IBM Cúram Social Program Management
Version 7.0.3

IBM Social Program Management Design System 1.0.1

Note

Before using this information and the product it supports, read the information in “Notices” on page 25

Edition

This edition applies to IBM® Cúram Social Program Management v7.0.3 and to all subsequent releases unless otherwise indicated in new editions.

Contents

List of Tables

Chapter 1. IBM Social Program Management Design System

7.0.3.0

You can use the design system to develop your own custom web applications to complement the standard IBM Cúram Social Program Management web client. The design system provides the foundational packages for building accessible and responsive web applications. It consists of a React UI component library, React development resources, and a style guide for creating web applications.

The design system incorporates the US Web Design Standards and also supports additional CSS, utility classes, and a layout framework to enable teams to quickly build Section 508 compliant, responsive, and production-ready web applications.

Documentation versions

The online documentation applies only to the most recent version of the design system. To read the documentation in PDF format for earlier versions, see the IBM Cúram Social Program Management PDF library.

IBM Social Program Management Design System release notes

Read the design system release notes to see what is new or changed.

1.0.1 release notes

Read about updates and changes in IBM Social Program Management Design System version 1.0.1.

Govhhs-design-system

Change to package structure

A change to the package structure and package naming for the design system was required to improve its overall maintainability.

So in your `package.json`, instead of having the following package names:

```
"dependencies": {
  "@govhhs/govhhs-wds": "1.0.x",
  "@govhhs/govhhs-wds-react": "7.0.x",
  "react and other dependencies..."
 }
```

Change to the following package names:

```
"dependencies": {
  "@govhhs/govhhs-design-system-core": "^1.0.7",
  "@govhhs/govhhs-design-system-react": "^1.0.7",
  "react and other dependencies..."
},
```

Renaming imports

Because the repository name is now updated, your import paths must change also. Search and replace for the following in your favorite editor:

- WDS base CSS and JS files:

```
/* Previous paths */
```

```
import '@govhhs/govhhs-wds/dist/css/govhhs-wds.min.css';
import '@govhhs/govhhs-wds/dist/js/@govhhs/govhhs-wds.min';
```

Change to:

```
/* New paths */
import '@govhhs/govhhs-design-system-core/dist/css/govhhs-wds.min.css';
import '@govhhs/govhhs-design-system-core/dist/js/@govhhs/govhhs-wds.min';
```

- React components:

```
/* Previous paths */
import { Alert, Button, Form, TextInput } from '@govhhs/govhhs-wds-react';
```

change to:

```
/* New paths */
import { Alert, Button, Form, TextInput } from '@govhhs/govhhs-design-system-react';
```

- Saas files from WDS:

```
/* Previous paths */
@import "../../node_modules/@govhhs/govhhs-wds/src/stylesheets/all";
```

change to:

```
/* New paths */
@import "../../node_modules/@govhhs/govhhs-design-system-core/src/stylesheets/all";
```

govhhs-design-system-react release notes

v1.0.7

Addresses an issue found in 1.0.6 where the react documentation is not published as part of the package.

v1.0.6

- Guidance. Add new detail page template
- Components:
  - Update to Button component to allow Spinner as the Button icon
  - Fixed header mask demo
  - The *navClose* attribute now sets *data-nav-close* attribute on button instead of class

v1.0.5

- Guidance:
  - Update installation instructions
  - Added image in tooltip to example page
- Components:
  - Add new download button pattern
  - Add *actionElement* property to *IllustratedMessage*
  - Use the *RadioButton* value the on component update
  - SVG elements used as tooltip triggers no longer create pointer events
  - Improved tooltip arrow appearance, removed gap
  - Tooltip displaying correctly on small screens
  - Set correct background position for flipped link card arrow in rtl

v1.0.4

- Guidance:
  - Added Join utility documentation
  - Added text utilities documentation
  - Added Text truncate and pre-line utility docs
  - Added no-hyphens text utility and sass mixin
  - Fixed typo in text utilities docs
  - Removed primary styles from layered page header sample
- Components:
  - Avatar can be used with an icon
  - Add section heading component
  - Added text work break utility class
  - Added attachment icon
  - Added text work break utility class
  - Add *addonClassName* to *TextInput*
  - Update icon used in icon avatar
  - Updates to avatar documentation layout
  - Image will unsubscribe for events on *componentWillUnmount*
  - *FooterSection* wasn't passing *className* from props

v1.0.3

Bug fixes. Button icon color fix for buttons inside panels

v1.0.2

- Bug fixes:
  - Prevented directional icons styles being broken by post-css RTL plugin
  - Fixed Avatar zoom border behavior on chrome
- Guidance. Moved Media object to layout section
- Components. Changed media object to a type Layout (wds-l-) component

v.1.0.1

- Additions:
  - Allow *Header.title* to accept Objects
  - Adding region role on Panel Component
- Bug fixes:
  - Removing tabindex from Card
  - Textinput value can't be updated by passing new props
- Documentation. Updating accessibility documentation for Card

v.1.0.0

First release

spm-core

185 j_security_check and Rest/ calls not canceled by superagent.timeout

Previously, some REST API calls did not respect the timeout for that call to finish executing. An update has been made to ensure that all REST API calls respect the timeout for those calls to finish executing.

spm-intelligent-evidence-gathering

Read-only values filtered-out from submitted answers
Values were being submitted on summary pages, and more generally on read-only questions. They are now filtered out from the answers being sent back.

228 and 229 Page in a loop seems to be pre-populated
In pages that are contained within loops, the first iteration was displayed as expected, but subsequent ones were showing values from the previous page.

135 IEG: JavaScript warnings displayed on the console when the application form is displayed
Previously, JavaScript warnings were displayed on the console when the component *ApplicationFormQuestions* was displayed, now these warnings are fixed and are no longer displayed on the console.

147 IEG - allow header title to be displayed as text, not a link
Previously, the IEG header title was displayed as a link when it was not clickable, now it is displayed as text.

IEG: Check boxes when selected and then cleared by the user do not trigger the required validations
Previously, empty mandatory check boxes were not validated when clicking **Next** within an application form. Now, mandatory check box are validated correctly.

182 IEG: JavaScript warnings displayed on the console when the submission form is displayed
Previously, JavaScript warnings were displayed on the console when the component *SubmissionFormQuestions* was displayed, now these warnings are fixed and no longer displayed on the console.

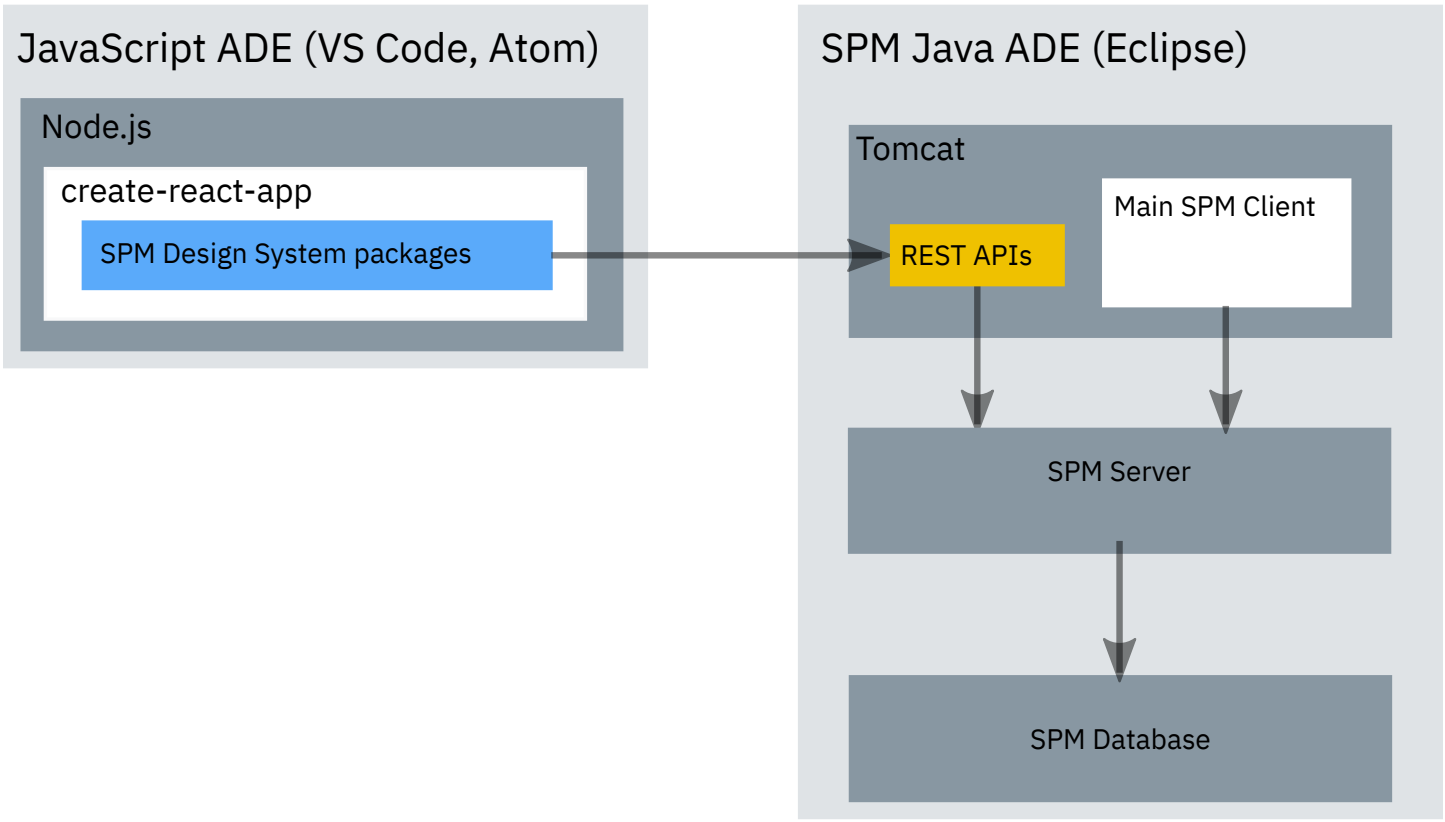
248 Header component - small form-side menu - remains in focus and is displayed when an item is selected from the side menu
Previously, the header component remained in focus and was displayed when the citizen logged in. Now, the header component does not display when an item is selected from the side menu.

Getting started with the design system

To get started quickly, install Node.js, install a React application, install the Design System Node packages, and complete the short tutorial. When it is time to develop and test your REST APIs, install the IBM Cúram Social Program Management Java Application Development Environment (ADE).

About this task

You need a React application into which you can install the Design System node packages. You can use any React application. However, the Facebook `create-react-app` contains some useful tools that simplify getting started with React development.



The Design System consists of the following Node packages:

- @spm/core
- @spm/intelligent-evidence-gathering
- @govhhs/govhhs-design-system-core
- @govhhs/govhhs-design-system-react

Procedure

1. Download and install Node.js version 8 or later from https://nodejs.org. The installation includes the npm (Node package manager), which you can use to install your Node packages.

   For more information about Node.js, see this Node.js developerWorks article.
2. Create your React application by using Facebook's `create-react-app`:
   a) From the directory where you want to create your React application, enter the following command, where *my-app* is the name that you want to call your application. A *my-app* directory is created to contain the application files.

   ```
   npx create-react-app my-app
   ```

   For more information about `create-react-app`, see the create-react-app user guide.
   b) You now have a basic React application in the *my-app* directory. You can run the application by entering the following commands:

   ```
   cd my-app
   npm start
   ```

   If the local host does not start automatically, browse to http://localhost:3000/ to see the running application.
3. Download, install, and configure the IBM Social Program Management Design System Node packages.

a) Open IBM Fix Central, select **Cúram Social Program Management**, your installed version and platform, and click **Continue**.

b) Ensure that **Browse for fixes** is selected, and click **Continue**.

c) Select the check box for `IBMSocialProgramManagementDesignSystem` and click **Continue**.

d) Only versions that are compatible with your IBM Cúram Social Program Management version are shown. Download SPM_DS_*<version>*`.zip` and extract the packages in the archive file to any directory.

e) From your *my-app* React application directory, enter the following command for each design system package to install the packages.

```
npm install <path>/<package-name>-<version>.tgz
```

Where *<path>* is the download path, *<package-name>* is a downloaded package, and *<version>* is the package version.

Note: Ignore any Node package dependency warnings for now. If needed, you can resolve them later.

f) To import the minified JavaScript and CSS files for the design system components, edit the *my-app* `\src\App.js` file and insert the following lines:

```
import '@govhhs/govhhs-design-system-core/dist/js/@govhhs/govhhs-wds.min';
import '@govhhs/govhhs-design-system-core/dist/css/govhhs-wds.min.css';
```

4. Now, you can do the short tutorial to get started, or you can install the IBM Cúram Social Program Management Java ADE that you need to develop and test your REST APIs, and do the tutorial later.

a) Complete the short tutorial to get started, see "Tutorial: Creating a page in your application" on page 7.

b) Install the Cúram SPM Java ADE, see Installing a Development Environment.

Upgrading to a new version of the design system

The design system is released on a frequent schedule. To upgrade to a new version, update the design system packages. Only the most recent version and two previous versions are supported. Ensure that you are reading the documentation for your version of the design system.

About this task

Search IBM Fix Central for your version of IBM Cúram Social Program Management to find and download the latest compatible version of the IBM Social Program Management Design System. Replace the Design System Node packages in your React application with the newer versions.

Procedure

1. Download and install the IBM Social Program Management Design System Node packages.

a) Select `IBMSocialProgramManagementDesignSystem` from IBM Fix Central, download SPM_DS_*<version>*`.zip`, and extract the packages in the archive file to any directory.

b) From your React application directory, enter the following command for each design system package to install the packages.

In the command, *<path>* is the download path, *<package-name>* is a downloaded package, and *<version>* is the package version.

Note: Ignore any Node package dependency warnings for now. If needed, you can resolve them later.

```
npm install <path>/<package-name>-<version>.tgz
```

2. Complete any upgrade steps in the "IBM Social Program Management Design System release notes" on page 1.

Developing your web application

Create your web application with the help of the development resources that are included in the IBM Social Program Management Design System.

Tutorial: Creating a page in your application

In this short tutorial, you update the create-react-app App.js file in your React application to build an application page with the Design System components. For more information about how to design and use Design System components, see the Storybook documentation in `@govhhs/govhhs-design-system-react/doc/index.html`.

Procedure

1. Let's run the application. While it is running, you can see your changes as you make them. Run the application by entering the following commands:

```
cd my-app
npm start
```

   Browse to http://localhost:3000/ to see the running application.

2. Edit *my-app*`/src/App.js`.

First, let's add a header, which includes a nav bar, logo, and the agency name. For the header to load, we must import four components.

- Header
- Primary Navigation
- NavigationLink
- Link

3. We need to import the image for the header logo, so add the following line after the JavaScript and CSS imports lines we added during installation.

```
import logo from '@govhhs/govhhs-design-system-core/dist/img/logo-mark.svg';
```

   If a duplicate `import logo` statement is already included in the App.js file, delete it.

4. To make the components available for us to use, insert the following component import statement after the logo import statement.

```
import {
  Header,
  PrimaryNavigation,
  SecondaryNavigation,
  NavigationLink,
  Link,
  HeaderOverflowMenu,
  OverflowMenuOption
} from "@govhhs/govhhs-design-system-react"
```

5. Replace the code inside the create-react-app render function with the following code.

```
<div>
        <Header
          title="Agency name"
          logo={<img src={logo} alt="logo"/>}
        >
          <PrimaryNavigation>
            <NavigationLink
              title="Section One"
              id="basic-nav-section-one"
            >
              <Link>
              Sub-Link One
              </Link>
              <Link>
              Sub-Link Two
              </Link>
              <Link>
              Sub-Link Three
              </Link>
            </NavigationLink>
            <NavigationLink
              title="Section Two"
              id="basic-nav-section-two"
            >
              <Link>
              Sub-Link One
              </Link>
              <Link>
              Sub-Link Two
              </Link>
              <Link>
              Sub-Link Three
              </Link>
            </NavigationLink>
            <NavigationLink
              title="Link Three"
              id="basic-nav-section-three"
            />
            <NavigationLink
              title="Link Four"
              id="basic-nav-section-four"
            />
          </PrimaryNavigation>
        </Header>
       </div>
```

Add the Hero component.

6. In the component import statement, insert Hero into the list of imported components.

```
import {
  Hero,
  Header,
  PrimaryNavigation,
  SecondaryNavigation,
  SecondaryNavigationList,
  NavigationLink,
  Link,
  HeaderOverflowMenu,
  OverflowMenuOption
} from "@govhhs/govhhs-design-system-react"
```

7. When you add more than one component in the `return()` function, the JSX code must be wrapped in a surrounding element. We want the Hero component to load underneath the header, so insert the <Hero /> tag after the </Header> tag in the `return()` function. The `return()` function now has the following structure:

```
<div>
        <Header>
        ...
            <NavigationLink
              title="Link Four"
              id="basic-nav-section-four"
            />
          </PrimaryNavigation>
        </Header>
```

```
        <Hero />
      </div>
```

The hero component is displayed in the running application.

Now let's add some cards to our page. Cards are a great way to provide an entry point to more details, or access to actions on card content.

We must add some layout components to our list of imports:

- Grid
- Column
- Section

And add the card components to our list of imports:

- Card
- CardHeader
- CardBody
- CardFooter

8. Update your component import statement by inserting the following card components:

```
import {
  .....
  Card,
  CardHeader,
  CardBody,
  CardFooter,
  Grid,
  Column,
  Section,
} from "@govhhs/govhhs-design-system-react"
```

9. To display the cards underneath the </Hero> component, append the following JSX code to lay three link cards out in a grid:

```
 <Section>
          <Grid>
          <h2>Card Actions</h2>
          <p>Before using cards in your project, have a look at the guidance in Storybook
on how to use cards correctly.</p>
            <Column width="1/3">
              <Card href="/404" className="wds-u-mr--small wds-u-mt--small">
                <CardHeader title="Card header" highlight/>
                  <CardBody>
                    <p>
                      Neque porro quisquam est qui dolorem ipsum quia dolor sit amet,
consectetur, adipisci velit.
                    </p>
                  </CardBody>
                <CardFooter>
                </CardFooter>
              </Card>
              </Column>
              <Column width="1/3">
              <Card href="" className="wds-u-mt--small">
                <CardHeader title="Card header" highlight/>
                  <CardBody>
                    <p>
                      Neque porro quisquam est qui dolorem ipsum quia dolor sit amet,
consectetur, adipisci velit.
                    </p>
                  </CardBody>
                <CardFooter>
                </CardFooter>
              </Card>
              </Column>
              <Column width="1/3">
              <Card href="" className="wds-u-mt--small">
                <CardHeader title="Card header" highlight/>
                  <CardBody>
                    <p>
                      Neque porro quisquam est qui dolorem ipsum quia dolor sit amet,
consectetur, adipisci velit.
```

```
                  </p>
                </CardBody>
              <CardFooter>
              </CardFooter>
            </Card>
          </Column>
        </Grid>
      </Section>
```

The cards are displayed in the running application.

Next, add a section that contains a form, an image, and a content area. For this section, we need to add some more components to our component import statement. Hopefully you're getting the hang of this!

- FieldSet
- TextInput
- Content
- TextArea
- Button
- Image

10. Add the following items to your component import statement:

```
import {
  ....
  FieldSet,
  TextInput,
  Content,
  TextArea,
  Button,
  Image,
  Form
} from '@govhhs/govhhs-design-system-react';
```

11. Because we are using an image in this part of the page, we need to import that too. Add the following line before our component import statement:

```
import img from '@govhhs/govhhs-design-system-core/dist/img/hero-480.jpg';
```

12. For the last step, to append the JSX code to our `return()` function, add the following code after the last `</Section>` tag:

```
<Section className="wds-c-section--light">
          <Grid>
            <Column width="1/2">
            <Content>
              <h3 className="wds-u-mt--small">About</h3>
              <p>All design system components are fully responsive and will work on mobile,
tablet and desktop browsers.
              Try it out by resizing the browser window.</p>
              <Image
                src={img}
                alt="Error loading Image"
                fallback={<span />}
              />
            </Content>
            </Column>
            <Column width="1/2">
              <h3 className="wds-u-mt--small">Form title</h3>
                <form>
                  <FieldSet>
                    <TextInput
                      label="Text input"
                      value="Input text"
                    />
                    <TextInput
                      label="Text input"
                      value="Input text"
                    />
                    <TextArea label="Text area label"></TextArea>
                    <Button category="primary" style={{float: "right"}}>Sign up</Button>
                  </FieldSet>
                </form>
            </Column>
```

```
          </Grid>
        </Section>
```

Results

That's it, your page is now complete, you have created a fully responsive page in very little time! Hopefully you enjoyed this tutorial and see the benefits of using our design system.

Example

Here's the final App.js file for reference:

```
import React, { Component } from 'react';
import './App.css';
import '@govhhs/govhhs-design-system-core/dist/js/@govhhs/govhhs-wds.min';
import '@govhhs/govhhs-design-system-core/dist/css/govhhs-wds.min.css';
import logo from '@govhhs/govhhs-design-system-core/dist/img/logo-mark.svg';
import img from '@govhhs/govhhs-design-system-core/dist/img/hero-480.jpg';
import {
  Hero,
  Header,
  PrimaryNavigation,
  SecondaryNavigation,
  SecondaryNavigationList,
  NavigationLink,
  Link,
  HeaderOverflowMenu,
  OverflowMenuOption,
  Card,
  CardHeader,
  CardBody,
  CardFooter,
  Grid,
  Column,
  Section,
  FieldSet,
  TextInput,
  Content,
  TextArea,
  Button,
  Image,
  Form
} from "@govhhs/govhhs-design-system-react";

class App extends Component {
  render() {
    return (
      <div>
        <Header
          title="Agency name"
          logo={<img src={logo} alt="logo"/>}
        >
          <PrimaryNavigation>
            <NavigationLink
              title="Section One"
              id="basic-nav-section-one"
            >
              <Link>
              Sub-Link One
              </Link>
              <Link>
              Sub-Link Two
              </Link>
              <Link>
              Sub-Link Three
              </Link>
            </NavigationLink>
            <NavigationLink
              title="Section Two"
              id="basic-nav-section-two"
            >
              <Link>
              Sub-Link One
              </Link>
              <Link>
              Sub-Link Two
              </Link>
              <Link>
```

```
              Sub-Link Three
              </Link>
            </NavigationLink>
            <NavigationLink
              title="Link Three"
              id="basic-nav-section-three"
            />
            <NavigationLink
              title="Link Four"
              id="basic-nav-section-four"
            />
          </PrimaryNavigation>
        </Header>
        <Hero />
        <Section>
         <Grid>
         <h2>Card Actions</h2>
         <p>Before using cards in your project, have a look at the guidance in Storybook on how
to use cards correctly.</p>
           <Column width="1/3">
             <Card href="/404" className="wds-u-mr--small wds-u-mt--small">
               <CardHeader title="Card header" highlight/>
                 <CardBody>
                   <p>
                     Neque porro quisquam est qui dolorem ipsum quia dolor sit amet,
consectetur, adipisci velit.
                   </p>
                 </CardBody>
               <CardFooter>
               </CardFooter>
             </Card>
             </Column>
             <Column width="1/3">
             <Card href="" className="wds-u-mt--small">
               <CardHeader title="Card header" highlight/>
                 <CardBody>
                   <p>
                     Neque porro quisquam est qui dolorem ipsum quia dolor sit amet,
consectetur, adipisci velit.
                   </p>
                 </CardBody>
               <CardFooter>
               </CardFooter>
             </Card>
             </Column>
             <Column width="1/3">
             <Card href="" className="wds-u-mt--small">
               <CardHeader title="Card header" highlight/>
                 <CardBody>
                   <p>
                     Neque porro quisquam est qui dolorem ipsum quia dolor sit amet,
consectetur, adipisci velit.
                   </p>
                 </CardBody>
               <CardFooter>
               </CardFooter>
             </Card>
           </Column>
         </Grid>
       </Section>
       <Section className="wds-c-section--light">
          <Grid>
            <Column width="1/2">
            <Content>
              <h3 className="wds-u-mt--small">About</h3>
              <p>All design system components are fully responsive and will work on mobile,
tablet and desktop browsers.
              Try it out by resizing the browser window.</p>
              <Image
                src={img}
                alt="Error loading Image"
                fallback={<span />}
              />
            </Content>
            </Column>
            <Column width="1/2">
              <h3 className="wds-u-mt--small">Form title</h3>
                <form>
                  <FieldSet>
                    <TextInput
                      label="Text input"
                      value="Input text"
```

```
                    />
                    <TextInput
                      label="Text input"
                      value="Input text"
                    />
                    <TextArea label="Text area label"></TextArea>
                    <Button category="primary" style={{float: "right"}}>Sign up</Button>
                  </FieldSet>
                </form>
              </Column>
            </Grid>
          </Section>
        </div>
    );
  }
}

export default App;
```

JavaScript development environment

You can use any JavaScript development environment to develop your application, for example, Microsoft Visual Studio Code, Atom, or Sublime. Choose the tools that suits you best.

The IBM Social Program Management Design System does not depend on any specific tools, so you can choose your own environment. However, Microsoft Visual Studio Code supports many plug ins that make development faster and easier, for example:

- Linting tools (ESLint)
- Code formatters (Prettier)
- Debugging tools (Debugger for Chrome)
- Documentation tools (JSDoc)

IBM does not own, develop, or support any of the tools.

Design system packages

Use the design system packages to help you to develop and test your web client.

`govhhs-design-system-core` package

The *govhhs-design-system-core* package contains a style guide, a library of user interface components, and front-end development resources that you can use to create Section 508-compliant, responsive, consistent web applications. The design system provides CSS, utility classes, and a grid framework so that you can quickly build accessible, responsive, production-ready websites.

`govhhs-design-system-react` package

The *govhhs-design-system-react* package contains a React component library in Storybook to help you to build your application. It provides a collection of React components that align with the IBM Social Program Management Design System.

For more information about the design guidelines, utility classes, and React components, see `govhhs/govhhs-design-system-react/doc`.

`core` package

The *core* package provides JavaScript utilities to help you develop your application. For example, use the *RESTService* utility to connect to a IBM Cúram Social Program Management server-side REST API. Use *IntlUtils* to format numbers and dates for globalization.

For more information about the core package utilities, see the JSDoc API documentation in `spm/core/doc`.

`intelligent-evidence-gathering` package

The *intelligent-evidence-gathering* package enables IEG scripts that are configured in the IBM SPM application to run in your application. An API is provided to call the IEG scripts.

For more information, see the API documentation in `spm/intelligent-evidence-gathering/doc`.

Note: The *intelligent-evidence-gathering* package is not currently supported without IBM Cúram Universal Access.

Connecting to REST APIs

You can connect your web application to REST APIs, such as the IBM Cúram Social Program Management REST APIs.

For more information about IBM Cúram Social Program Management REST APIs, see *Developing Cúram REST APIs*.

Related information

Developing Cúram REST APIs

The RESTService utility

The `@spm/core` package provide the RESTService utility, which you can use to connect your application to a REST API. You can fetch resources with alternatives such as Fetch API, SuperAgent,or Axios. However, the RESTService utility provides some useful functions for connecting to SPM REST APIs.

The RESTService utility supports the GET, POST, and DELETE HTTP methods through the following JavaScript methods:

- `RESTService.get(url, callback, params)`
- `RESTService.post(url, data, callback)`
- `RESTService.del(url, callback)`

The full RESTService class documentation is in the `doc` folder in the `@spm/core` package.

The RESTService utility hides details of calls, such as passing credentials, language, and errors. The callback that is passed to the GET, POST, or DELETE methods is started after the API calls return. API calls are asynchronous, so write your code to expect and handle a delay in receiving a response.

The RESTService utility provides the following functions during communications.

Authentication

Authentication of the user is handled transparently by the RESTService utility. After a user is authenticated, the REST APIs automatically send the required 'credentials', that is, the authentication cookies, with each request. For more on how authentication is handled for REST, see Cúram REST API security.

If a user's session is invalidated before a new request is made to a REST API, then the '401 unauthorised' response is returned by the server. The the RESTService utility relays the response to the callback function passed by the caller.

Handling responses

The RESTService utility formats the response from the server to ensure that callbacks receive the response in a consistent manner.

Each get, post, and delete method accepts a callback function from the caller. When invoked by the RESTService utility, the callback function receives a boolean that indicates the success or failure of the API call and the response. The callback function can then deal with the result. For example, a failure can be used to trigger your code to throw an error with the response data that can be used to trigger an error

boundary. For more information on the callback function parameters, see the API documentation for the RESTService utility.

User Language

The 'Accept-Language' HTTP header is automatically set by the RESTService utilitye based on the user's selected language, which the user can select using the language picker in the reference application. This allows the server to respond in the correct locale where locale sensitive information is being handled on the server.

The locale passed in the header is set in the transaction that is initiated by that REST request, and is used for the duration of that transaction. For more on transactions, see Transaction control.

Handling Timeouts

The RESTService utility can manage unresponsive calls to the server. The following properties are set, and can be modified, in the `.env` files to set thresholds for timeouts.

- `REACT_APP_RESPONSE_TIMEOUT=10000 // Wait 10 seconds for the server to start sending`
- `REACT_APP_RESPONSE_DEADLINE=60000 // but allow 1 minute for the file to finish loading`

Simulating slow responses

During development, it is important to test that your application continues to operate in an acceptable way even when network responses are slow. You can simulate a slow network connection by setting a property in the `.env.development` file in the root of your project.

For example, setting `REACT_APP_DELAY_REST_API=2500` delays the response from all GET requests for 2.5 seconds.

The value can be set to any positive integer to adjust the delay.

Table 1 outlines the process environment variables that the API uses. Use the variables to configure the service.

Table 1: Process environment variables

Variable	Setting	Default	Description
REACT_APP_RESPONSE_TIMEOUT	Milliseconds	10 seconds	Sets the maximum time to wait for the first byte to arrive from the server, but does not limit how long the entire download can take. Set the response timeout to be a few seconds longer than the actual time it takes the server to respond. The lengthened response allows for time to make DNS lookups, TCP/IP, and TLS connections.
REACT_APP_RESPONSE_DEADLINE	Milliseconds	60 seconds	Sets a deadline for the entire request, including all redirects, to complete. If the response is not fully downloaded within REACT_APP_RESPONSE_DEADLINE, the request is aborted.
REACT_APP_DELAY_REST_API	Milliseconds		Use only for development testing to simulate a delay in the response from the API.

Connecting to REST APIs on Tomcat

If you have deployed REST APIs to Tomcat in your Eclipse development environment, you can connect to them from your application.

Before you begin

For more information about building and deploying IBM Cúram Social Program Management REST APIs, see *Building a Cúram REST API* and *Deploying Cúram REST APIs on Tomcat*.

About this task

Use the API for Tomcat in the URL for your call to the RESTService utility. See the following example RESTService call.

```
RESTService.get('http://localhost:9080/Rest/v1/myAPI', (success, response) => {
  if (success) {
    // deal with response containing the json body
  } else {
    // deal with error contained in response
  }
});
```

Related information

Building a Cúram REST API

Deploying Cúram REST APIs on Tomcat

Authenticating against REST APIs

You can authenticate against REST APIs by using the RESTService utility or by using the IBM Cúram Social Program Management REST API security feature.

Authenticating against REST APIs by using the RESTService utility

Start the authentication URL *j_security_check* by using the POST function on the RESTService utility.

The following example shows a sample log-in invocation:

```
const callbackAfterLogin = (success, response) => {
  if (success) {
    // Login succeeded
  } else {
    // Login failed
  }
};
const loginUrl = 'http://localhost:9080/Rest/j_security_check';
const loginData = {
  j_username: username,
  j_password: password,
  user_type: 'EXTERNAL',
};
RESTService.post(loginUrl, loginData, callbackAfterLogin, 'form');
```

Authenticating against REST APIs by using the IBM Cúram Social Program Management REST API security feature

For more information, see Cúram REST API security.

Authenticating against REST APIs in Tomcat

A JAAS authentication mechanism is not exposed by the Tomcat and Eclipse environment, so you cannot authenticate your web application through the RESTService utility. To authenticate against a REST API in a Tomcat and Eclipse environment, use the Eclipse RMILoginClient class.

For more information about authenticating against REST APIs in a Tomcat and Eclipse environment by using the Eclipse RMILoginClient class, see *Starting the clients*.

When you authenticate, log in by using your external user name and password to authenticate against the server. Subsequent calls to your REST API simulate this user.

Related information

Starting the clients

Deploying your web application to a web server

You can deploy your web application on a web server in a production-like environment as part of your development process. Deployment in a production environment is outside the scope of this documentation, but you can use the instructions in this section for guidance.

Prerequisites and supported software

Deploying a responsive web application requires a web server in the IBM Cúram Social Program Management topology.

The following application server, web server, and DBMS combinations are supported.

- IBM WebSphere® Application Server, IBM HTTP Server, and IBM DB2®
- IBM WebSphere Application Server, IBM HTTP Server, and Oracle Database
- Oracle WebLogic Server, Oracle HTTP Server, and Oracle Database

For more information about installing an application server for IBM Cúram Social Program Management, see *Installing an enterprise application server*.

HTTP servers

Choose a web server version that is compatible with your application server.

Table 2: HTTP servers

Supported software	Version	Prerequisite minimum version	Universal Access minimum release	Operating system restrictions
IBM HTTP Server	9.0	9.0.0.5	1.0.0	No
	8.5.5	8.5.5.9	1.0.0	No
Oracle HTTP Server	(12.1.3)	(12.1.3)	1.0.0	No

Web browsers

IBM Cúram Universal Access is developed for public-facing applications. Every effort has been made to ensure that the pages that are specified for the application use standard web technologies and formats, which should be compatible with all browsers that are listed. However, the browsers that are listed in the following table are the only browsers that are officially supported.

New versions of Chrome, Firefox, Edge, and Safari are released more frequently than Internet Explorer, and updates are installed automatically by default for these browsers. Universal Access releases are tested on the latest versions of the browsers that are available at the start of IBM's development cycle;

Note: Only stable Chrome releases are tested.

If a browser is tested and no issues are found, IBM certifies that version. The prerequisites advise the version that is certified at each new product release. If IBM cannot certify a version, you might need to

revert to a previous, fully certified version. While IBM supports customers who are on newer versions of the browsers than the last certified version, customers must know that those versions of the browsers are not fully tested.

The following browsers are supported:

Table 3: Web browsers

Supported software	Version	Prerequisite minimum version	Universal Access minimum release	Operating system restrictions
Apple Safari	11 and future fix packs	11	1.0.0	No
Google Chrome	64 and future fix packs	64	1.0.0	No
Microsoft Edge	41 and future fix packs	41	1.0.0	No
Microsoft Internet Explorer	11 and future fix packs	11	1.0.0	No
Mozilla Firefox	58 and future fix packs	58	1.0.0	No

Related information

Installing an enterprise application server

Install and configure IBM HTTP Server with WebSphere Application Server

Install and configure IBM HTTP Server either on the same server as WebSphere Application Server or on a remote server. To enable cross-origin resource sharing (CORS), you can set the curam.rest.allowedOrigins property for the REST application on your application server, or install the IBM HTTP Server plug-in for WebSphere Application Server.

Before you begin

WebSphere Application Server must be installed and configured.

Install IBM Installation Manager. For more information, see the IBM Installation Manager documentation. You can download IBM Installation Manager from Installation Manager and Packaging Utility download documents.

About this task

To enable cross-origin resource sharing (CORS), choose one of the following options:

- Set the curam.rest.allowedOrigins property for the REST application that is deployed on the application server. For more information about setting the curam.rest.allowedOrigins property, see Cúram REST configuration properties.
- Install and configure the IBM HTTP Server plug-in for WebSphere Application Server to enable IBM HTTP Server to communicate with WebSphere Application Server. WebSphere Customization Toolbox is needed to configure the plug-in.

Procedure

1. Install IBM HTTP Server. For more information, see Migrating and installing IBM HTTP Server.
2. Optional: If you don't set the curam.rest.allowedOrigins property, you must install the following software:

a) Install the IBM HTTP Server plug-in for WebSphere Application Server.
   For more information, see Installing and configuring web server plug-ins.
b) Install the WebSphere Customization Toolbox.
   For more information, see Installing and using the WebSphere Customization Toolbox.

3. Start IBM HTTP Server. For more information, see Starting and stopping the IBM HTTP Server administration server.
4. To secure IBM HTTP Server, see Securing IBM HTTP Server.

Generating an IBM HTTP Server plug-in configuration
This task is needed only if you install the IBM HTTP Server plug-in for WebSphere Application Server. Use WebSphere Customization Toolbox to generate a plug-in configuration.

Before you begin
Start WebSphere Application Server. For more information, see Starting a WebSphere Application Server traditional server.

Procedure

To generate the IBM HTTP Server plug-in configuration, complete the steps at the WebSphere Application Server Network Deployment plug-ins configuration topic.

Configuring the IBM HTTP Server plug-in
Configure the IBM HTTP Server plug-in for WebSphere Application Server and WebSphere Customization Toolbox. This task is necessary only if you have chosen to install the IBM HTTP Server plug-in, instead of setting the curam.rest.allowedOrigins property for the REST application that is deployed on the application server.

About this task
You can run the `configurewebserverplugin` target to complete the following tasks:

- Add the web server virtual hosts to the client hosts configuration in WebSphere Application Server.
- Propagate the plug-in key ring for the web server.
- Map the modules of any deployed applications to the web server.

Procedure

1. Start IBM HTTP Server.
   For more information, see Starting and stopping the IBM HTTP Server administration server.
2. On the remote WebSphere Application Server, run the following command.

   ```
   build configurewebserverplugin -Dserver.name=server_name
   ```

   The `configurewebserverplugin` target requires a mandatory `server.name` argument that specifies the name of the server when the target is invoked. For more information about the `configurewebserverplugin` target, see Configuring a web server plug-in in WebSphere Application Server.
3. Consider adding extra aliases to the *client_host*, as shown in the following examples:
   - For WebSphere Application Server, add port number 9044.
   - For the default HTTP port, add port number 80.
   - For HTTPS ports, add port number 433.

   For more information about client host setup, see step 19 in the WebSphere Application Server port access setup topic.
4. To avoid port mapping issues from web applications, restart WebSphere Application Server and IBM HTTP Server.

For more information, see Starting and stopping the IBM HTTP Server administration server.

Related information

Set up the Port Access
Start a WebSphere Server
Restart a WebLogic Server

Install and configure Oracle HTTP Server with Oracle WebLogic Server

Install and configure Oracle HTTP Server on either the same server as Oracle WebLogic Server or on a remote server.

Before you begin

Oracle WebLogic Server must be installed and configured. For more information, see Installing and Configuring Oracle WebLogic Server and Coherence.

Installing Oracle HTTP Server and its components

Install and configure Oracle HTTP Server in either a stand-alone domain, or in an Oracle WebLogic Server domain. If Oracle HTTP Server and Oracle WebLogic Server are on different computers, you must install and configure an Oracle web server plug-in for proxying requests.

About this task

The Oracle web server plugin allows requests to be proxied from Oracle HTTP Server to Oracle WebLogic Server. If you install and configure the Oracle web server plug-in, requests that are delegated to Oracle WebLogic Server still appear to originate from the Oracle HTTP Server, even if Oracle HTTP Server and Oracle WebLogic Server are hosted on two different servers.

Because of the web browser same-origin policy, cross-origin resource sharing (CORS) is restricted in many browsers by default. The web server plug-into enables CORS where Oracle HTTP Server and Oracle WebLogic Server are installed on different computers.

CORS enables an instance of your web application that is deployed on Oracle HTTP Server in one domain to request the REST services that are deployed on Oracle WebLogic Server in another domain.

Procedure

1. Install Oracle HTTP Server for Oracle WebLogic Server. For more information, see Installing and Configuring Oracle HTTP Server.
2. To configure Oracle HTTP Server, choose one of the following options:
   - To configure Oracle HTTP Server in a stand-alone domain, follow the instructions at Configuring Oracle HTTP Server in a Standalone Domain.
   - To configure Oracle HTTP Server in an Oracle WebLogic Server domain, follow the instructions at Configuring Oracle HTTP Server in a WebLogic Server Domain.
3. If Oracle HTTP Server and Oracle WebLogic Server are installed in different domains, to enable CORS, install a web server plug-in.

   For information about configuring an Oracle WebLogic Server proxy plug-in, see Configuring the Plug-In for Oracle HTTP Server.
4. To secure Oracle HTTP Server, follow the procedure at Managing Application Security.

Results

The Oracle HTTP Server instance is now ready to for you to deploy the application. The default location for deploying the application is *OHS_INSTANCE*`/config/fmwconfig/components/${COMPONENT_TYPE}/instances/${COMPONENT_NAME}/htdocs`. However, you can configure the default location value to a different location.

What to do next

Start Oracle HTTP Server. For more information, see Starting the Servers.

Configuring the Oracle HTTP Server plug-in

If a web server such as Oracle HTTP Server is configured in the topology, you must configure a web server plug-in in Oracle WebLogic Server. The web server plug-in enables Oracle WebLogic Server to communicate with Oracle HTTP Server.

About this task

To enable an Oracle HTTP Server web server plug-in in Oracle WebLogic Server, you can run the `configurewebserverplugin` target.

Procedure

1. Start Oracle HTTP Server.

   For more information, see Starting the Servers.
2. On the remote Oracle WebLogic Server, run the following command.

   The `configurewebserverplugin` target requires a mandatory `server.name` argument that specifies the name of the server when the target is invoked.

   ```
   build configurewebserverplugin -Dserver.name=server_name
   ```

   For more information about the `configurewebserverplugin` target, see Configuring a web server plug-in in Oracle WebLogic Server.

Building your web application for deployment

You must build your web application for deployment on a web server. A `build` directory is created that contains all of the required files for your web application.

About this task

For more information about npm build and deployment, see npm run build in the `create-react-app` GitHub documentation.

Procedure

From your application root directory, run the following command to create the `build` directory.

```
npm run build
```

Deploying your web application

To test your web application against an existing IBM Cúram Social Program Management application that is deployed on an enterprise application server, you can deploy the web application on IBM HTTP Server or Oracle HTTP Server. Both web servers are based on Apache HTTP server so the deployment procedure is similar.

Before you begin

You must have built your application for deployment.

About this task

The built deliverable comes with a preconfigured `.htaccess` configuration file for the Content-Security-Policy (CSP) header. When you configure the CSP header in the web server, the `.htaccess` file is detected and executed by the web server to alter the web server configuration by enabling or disabling additional functionality. For more information about CSP, see the *Content Security Policy Quick Reference Guide* related link.

Procedure

1. Copy and paste the `build` directory contents to the appropriate directory for your HTTP server.

   For more information about the `<directory>` directive, see the related links.
2. Configure the web server to call the `.htaccess` file.

   For more information about how to configure `.htaccess` files in a web server, see the *Apache HTTP Server Tutorial: .htaccess files* related link.

Related information

GitHub documentation: npm run build
Content Security Policy Quick Reference Guide
Apache core features V2.0: <Directory> Directive
Apache core features V2.4: <Directory> Directive
Apache HTTP Server Tutorial: .htaccess files

Troubleshooting and support

Use this information to help you to troubleshoot issues with the IBM Cúram Universal Access Responsive Web Application or IBM Social Program Management Design System.

The IBM Cúram Social Program Management supported assets can be installed, customized, and deployed separately from IBM Cúram Social Program Management, before being integrated into the system.

When troubleshooting web applications that are integrated with IBM Cúram Social Program Management, use this troubleshooting information in conjunction with the troubleshooting information for IBM Cúram Social Program Management. For more information, see the *Troubleshooting and support* related link.

Related information

Troubleshooting and support

Citizen Engagement components and licensing

You can use and customize the new Universal Access web application for your organization, or develop your own custom web applications to complement the standard IBM Cúram Social Program Management web client. Use this information to understand the IBM Cúram Social Program Management components, supported assets, and licenses that you need.

Installable components

IBM Social Program Management Design System supported asset
The design system provides the foundational packages for building accessible and responsive web applications. It consists of a React UI component library, React development resources, and a style guide for creating web applications.

Universal Access Responsive Web Application supported asset
The new Universal Access web application, which you can use and customize for your organization. The responsive web application requires the IBM Social Program Management Design System and the Universal Access application module.

Universal Access application module
The Universal Access (UA) application module includes REST APIs that expose interfaces to Universal Access and IEG functions for consumption by the Universal Access Responsive Web Application. Universal Access requires the IBM Cúram Social Program Management Platform.

Licensing Universal Access (New)

To use and customize the new Universal Access web application, customers can buy the Universal Access application module, which entitles the Universal Access Responsive Web Application asset, and IBM Cúram Social Program Management Platform, which entitles the IBM Social Program Management Design System asset. Customers can also buy Citizen Engagement, which includes the Universal Access application module, the IBM Cúram Social Program Management Platform, and the assets.

Licensing the IBM Social Program Management Design System

To develop custom web applications to complement the standard IBM Cúram Social Program Management web client, customers must buy the IBM Cúram Social Program Management Platform, which entitles the IBM Social Program Management Design System asset.

Citizen Engagement support strategy

The Citizen Engagement assets are expected to be released monthly, and they can be upgraded independently of the base IBM Cúram Social Program Management product. Due to the more frequent release schedule, the asset support strategy is to maintain a single product line for the supported assets for both new features and maintenance. Where possible, all updates are planned for the latest version of the assets. Security and defect fixes will be delivered in the latest release only.

Support strategy for the Universal Access application module

Where possible, Universal Access REST APIs changes are delivered in refresh pack or other impact-free releases that impose no forced upgrade impact.

Support strategy for the supported assets

Assets versions are supported for the lifetime of the latest supported IBM Cúram Social Program Management version available at the time of the asset release.

The assets use semantic versioning. As a general guideline, this means:

- MAJOR version for incompatible API changes
- MINOR version for adding functionality in a backwards-compatible manner
- PATCH version for backwards-compatible bug fixes

The assets will be full releases rather than delta releases regardless of version type.

Although new features (pages) can be delivered in any minor release, new pages will generally be delivered at the same time as the Universal Access release that contains the new APIs for those features.

Compatibility

You can confirm compatibility between a version of the supported assets and the IBM Cúram Social Program Management software by searching for your software version in Fix Central. Only compatible versions of the asset are available. In addition, the asset release notes will contain information on compatibility. For example, "This version is compatible with versions 7.0.3 and 7.0.4."

Examining log files

Log files are a useful resource for troubleshooting problems.

Examining the browser console logs

For JavaScript applications, you can examine the browser console logs for errors that might be relevant to investigating problems. For the exact details about how to locate the console logs within the browser, see your browser documentation.

Note: When you are developing applications with the IBM Social Program Management Design System, console logging information might also be displayed within the console that runs the start process for the application.

Examining the HTTP Server log files

When you deploy a built application on an HTTP Server, the built application introduces a new point with which logging is captured in your system topology. The IBM HTTP Server and the Oracle HTTP Server include comprehensive logging system and related information.

For more information about troubleshooting the IBM HTTP Server, see Troubleshooting IBM HTTP Server.

For more information about troubleshooting the Oracle HTTP Server, see Managing Oracle HTTP Server Logs.

Known limitations

Review the known limitations for the IBM Social Program Management Design System and IBM Cúram Universal Access, and, where available, workaround information.

Note: IBM Social Program Management Design System and IBM Cúram Universal Access are provided only in US English.

Symptom

Translation packs are not provided with IBM Social Program Management Design System and IBM Cúram Universal Access. Therefore, translations are not currently provided for the languages that are supported by the corresponding IBM Cúram Social Program Management or IBM Universal Access products.

Resolution

Support for regionalizing or localizing your user interface (UI) is not affected.

Notices

This information was developed for products and services offered in the United States.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not grant you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing IBM Corporation North Castle Drive, MD-NC119 Armonk, NY 10504-1785 US

For license inquiries regarding double-byte character set (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

Intellectual Property Licensing Legal and Intellectual Property Law IBM Japan Ltd. 19-21, Nihonbashi-Hakozakicho, Chuo-ku Tokyo 103-8510, Japan

INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some jurisdictions do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM websites are provided for convenience only and do not in any manner serve as an endorsement of those websites. The materials at those websites are not part of the materials for this IBM product and use of those websites is at your own risk.

IBM may use or distribute any of the information you provide in any way it believes appropriate without incurring any obligation to you.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

IBM Director of Licensing IBM Corporation North Castle Drive, MD-NC119 Armonk, NY 10504-1785 US

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The licensed program described in this document and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement or any equivalent agreement between us.

The performance data and client examples cited are presented for illustrative purposes only. Actual performance results may vary depending on specific configurations and operating conditions.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBMproducts. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

Statements regarding IBM's future direction or intent are subject to change or withdrawal without notice, and represent goals and objectives only.

All IBM prices shown are IBM's suggested retail prices, are current and are subject to change without notice. Dealer prices may vary.

This information is for planning purposes only. The information herein is subject to change before the products described become available.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to actual people or business enterprises is entirely coincidental.

COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs. The sample programs are provided "AS IS", without warranty of any kind. IBM shall not be liable for any damages arising out of your use of the sample programs.

Privacy Policy considerations

IBM Software products, including software as a service solutions, (“Software Offerings”) may use cookies or other technologies to collect product usage information, to help improve the end user experience, to tailor interactions with the end user or for other purposes. In many cases no personally identifiable information is collected by the Software Offerings. Some of our Software Offerings can help enable you to collect personally identifiable information. If this Software Offering uses cookies to collect personally identifiable information, specific information about this offering’s use of cookies is set forth below.

Depending upon the configurations deployed, this Software Offering may use session cookies or other similar technologies that collect each user’s name, user name, password, and/or other personally identifiable information for purposes of session management, authentication, enhanced user usability, single sign-on configuration and/or other usage tracking and/or functional purposes. These cookies or other similar technologies cannot be disabled.

If the configurations deployed for this Software Offering provide you as customer the ability to collect personally identifiable information from end users via cookies and other technologies, you should seek your own legal advice about any laws applicable to such data collection, including any requirements for notice and consent.

For more information about the use of various technologies, including cookies, for these purposes, see IBM’s Privacy Policy at http://www.ibm.com/privacy and IBM’s Online Privacy Statement at http://www.ibm.com/privacy/details the section entitled “Cookies, Web Beacons and Other Technologies” and the “IBM Software Products and Software-as-a-Service Privacy Statement” at http://www.ibm.com/software/info/product-privacy.

Trademarks

IBM, the IBM logo, and ibm.com are trademarks or registered trademarks of International Business Machines Corp., registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available on the Web at “ Copyright and trademark information ” at http://www.ibm.com/legal/copytrade.shtml.

Adobe, the Adobe logo, PostScript, and the PostScript logo are either registered trademarks or trademarks of Adobe Systems Incorporated in the United States, and/or other countries.

Java™ and all Java-based trademarks and logos are trademarks or registered trademarks of Oracle and/or its affiliates.

Linux is a registered trademark of Linus Torvalds in the United States, other countries, or both.

Microsoft, Windows, Windows NT, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Other names may be trademarks of their respective owners. Other company, product, and service names may be trademarks or service marks of others.

Part Number:

(1P) P/N: