration



Store

- clientlib based (implementation location does not matter)
- implements specific **storeType**
- where **storeType**: **category.storeName**, for example:
 - **storeType**: **contexthub.location**
 - clientlib name: **contexthub.store.contexthub.location**
- multiple implementations of given **storeType** can exist (best candidate is used)
- implementation candidate comes with condition determining if store can be used
- implementation extends one of **generic stores**:
 - SessionStore
 - PersistedStore
 - JSONPStore
 - PersistedJSONPStore

Store implementation

```
var defaultConfig = {  
  service: {  
    host: '...',  
    port: 1234,  
    path: '/foo'  
  },  
  initialValues: { ... }  
};
```

```
var MyStore = function(name, config) {  
  this.config = $.extend(true, {}, defaultConfig, config);  
  this.init(name, this.config);  
};
```

```
(...) /* other functions: MyStore.prototype.myFunction = function() { ... }; */
```

```
ContextHub.Uutils.inheritance.inherit(MyStore, ContextHub.Store.PersistedJSONPStore);
```

```
/* store must be a clientlib: "contexthub.store.storeType", in this example: "contexthub.store.contexthub.xyz" */
```

```
ContextHub.Uutils.storeCandidates.registerStoreCandidate(MyStore, 'contexthub.xyz', 0, function(store) {  
  return (...); /* true or false - depending on the result of custom condition */  
});
```

Store API

```
> var geolocation = ContextHub.getStore('geolocation');  
> geolocation.getTree();  
{defaultLocation: {...}, latitude: 47.555115, longitude: 7.590184, address: {...}, addressDetailsOf: {...}}
```

```
/* getting items */
```

```
> geolocation.getItem('address/country');  
"Switzerland"  
> ContextHub.getItem('geolocation/address');  
{country: "Switzerland", countryCode: "CH", city: "Basel", street: "Streitgasse", streetNumber: "5", ...}
```

```
/* setting items */
```

```
> geolocation.setItem('address/city', 'Zurich');  
> geolocation.getItem('address/city');  
"Zurich"
```

```
/* removing items */
```

```
> geolocation.removeItem('address');  
> geolocation.getItem('address/countryCode');  
null
```


Module

- clientlib based (implementation location does not matter)
- implements specific **moduleType**
- where **moduleType**: **category.moduleName**, for example:
 - moduleType: **contexthub.location**
 - clientlib name: contexthub.module.**contexthub.location**
- can be represented as one of three figures
 - minimized (not clickable)
 - expandable (clickable, popover appears)
 - fullscreen
- default config (part of implementation) can be overlaid in ui module configuration
- can be codeless – defined only as a config that is basing on generic module renderer

Module implementation

```
var defaultConfig = {  
  icon: 'coral-icon-xyz',  
  title: 'Popover title',  
  clickable: true,  
  storeMapping: { myStore: 'storeName' },  
  key: '/store/myStore/xyz',  
  template:  
    '<p class="contexthub-module-line1">{{i18n "Module title"}}</p>' +  
    '<p class="contexthub-module-line2">{{myStore.keyName}}</p>'  
};
```

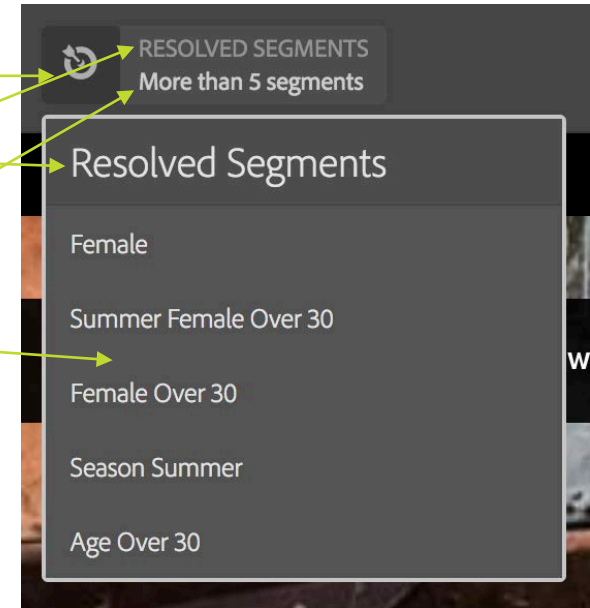
```
var MyModule = function() {};
```

```
MyModule.prototype.render = function(module) {  
  var config = $.extend(true, {}, defaultConfig, module.config);  
  (...)  
};
```

```
ContextHub.Uutils.inheritance.inherit(MyModule, ContextHub.UI.BaseModuleRenderer);
```

```
/* module must be a clientlib: "contexthub.module.moduleType", in this example: "contexthub.module.contexthub.xyz" */
```

```
ContextHub.UI.ModuleRenderer('contexthub.xyz', new MyStore());
```



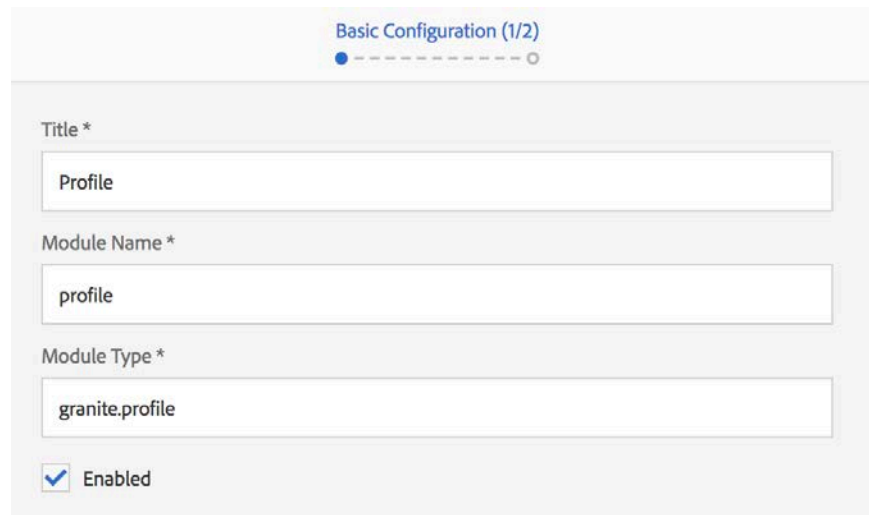
Module customization

Following functions can be overlaid in order to customize module rendering:

- **MyModule**.prototype.render(module) – renders a module
- **MyModule**.prototype.updatePopoverContent(popover, content) – updates content of popover
- **MyModule**.prototype.onClickModuleIcon(module, event) – called when module icon was clicked
- **MyModule**.prototype.onClickModuleDescription(module, event) – called when module container was clicked (by default it opens a popover)
- **MyModule**.prototype.moduleEditing(module, event, config) – renders a form inside of popover that allows to edit data of the store mapped to a given module
- **MyModule**.prototype.onFullscreenClicked(module, event) – called when full-screen icon was clicked
- **MyModule**.prototype.onListItemClicked(module, position, data, event) – called when one of items inside of popover's list gets clicked

Creating ContextHub config

- navigate to <http://localhost:4502/libs/granite/cloudsettings/ui.html/conf/<your-brand>>
- create a new configuration container
- inside of the container create a new empty configuration
- create store entries
- create mode entries (optional)
- create module entries



Basic Configuration (1/2)

Title *

Profile

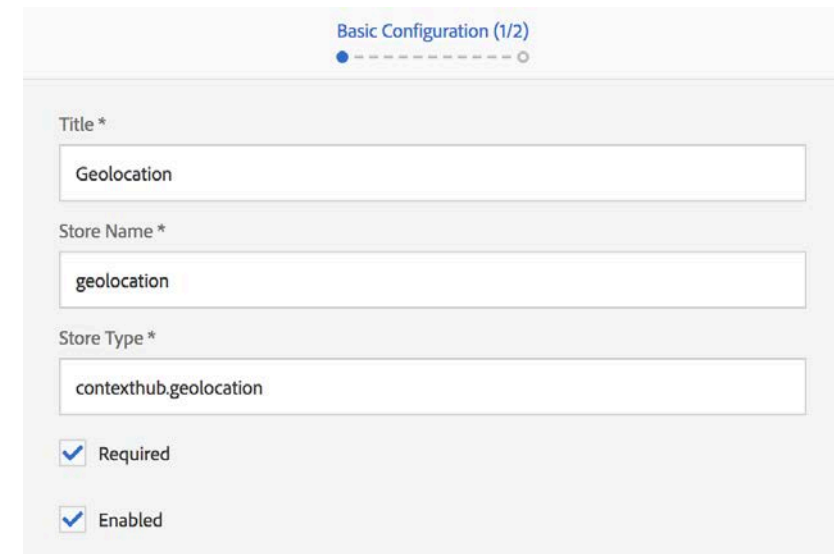
Module Name *

profile

Module Type *

granite.profile

Enabled



Basic Configuration (1/2)

Title *

Geolocation

Store Name *

geolocation

Store Type *

contexthub.geolocation

Required

Enabled

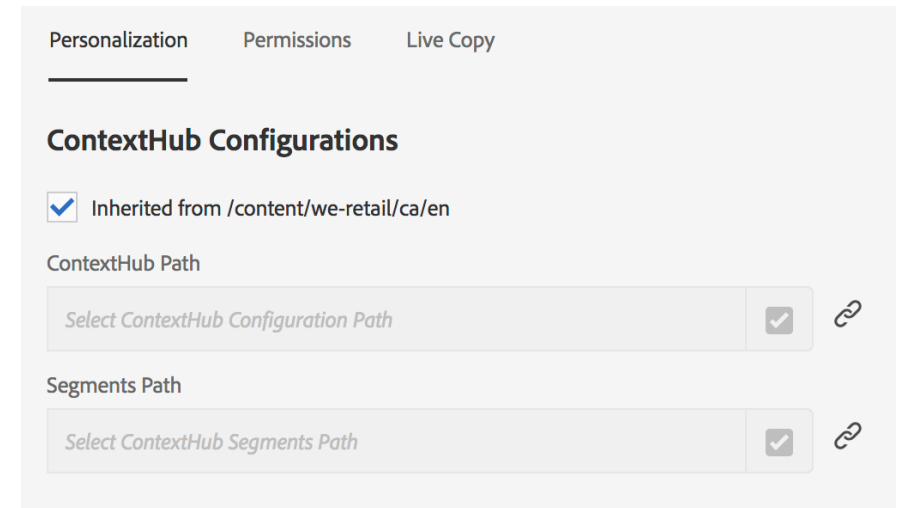
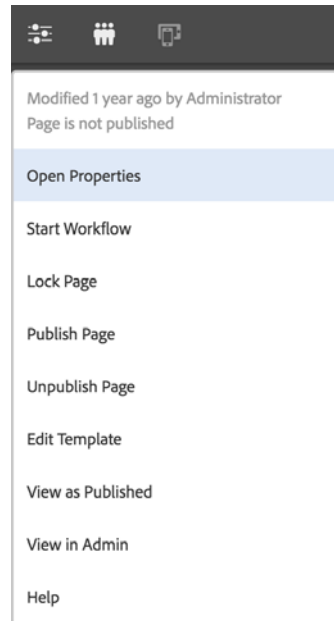
Adding ContextHub on a page

ContextHub component inclusion:

- HTL
`<sly data-sly-resource="${ 'contexthub' @ resourceType='granite/contexthub/components/contexthub' }"/>`
- JSP
`<sling:include path="contexthub" resourceType="granite/contexthub/components/contexthub"/>`

Configuration:

- navigate to a page
- open **Properties**
- switch to **Personalization** tab
- select **config** and **segments** path
- save

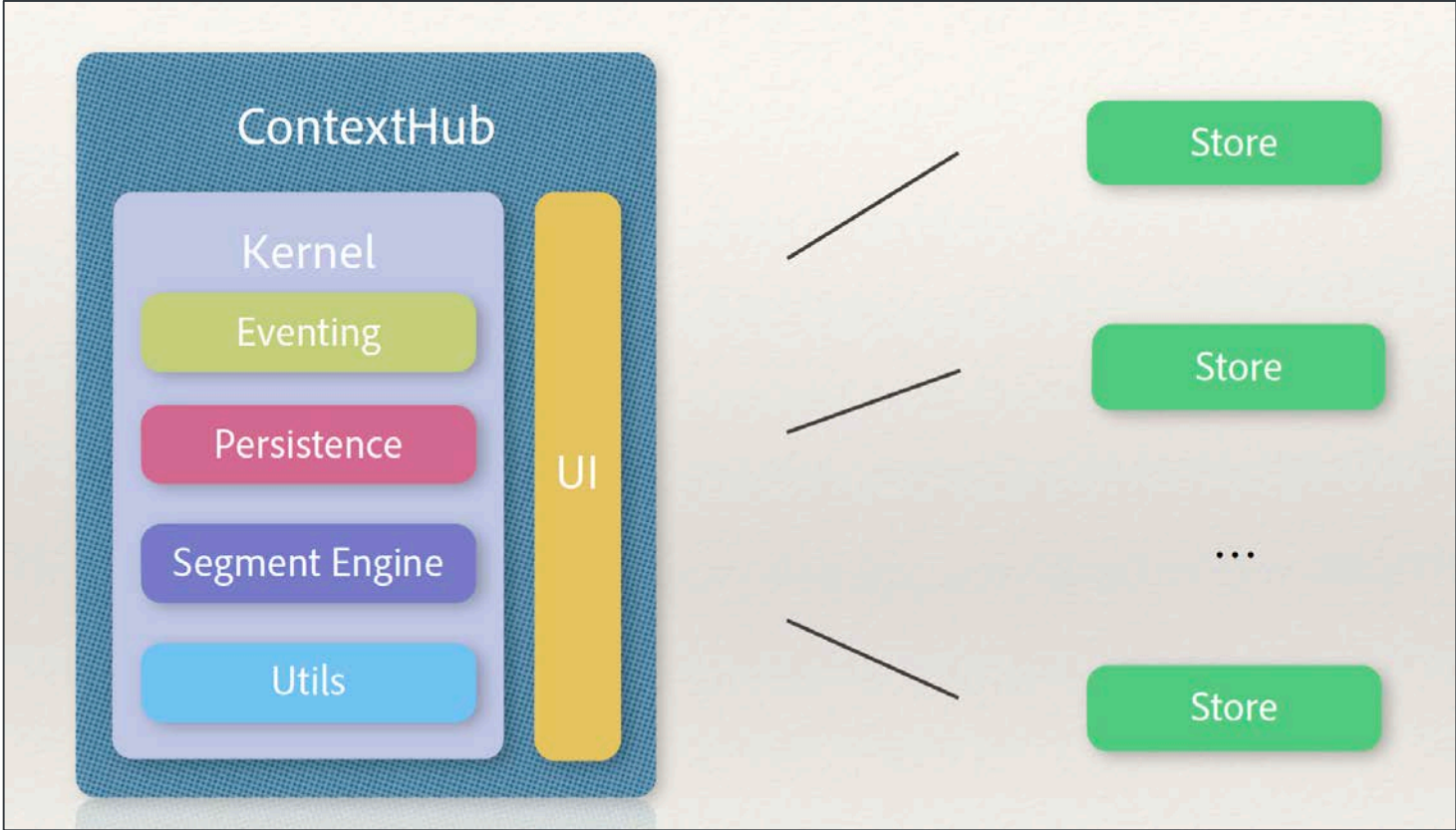


High level architecture

Main modules:

- eventing
- persistence
- segment engine
- generic stores
- ui

Publish instance does not include ContextHub ui.



Eventing

Used by:

- stores to indicated data updates in their persistence
- UI to announce different actions (UI ready, mode / module registration, container opened / collapsed, etc)
- segment engine (segments loaded, un / resolved)

Some features:

- **throttling** – by default events are not fired more often than every 16 ms, this allows to buffer them and optimize in case event with same namespace comes with few different values (only last will be fired)
- event **pausing** / **resuming** – when performing large data updates it's advised to pause eventing before updates and resume when everything is ready (events get optimized here as well)
- event **disabled** / **enabled** – when eventing is disabled nothing gets to the queue
- both pausing / disabling can be done globally or per store

Event handlers should use constants (ContextHub.Constants.EVENT_*) to avoid hardcoding event names.

Eventing

```
/* triggering an event */
ContextHub.eventing.trigger('event-name', { foo: 'bar' });

/* 2018-09-19 08:07:06.105 [event] ch-event-name - Object {
  executeAt: ..., data: Array[1], keys: Object, event: 'ch-event-name', channel: 'event-name' } */

/* creating event handler */
ContextHub.eventing.on(ContextHub.Constants.EVENT_STORE_UPDATED + ':profile', function(event, data) { /* handler */ });

/* removing event handler */
ContextHub.eventing.off(ContextHub.Constants.EVENT_STORE_UPDATED + ':profile');

/* using store's instance to attach / detach the handler */
var profile = ContextHub.getStore('profile');

if (profile) {
  /* binds the handler */
  profile.onUpdate('handler-name', function(event, data) { /* handler */ });

  /* unbinds the handler */
  profile.onUpdate('handler-name');
}
```

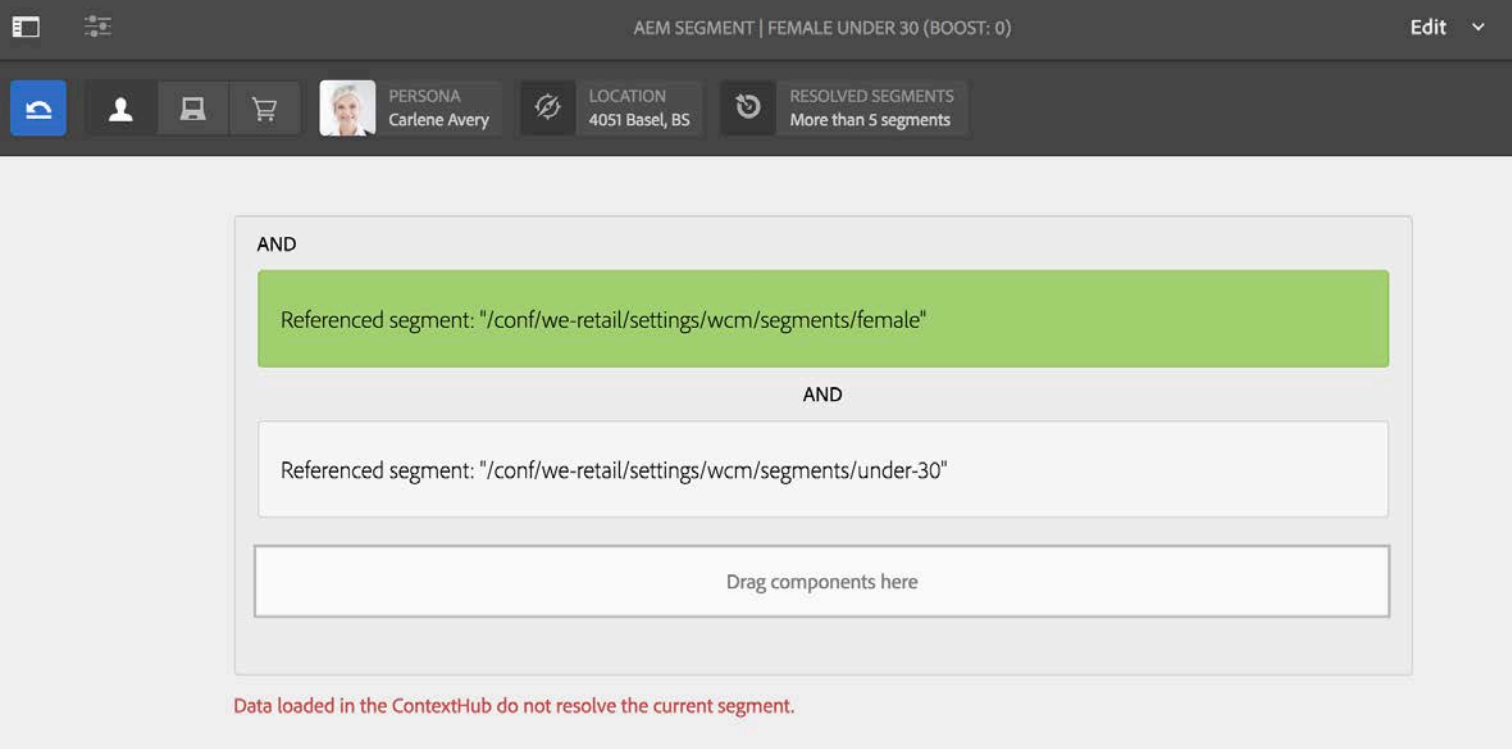

Persistence

Available persistence modes:

- HTML5 localStorage (**ContextHub.Utils.Persistence.Modes.LOCAL**) – storage of around ~5MB, data has no expiration and is available after reloading page or closing/re-opening browser
- HTML5 sessionStorage (**ContextHub.Utils.Persistence.Modes.SESSION**) – similar to localStorage, except that data gets cleared when the page session ends
- Cookie (**ContextHub.Utils.Persistence.Modes.COOKIE**) – holds up to 4KB of data (this storage type should be avoided - very low capacity, is sent to the server within every browser request)
- window.name (**ContextHub.Utils.Persistence.Modes.WINDOW**) – last resort (should be avoided as well) when other storages are not supported
- it's possible to implement own persistence mode

Segment engine

- allows to build conditions that gets resolved into a boolean values
- available condition traits:
 - property – property
 - property – segment reference
 - property – script reference
 - property – value
 - segment reference
 - script reference
- trait values can be compared using following operators:
>, >=, <, <=, ==, !=
- condition blocks can be chained by “and” & “or” operators



The screenshot displays the Adobe Segment Engine interface for configuring a rule. The title bar reads "AEM SEGMENT | FEMALE UNDER 30 (BOOST: 0)" with an "Edit" dropdown. Below the title bar, there are several trait cards: "PERSONA" (Carlene Avery), "LOCATION" (4051 Basel, BS), and "RESOLVED SEGMENTS" (More than 5 segments). The main workspace shows a rule structure with an "AND" operator connecting two conditions. The first condition is highlighted in green and reads "Referenced segment: '/conf/we-retail/settings/wcm/segments/female'". The second condition is in a white box and reads "Referenced segment: '/conf/we-retail/settings/wcm/segments/under-30'". Below these conditions is a white box with the text "Drag components here". At the bottom of the workspace, a red warning message states: "Data loaded in the ContextHub do not resolve the current segment."

Generic stores

Available generic stores:

- ContextHub.Store.**SessionStore** – in-memory persistence, that gets cleared when leaving a page
- ContextHub.Store.**PersistedStore** – uses **ContextHub.Utils.Persistence.Modes.LOCAL** persistence by default
- ContextHub.Store.**JSONPStore** – in-memory persistence, data is pulled from external jsonp service
- ContextHub.Store.**PersistedJSONPStore** – similar to JSONPStore, except data is persisted in local storage, thus it's still available after leaving a page

To base on one of generic stores, one have to extend his implementation by calling:

```
ContextHub.Utils.inheritance.inherit(MyStore, ContextHub.Store.<generic-store>);
```

Resources and clientlibs

- **ContextHub resources** are compacted into 3 requests
 - **kernel.js**
 - **ui.js** (loaded only on author)
 - **styles.css** (loaded only on author)
- minified, compressed and cached on client-side (using ETag as cache validation)
- set of stores and modules is determined by configuration
- disabled modes, modules and stores are not part of the response
- resources also contain json configs of all the stores and modules
- third party libraries included: jquery, handlebars

Resources: kernel (author and publish)

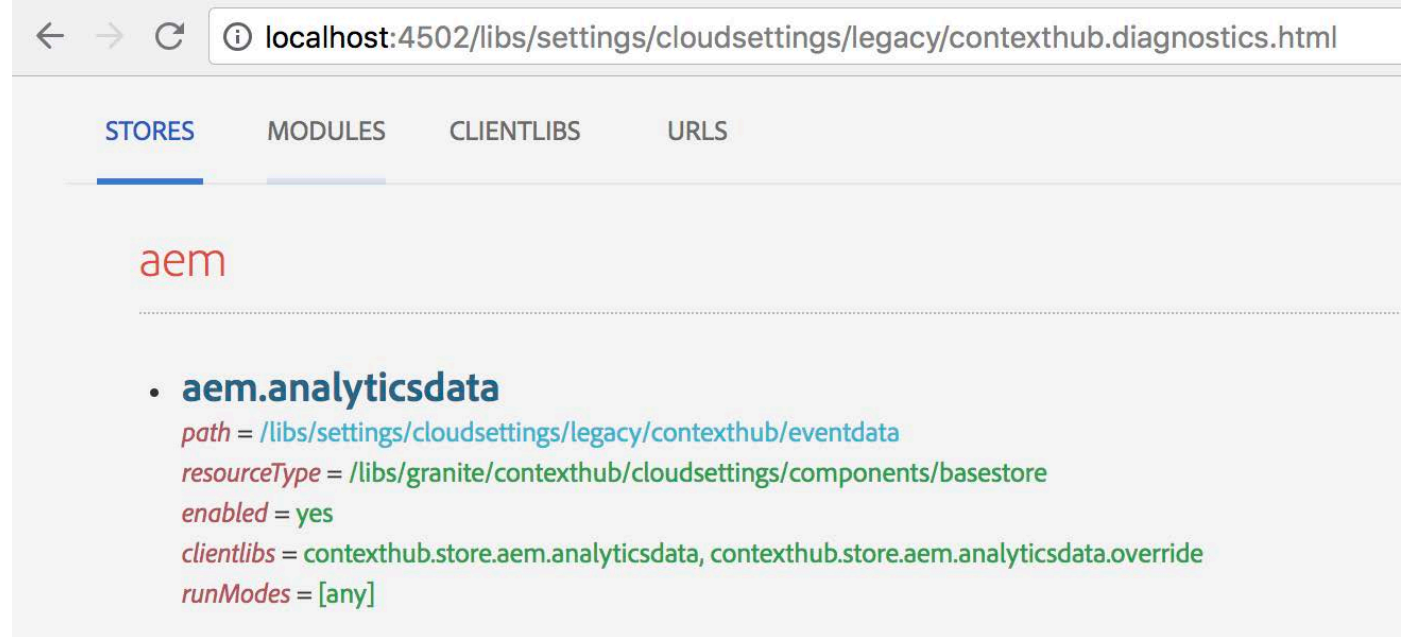
- *config*
- contexthub.utils
- contexthub.kernel
- contexthub.generic-stores
- contexthub.segment-engine
- contexthub.segment-engine.operators
- contexthub.segment-engine.scripts
- contexthub.segment-engine.page-interaction
- contexthub.store.contexthub.<sN>
- contexthub.finalize
- contexthub.ui.inject

Resources: ui – js and css (author only)

- *config*
- contexthub.ui
- contexthub.ui.initialization
- contexthub.ui.default
- contexthub.module.<mN>
- contexthub.ui.finalization

Diagnostics page

- lists all stores, modes, modules which are part of given configuration
- lists clientlibs used to build ContextHub resources
- lists important URLs
- diagnostics page is available if **debug** flag is **enabled** in ContextHub config



localhost:4502/libs/settings/cloudsettings/legacy/contexthub.diagnostics.html

STORES MODULES CLIENTLIBS URLs

aem

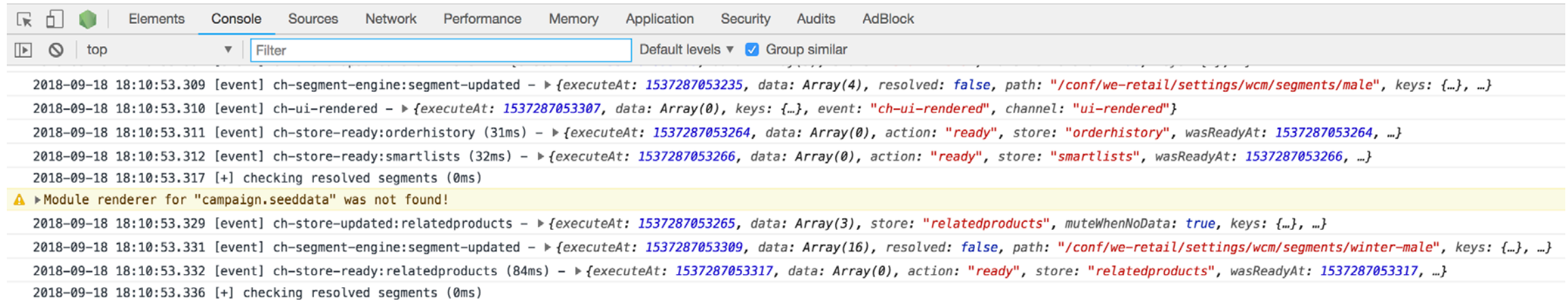
- **aem.analyticsdata**
 - path = /libs/settings/cloudsettings/legacy/contexthub/eventdata
 - resourceType = /libs/granite/contexthub/cloudsettings/components/basestore
 - enabled = yes
 - clientlibs = contexthub.store.aem.analyticsdata, contexthub.store.aem.analyticsdata.override
 - runModes = [any]

To access diagnostics page:

- <http://localhost:4502/path/to/config.diagnostics.html>
- <http://localhost:4502/libs/...dsettings/legacy/contexthub.diagnostics.html>

Logging

- debug information available in browser's console
- available if **debug** flag is **enabled** in ContextHub config
(debug = true @ /path/of/config – for example: /libs/settings/cloudsettings/legacy/contexthub)



```
2018-09-18 18:10:53.309 [event] ch-segment-engine:segment-updated - ▶ {executeAt: 1537287053235, data: Array(4), resolved: false, path: "/conf/we-retail/settings/wcm/segments/male", keys: {...}, ...}
2018-09-18 18:10:53.310 [event] ch-ui-rendered - ▶ {executeAt: 1537287053307, data: Array(0), keys: {...}, event: "ch-ui-rendered", channel: "ui-rendered"}
2018-09-18 18:10:53.311 [event] ch-store-ready:orderhistory (31ms) - ▶ {executeAt: 1537287053264, data: Array(0), action: "ready", store: "orderhistory", wasReadyAt: 1537287053264, ...}
2018-09-18 18:10:53.312 [event] ch-store-ready:smartlists (32ms) - ▶ {executeAt: 1537287053266, data: Array(0), action: "ready", store: "smartlists", wasReadyAt: 1537287053266, ...}
2018-09-18 18:10:53.317 [+] checking resolved segments (0ms)
⚠ ▶ Module renderer for "campaign.seedata" was not found!
2018-09-18 18:10:53.329 [event] ch-store-updated:relatedproducts - ▶ {executeAt: 1537287053265, data: Array(3), store: "relatedproducts", muteWhenNoData: true, keys: {...}, ...}
2018-09-18 18:10:53.331 [event] ch-segment-engine:segment-updated - ▶ {executeAt: 1537287053309, data: Array(16), resolved: false, path: "/conf/we-retail/settings/wcm/segments/winter-male", keys: {...}, ...}
2018-09-18 18:10:53.332 [event] ch-store-ready:relatedproducts (84ms) - ▶ {executeAt: 1537287053317, data: Array(0), action: "ready", store: "relatedproducts", wasReadyAt: 1537287053317, ...}
2018-09-18 18:10:53.336 [+] checking resolved segments (0ms)
```


Since AEM 6.4

- we are using `/conf/<tenant>/settings/wcm/segments` to store segments
- we are using `/conf/<tenant>/settings/cloudsettings/<container>/contexthub` for configurations
- sample config moved from `/etc/settings/cloudsettings/default/contexthub` to `/libs/settings/cloudsettings/legacy/contexthub`
- sample segments moved from `/etc/segmentation/contexthub` to `/conf/we-retail/settings/wcm/segments`
- segment generation and resolving performance improvements (`/path/to/segments.seg.js`)

Links

- <http://localhost:4502/etc/cloudsettings/default/contexthub.html>
- <http://localhost:4502</path/to/config>/contexthub.diagnostics.html>
- <http://localhost:4502/etc/cloudsettings.kernel.js</path/to/config>/contexthub>
- <http://localhost:4502/etc/cloudsettings.ui.js</path/to/config>/contexthub>
- <http://localhost:4502/etc/cloudsettings.styles.css</path/to/config>/contexthub>
- <http://localhost:4502/etc/cloudsettings.config.kernel.js </path/to/config>/contexthub>
- http://localhost:4502</path/to/config>/contexthub.config_kernel.json
- <http://localhost:4502/etc/cloudsettings.config.ui.js </path/to/config>/contexthub>
- http://localhost:4502</path/to/config>/contexthub.config_ui.json



Questions and answers



Adobe