Web-UI (cs.web)

Release 15.3.0.6

CONTACT Software

Nov 01, 2018

Contents

1 Release Notes for 15.3 **1**

1.1 New backend dialog hook function . . . 1
1.2 Dialog Hooks for buttons . . . 1
1.3 JavaScript library reorganization . . . 1
1.4 React.js Update . . . 1
1.5 Replacement of Date Picker in Calendar component . . . 2
1.6 Changes to Button Styles . . . 2
1.7 Frontend Framework for Operations changed . . . 2
1.8 Changes for State Color Calculation . . . 2

2 Release Notes for 15.2.1 **3**

2.1 Operation Frontend API changes . . . 3
2.2 Changes in Frontend Components . . . 3
2.3 New SVG Icons . . . 4
2.4 Internationalization . . . 4
2.5 Build tool updates . . . 4
2.6 New version of React-Router for Frontend Routing . . . 5
2.7 Licensing Change . . . 6

Index **9**

CHAPTER 1

Release Notes for 15.3

1.1 New backend dialog hook function

The function `set_next_dialog` has been added to backend dialog hooks running `::PRE_SUBMIT::` calls. Calling this function allows replacing the currently used dialog.

1.2 Dialog Hooks for buttons

The predefined hook `EmulateLegacyDialogButton` has to be used to handle mask buttons with the Elements UI. The use of `EmulateLegacyDialogItemChange` is no longer possible for Buttons.

1.3 JavaScript library reorganization

Most of the external JavaScript libraries that are used are now bundled into the `cs-web-components-external` library. For `react`, `react-bootstrap`, `immutable` and `momentjs` nothing needs to change in other JavaScript code.

However, `jquery` or `underscore` are no longer automatically available in all applications. If a library uses one of these, there are two options:

- Change the implementation, so that the dependency is no longer needed. What exactly needs to be done depends on which library functionality was utilized.
- Add an explicit dependency to one or both in a configuration file, or load them through `cs.web.components.base.main.BaseApp.include` calls in a view function.

1.4 React.js Update

The React library has been updated from major version 15 to 16. React components that declare their property types using `React.PropTypes` have to be updated. This usage is deprecated since React 15.5, and no longer works under React 16. The replacement is to use `PropTypes` from the `cs-web-components-external` library.

1.5 Replacement of Date Picker in Calendar component

The previous date picker for the `Calendar` component is obsolete and replaced by a new one. This replacement brings some minor style changes. The clock, which might be displayed as read-only previously under certain conditions, is removed for now.

1.6 Changes to Button Styles

The `buttonStyle` parameter value `default` is deprecated. For non-semantic (i.e. uncolored) buttons use the value `outline`. For an `IconButton` with subordinate functionality use the value `auxiliary`.

1.7 Frontend Framework for Operations changed

The framework for executing operations in the frontend has been reimplemented using Redux Saga. This requires porting of all user code that:

- implements custom OperationHandler and OperationTrigger components
- uses the Redux Thunk-based API directly
- Uses ConfiguredForm directly

For information on how to port your existing code, see the in-depth guide in the Web Developer Manual

1.8 Changes for State Color Calculation

The implementation of `cs.web.components.ui_support.StateColors.get_state_colors` has changed in a way that you will get the state colors as a CSS string, like `#00FF00` instad of a hex representation of the number. This affects the internal REST call `/internal/uisupport/state_colors/<livecycle_class>`.

CHAPTER 2

Release Notes for 15.2.1

2.1 Operation Frontend API changes

2.1.1 Redux Action `startOperation`

Due to the introduction of OperationHandler Extensions the redux action `startOperation` defined in `cs-web-components-base` has a new signature:

```
startOperation(
    instanceName,
    operationInfo,
    operationParameters,
    extensionParameters,
    submitHandler
)
```

If you call this API directly and pass a submit handler definition, make sure you update your calls accordingly by passing an empty object for the fourth parameter.

2.1.2 `OperationTrigger`: `handlerName` evaluation

In cs.web 15.2.0 the startOperation call evaluated the provided `handlerName` property only if no handler was set by the Web-UI operation definition. This was changed: this property – if set – now overrides the `handlerName` provided by the backend.

2.2 Changes in Frontend Components

2.2.1 RelatedObjects

The `RelatedObjects` component now actively fetches the data it requires from the backend. This allows for more flexible page configurations, but also requires at least Elements Version 15.2 SL 26.

2.2.2 Attributes

The `Attributes` component has been changed to retrieve its configuration from a DisplayContext named `attributes`. Formerly, the component could only be used on an ObjectPage for the main object, or a ClassPage beside the search result. Now, the only requirement for the component is that it has to be provided with a `contextObject` property. For each class this component is used for, a form must be configured in the DisplayContext for that class.

2.3 New SVG Icons

The PNG icons are replaced by vector-based SVG icons in this version. The new SVG icons are rebranded and redesigned. And the SVG icons can be scaled up without looking pixelated.

2.3.1 Migrating to the web

The PNG icons are still in use in the PC client, therefore it is important to create corresponding SVG icons when a customized application is migrated to the web frontend. Otherwise in certain components the icons may be rendered inappropriately.

The Elements icon library under cs.designsystem/cs/designsystem/resources/icon/svg provides an essential list of icon candidates. Free vector graphic editor such as Inkscape, Vectr can be used to modify and create new icons.

Give the SVG icon the same name as the PNG icon, if the PNG icons are still in use in the PC client. The SVG icons will overwrite the PNG icons in the web.

2.4 Internationalization

The previous mechanism for internationalization used JSON files (typically named `messages.json`) to store labels available to the CONTACT Elements Web UI. These files are no longer supported, see the Web UI Development book for details.

2.5 Build tool updates

2.5.1 NPM replaced by yarn

`npm` was replaced by `yarn` due to build and performance issues. Yarn accepts mostly the same arguments as npm and can be used as a drop-in replacement. The node_modules which yarn produces are equivalent to those created by npm.

2.5.2 node_modules dependency management

The base dependencies used by all web enabled packages, e.g. webpack, babel, etc, are now retrieved from the node_modules in the instance directory. These are installed and updated by the new command `webmake devupdate`. See web_ui_make_build_process for further information.

2.5.3 Webpack config

Before this update, each CDB package contained a `webpack.common.config.js` which was referenced by the `webpack.config.js` files in the web app bundles. Across the packages, these files werde almost identical, which led to the following change. There is only one `webpack.common.config.js` contained in `cs.web`. This file can be referenced by a `webpack.config.js` file using

```
const configModule = require(process.env.WEBPACK_CONFIG_COMMON);
```

The environment variable `WEBPACK_CONFIG_COMMON` is set by webmake to point to the `cs.web` config.

An example configuration can be found in the directory `templates/app/base_impl/js/` in `cs.web`, called `webpack.config.js.template`.

2.5.4 apps.json

This file is now a list of relative paths to the apps. There are no IDs or componentNameSpaces defined anymore. The namespace of an app is defined in a file called `namespace.json` next to the `webpack.config.js` inside an app directory.

You can use `webmake create` to create a new app and check out how the layout of the app is meant to be structured.

2.6 New version of React-Router for Frontend Routing

The 3rd party library `react-router` gets updated from version 3.x to 4.x. There are some breaking changes in the new version, therefore the frontend routing using `react-router` must be migrated.

For instance, the usage of `Router` component can be changed from

```
// in cs.web 15.2.0
import {Router, Route, IndexRoute, browserHistory} from 'cs-web-components-
↪externals';

const MyRouter = (
    <Router history={browserHistory}>
        <Route path={basePath} component={FrameComponent}>
            <IndexRoute component={PageContent}/>
            <Route path="**" component={NoRouteError}/>
        </Route>
        <Route path="**" component={NoRouteError}/>
    </Router>
);
```

to

```
// in cs.web 15.2.1
import {ReactRouter, browserHistory} from 'cs-web-components-externals';

const MyRouter = (
    <FrameComponent>
        <ReactRouter.Router history={browserHistory}>
            <ReactRouter.Switch>
                <ReactRouter.Route path={basePath} component={PageContent} />
                <ReactRouter.Route component={NoRouteError} />
            </ReactRouter.Switch>
        </ReactRouter.Router>
    </FrameComponent>
);
```

The `location` prop passed by `Router` component now has no `query` object anymore. The usage should be changed from

```
// in cs.web 15.2.0
const searchText = this.props.location.query.searchtext;
```

to

```
// in cs.web 15.2.1
import {parseQuery} from 'cs-web-components-base';

...
const searchText = parseQuery(this.props.location).searchtext;
```

For more details please look at the standard components using routing in `cs-web-components-base`, or read the official migration guide Migrating from v2/v3 to v4.

2.7 Licensing Change

Due to a mistake in the license feature allocation, users of the dashboard must install new license files.

List of Figures

List of Tables

Index

E

environment variable
    WEBPACK_CONFIG_COMMON, 5

W

WEBPACK_CONFIG_COMMON, 5