Unit 6 – Developing with SAP Extension Suite

Certification: C_CPE_13

Unit 6 – Authorization and Trust Management

Agenda

- What have we learned so far
- CDS Restrictions and Roles
- Authorization and Trust Management
- Creating AppRouter
- Adding AppRouter to MTA
- Assigning Role Collections
- Summary points

Steps involved

1. Initialize full-stack project – Completed (Unit 1, 2)
2. Create the tables – Data Modeling – Completed (Unit 1, 2)
3. Generic handlers – Out-of-the-box CRUD functionality – Completed (Unit 1, 2)
4. Basic UI – Completed (Unit 3)
5. List Report layout – Completed (Unit 3)
6. Custom event handling – Business logic – Completed (Unit 3)
7. Support for external API – Completed (Unit 4)
8. Connecting to Sandbox – Completed (Unit 4)

Steps involved

9. Consume external service in UI – Completed (Unit 4)
10. Manual deployment to CF using manifest.yml – Completed (Unit 5)
11. Manual deployment to CF using mta.yml – Completed (Unit 5)
12. Security – Restrictions and Roles
13. Security – Authorization and Trust Management
14. Creating an AppRouter
15. Adding AppRouter to mta
16. CI / CD Pipeline

Step 12 – Restrictions and Roles

```
git checkout 11_restrictions_and_roles (Use tab for branch name)
```

Modified Files

- srv/risk-service.cds
- .cdsrc.json

Run command

- cds watch

Step 12 – Restrictions and Roles

```
service RiskService {
    entity Risks @(restrict : [
        {
            grant : ['READ'],
            to    : ['RiskViewer']
        },
        {
            grant : ['*'],
            to    : ['RiskManager']
        }
    ])                          as projection on rm.Risks;

    annotate Risks with @odata.draft.enabled;

    entity Mitigations @(restrict : [
        {
            grant : ['READ'],
            to    : ['RiskViewer']
        },
        {
            grant : ['*'],
            to    : ['RiskManager']
        }
    ])                          as projection on rm.Mitigations;

    annotate Mitigations with @odata.draft.enabled;
```

restrict - keyword for Risks entity

RiskViewer Role - Can only READ

RiskManager Role - Can do everything

restrict - keyword for Mitigations entity

RiskViewer Role - Can only READ

RiskManager Role - Can do everything

Step 12 – Restrictions and Roles

```
"users": {
  "alice@tester.com": {
    "password": "initial",
    "ID": "alice",
    "userAttributes": {
      "email": "alice@tester.com"
    },
    "roles": ["RiskViewer"]
  },
  "bob@tester.com": {
    "password": "initial",
    "ID": "bob",
    "userAttributes": {
      "email": "bob@tester.com"
    },
    "roles": ["RiskManager"]
  }
}
```

2 users are defined for local testing

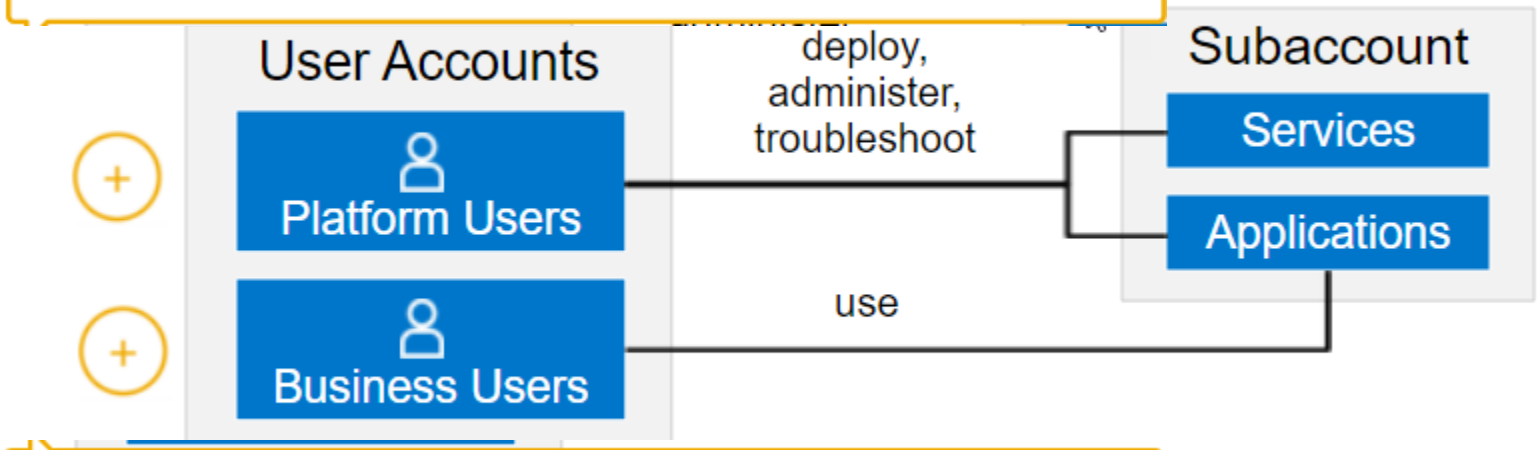
User alice has the RiskViewer role

User bob has the RiskManager role

Authorization and Trust Management

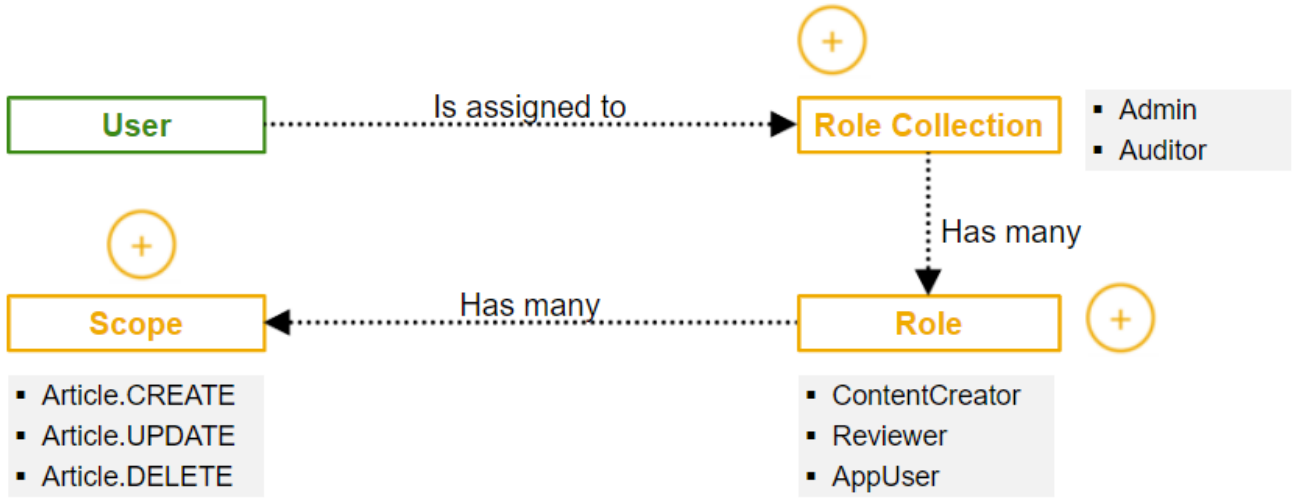
Platform users are usually developers, administrators or operators who deploy, administer, and troubleshoot applications and services on SAP BTP.

For platform users, the default identity provider is SAP ID Service.



Business users use the applications that are deployed to SAP BTP. For example, the end users of your deployed application or users of subscribed apps or services, such as SAP Business Application Studio or SAP Web IDE, are business users.

Authorization and Trust Management



- In SAP BTP, CF environment, a single authorization is called Scope
- Scopes cannot be assigned to users directly – They are packaged into Roles
- Scopes are prefixed with xsappname to make them uniquely identifiable

- Role has many Scopes

- Role-Collections contain 1 or more Roles
- Role-Collections can be assigned to a User

Step 13 – Authorization and Trust Management

```
git checkout 12_authorization_trust_management (Use tab for branch name)
```

Modified Files

- xs-security.json
- mta.yml

Run command

- cds watch

Step 12 – Restrictions and Roles

```
cds compile srv --to xsuaa >xs-security.json
```

Behind the scenes, CAP framework does the following

- Reads the authorization parts (@restrict …) from service definition file
- Creates scopes and role templates in xs-security.json file

Step 12 – Restrictions and Roles

```
"scopes": [
  {
    "name": "$XSAPPNAME.RiskViewer",
    "description": "RiskViewer"
  },
  {
    "name": "$XSAPPNAME.RiskManager",
    "description": "RiskManager"
  }
],
"attributes": [],
"role-templates": [
  {
    "name": "RiskViewer",
    "description": "generated",
    "scope-references": [
      "$XSAPPNAME.RiskViewer"
    ],
    "attribute-references": []
  },
  {
    "name": "RiskManager",
    "description": "generated",
    "scope-references": [
      "$XSAPPNAME.RiskManager"
    ],
    "attribute-references": []
  }
```

Step 13 – Authorization and Trust Management

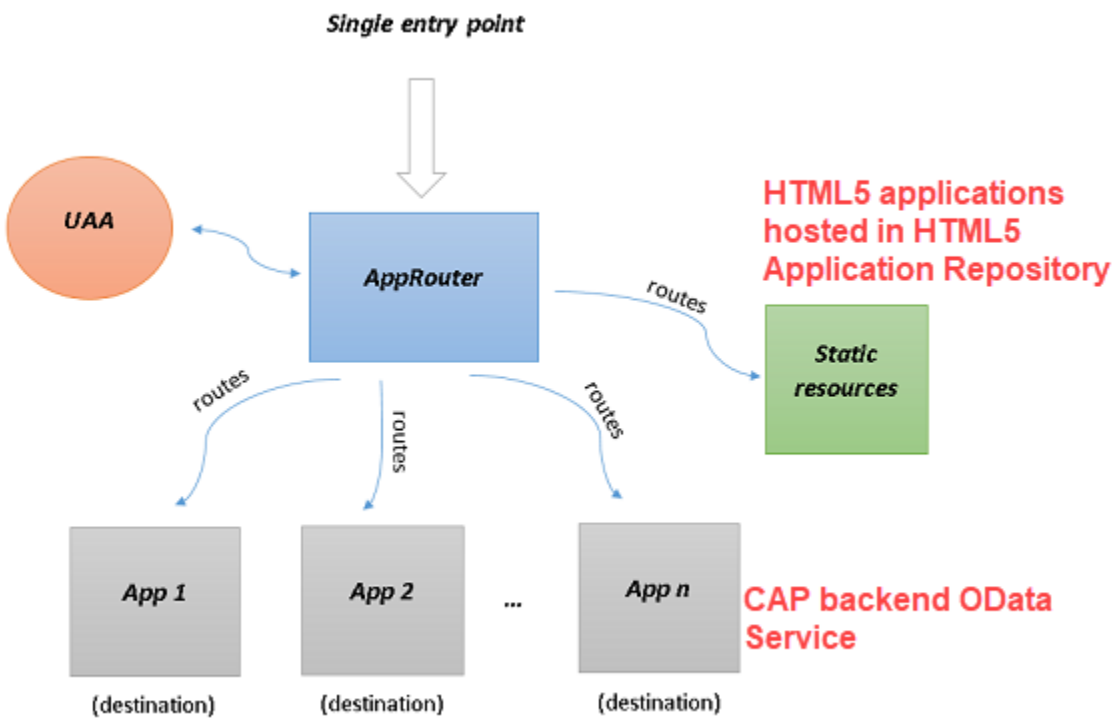
```
cds compile srv --to xsuaa > xs-security.json
```

Upate mta.yml

Run command

- `cds watch`

Step 14 – Create an AppRouter



AppRouter

- Routes request from browser to CAP Service
- Routes request from browser to provider of UI sources
- Ensures authenticated and authorized users get token from XSUAA Service and forwards it to CAP Service

Step 14 – Create an AppRouter

```
git checkout 13_approuter (Use tab for branch name)
```

New Files

- `approuter/*`

Run command

- `cds watch`

Step 14 – Create an AppRouter

Initialize Node.js project under approuter folder

```
npm init
```

Update package.json

```
"scripts": {"start": "node node_modules/@sap/approuter/approuter.js"}
```

Create xs-app.json

Run command

- cds watch

```json
{
    "welcomeFile": "/app/risks/webapp/index.html",
    "authenticationMethod": "route",
    "sessionTimeout": 30,
    "logout": {
        "logoutEndpoint": "/do/logout",
        "logoutPage": "/"
    },
    "routes": [
        {
            "source": "^/app/(.*)$",
            "target": "$1",
            "localDir": "resources",
            "authenticationType": "xsuaa"
        },
        {
            "source": "^/service/(.*)$",
            "destination": "srv-binding",
            "authenticationType": "xsuaa"
        }
    ]
}
```

Routes need authentication...

Path contains /app/
Retrieve from resources dir

Path contains /service/
Retrieve from srv-binding destination

srv-binding destination needs to be defined

Step 15 – Add AppRouter to MTA

```
git checkout 14_add_approuter_to_mta (Use tab for branch name)
```

Modified Files

- `mta.yml`

Deploy to CF

- `mbt build –t ./`
- `cf deploy risk-management_1.0.0.mtar`

Key Summary Points – Unit 6

JWT-based Authentication

This is the strategy to be used in production. User identity, as well as assigned roles and user attributes, are provided at runtime, by a bound instance of the 'user account and authentication' service (UAA). This is done in form of a JWT token in the `Authorization` header of incoming HTTP requests.

Prerequisites: You need to add passport to your project:

```
npm add passport
```

Prerequisites: You need to add @sap/xssec to your project:

```
npm add @sap/xssec
```

Configuration: Choose this strategy as follows:

```
"cds": { // in package.json
  "requires": {
    "auth": { "kind": "jwt-auth" }
  }
}
```

Key Summary Points – Unit 6

In *./package.json* Store Project Configurations

You can provide static settings in a *"cds"* section of your project's *package.json* as in the following example:

```
"cds": {
  "requires": {
    "db": { "kind": "sql" }
  }
}
```

In *./.cdsrc.json* Store Project Configurations

Alternatively, you can put static settings in *.cdsrc.json* file in your project root:

```
"requires": {
  "db": { "kind": "sql" }
}
```

Key Summary Points – Unit 6

Question 3

Choose the correct answer(s).

Which modules must you install to enable authentication support in CAP for BTP?

- [ ] xsjs
- [ ] xssecure
- [x] xssec
- [x] xsenv

Feedback

Correct. You must install xssec and xsenv to enable authentication support in CAP for BTP.

@sap/xsenv DT

3.2.2 • Public • Published 12 days ago

Readme | Explore BETA | 3 Dependencies

@sap/xsenv

Utility for easily reading application configurations for bound services and certificates in the SAP Cloud Platform Cloud Foundry environment, SAP XS advanced model and Kubernetes (K8S).

@sap/xssec

3.2.13 • Public • Published 2 months ago

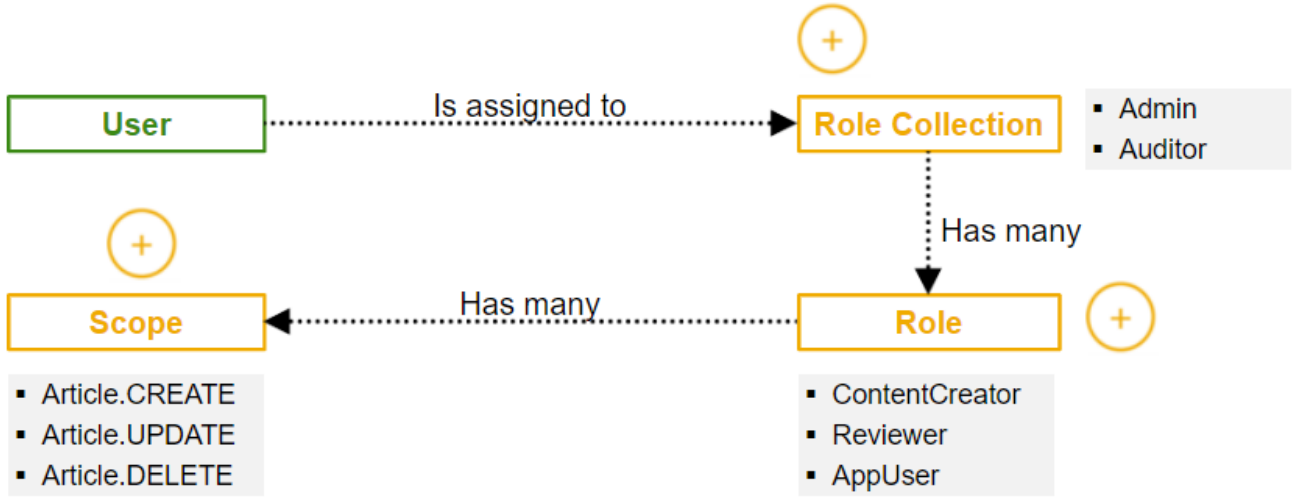
Readme | Explore BETA | 6 Dependencies | 34

@sap/xssec: XS Advanced Container Security API for node.js

XS Advanced Authentication Primer

Authentication for node applications in XS Advanced relies on a special usage of the OAuth 2.0 protocol, which is based on central authentication at the UAA server that then vouches for the authenticated user's identity via a so-called OAuth Access Token. The current implementation uses as access token a JSON web token (JWT), which is a signed text-based token following the JSON syntax.

Key Summary Points – Unit 6



- In SAP BTP, CF environment, a single authorization is called Scope
- Scopes cannot be assigned to users directly – They are packaged into Roles
- Scopes are prefixed with xsappname to make them uniquely identifiable

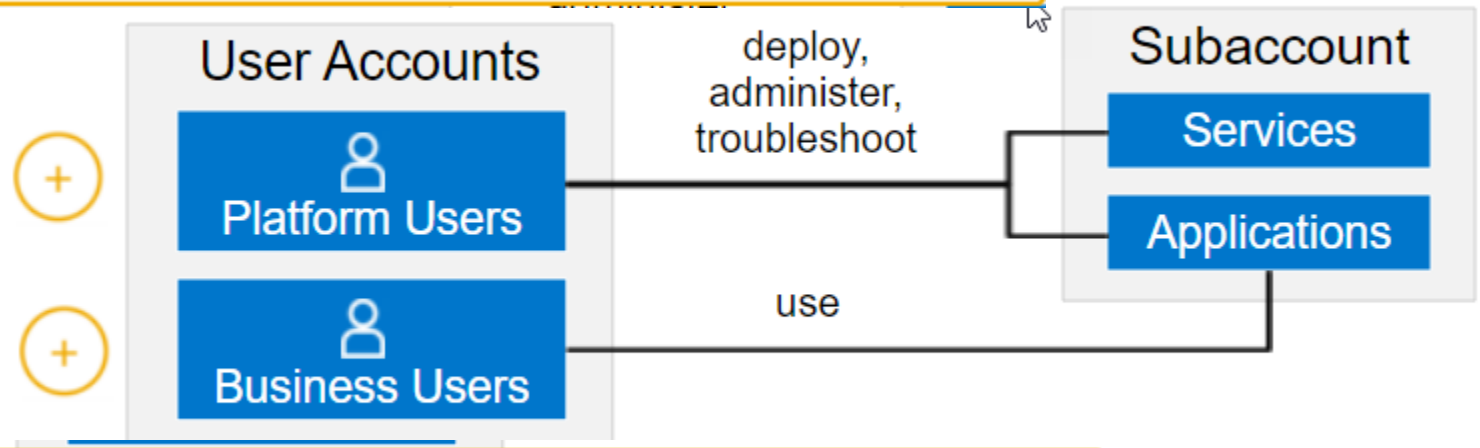
- Role has many Scopes

- Role-Collections contain 1 or more Roles
- Role-Collections can be assigned to a User

Key Summary Points – Unit 6

Platform users are usually developers, administrators or operators who deploy, administer, and troubleshoot applications and services on SAP BTP.

For platform users, the default identity provider is SAP ID Service.



Business users use the applications that are deployed to SAP BTP. For example, the end users of your deployed application or users of subscribed apps or services, such as SAP Business Application Studio or SAP Web IDE, are business users.

Key Summary Points – Unit 6

Question 8

Choose the correct answer(s).

What does the Extended Services - User Account and Authentication (XSUAA) service enable your app to do?

- [ ] Store "real" users.
- [ ] Identify users by address and social security ID.
- [x] Identify users by e-mail, userId, first and last name.
- [x] Check users' roles to allow or prohibit actions.

Feedback

Correct. XSUAA enables your app to identify users by e-mail, userId, first and last name and check users' roles to allow or prohibit actions.

Key Summary Points – Unit 6

`xs-security.json – Application Security Descriptor`

File that defines the details of authentication method and authorization types to use for access to your application

The contents of the `xs-security.json` are used to configure the OAuth 2.0 client; the configuration is shared by all components of an SAP multi-target application. The contents of the `xs-security.json` file cover the following areas:

- Authorization scopes
  A list of limitations regarding privileges and permissions and the areas to which they apply
- Attributes
  A list of as-yet undefined information or sources (for example, the name of a cost center)
- Role templates
  A description of one or more roles to apply to a user and any attributes that apply to the roles

Key Summary Points – Unit 6

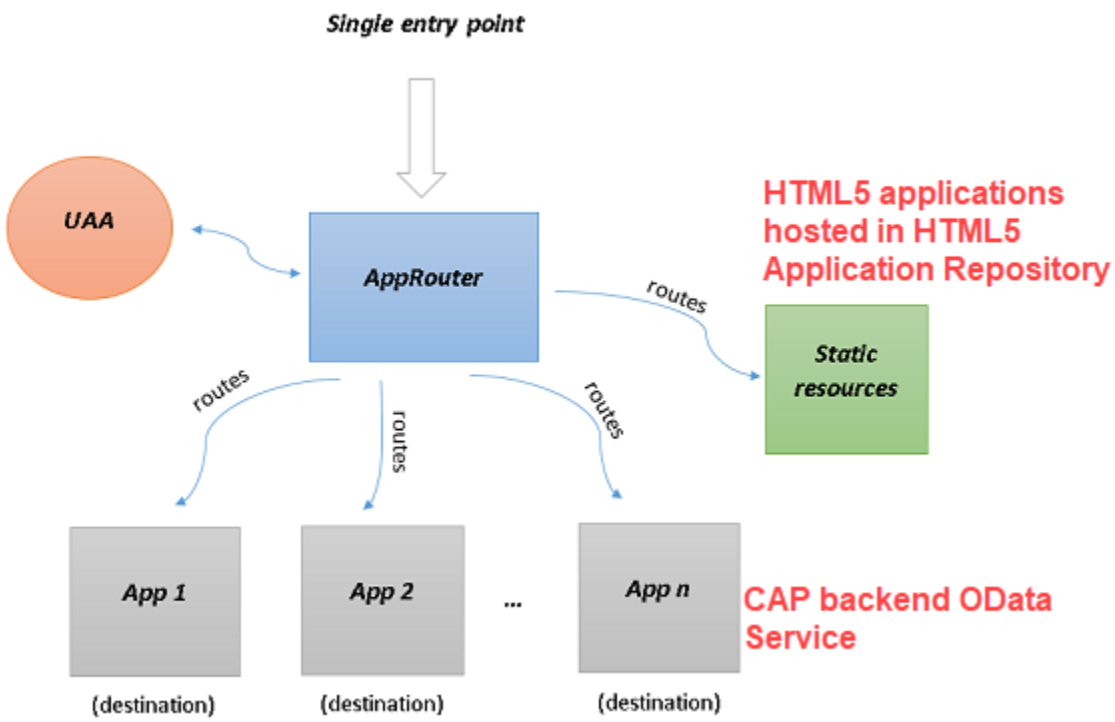
xs-app.json – Application Router Configuration

File contains configuration information used by the application router

package.json | xs-app.json | .cdsrc.json

```
1  {
2    "welcomeFile": "/app/risks/webapp/index.html",
3    "authenticationMethod": "route",
4    "sessionTimeout": 30,
5    "logout": {
6      "logoutEndpoint": "/do/logout",
7      "logoutPage": "/"
8    },
9    "routes": [
10     {
11       "source": "^/app/(.*)$",
12       "target": "$1",
13       "localDir": "resources",
14       "authenticationType": "xsuaa"
15     },
16     {
17       "source": "^/service/(.*)$",
18       "destination": "srv-binding",
19       "authenticationType": "xsuaa"
20     }
21   ]
22 }
```

Key Summary Points – Unit 6



AppRouter

- Routes request from browser to CAP Service
- Routes request from browser to provider of UI sources
- Ensures authenticated and authorized users get token from XSUAA Service and forwards it to CAP Service

Key Summary Points – Unit 6

Question 16

Choose the correct answer.

In the SAP BTP Cockpit, where can you assign role collections?

- ○ *Security → Roles*
- ✓ ● *Security → Trust Configuration*