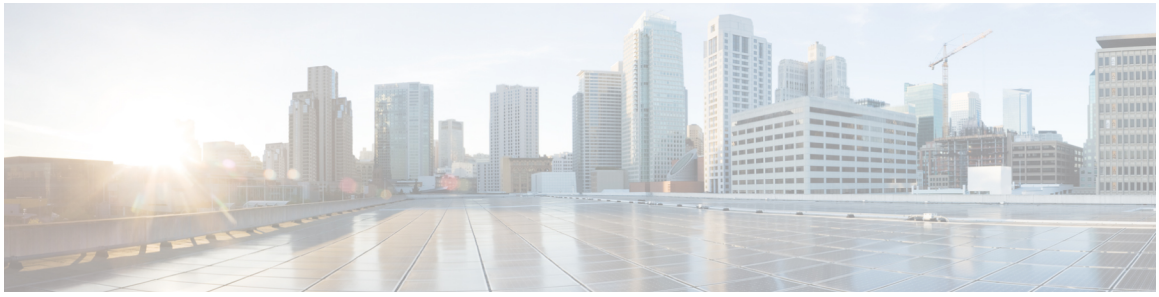

Creating Directory Structure for a Stateful Application

Use this procedure to create the directory structure and all the files required for developing a stateful application for the . See the *Appendix* for examples of the various files required to develop the application.

Procedure

Step 1 Create a directory for the app you are developing in your workspace. All the folders and files required for developing the application must be added to this folder.

Step 2 Create the metadata for the app in the `app.json` file.
This file is required and has information required by the to recognize the app and validate it. See _id_00000018WIA65A45A10GYZ_r_App_Metadata.xml#id_41316 for information regarding the metadata required for the `app.json` file.

Step 3 Create a `Media` folder and the files specified in this folder for your app.
This folder contains the following folders and files:

- `Readme` (Required) — The readme directory only contains the `readme.txt` file and cannot be empty. When you publish the app to the , the `readme.txt` file is used to present the information about the app to the user on the app description page in the .
- `License` (Required) — The license folder contains the `Cisco_App_Center_License.txt` file. It is the Cisco license file for the app and is added automatically when using the Cisco packager. Optionally, the developer can also add a separate app specific license file for the app in this location.
- `Snapshots` (Optional) — The snapshot folder contains files which provide a preview of the app before the user downloads the app from the . It is optional and provides information regarding the app in various modes.
- `IntroVideo` (Optional) — The IntroVideo folder is optional. It contains a video which introduces the app and give information on how the app works. The supported format for the video is mp4.

Step 4 Create a `Legal` folder and add the files containing the legal information required for your app.
The directory must include the following two files. These files are automatically provided when using the Cisco packager to package an app.

- `Cisco_App_Center_Customer_Agreement.docx`
- `Cisco_App_Center_Export_Compliance_Questionnaire.docx`

Step 5 Create a `UIAssets` folder and the files specified in this folder for your app.
The `UIAssets` folder is the core folder which contains all the intelligence about the app. This folder contains the HTML, CSS, and JavaScript files for the app. This folder must at least include the following files:

- `app.html` (Required) — A HTML file that implements the UI or the front-end of the application. The content of this file is specific to the app. This file contains the HTML page that will be embedded in APIC's UI. It can import various others files such as CSS or Javascript files provided within the UIAssets folder. This file must contain the function to use the tokens specified in `app-start.html`.
- `app-start.html` (Required) — A HTML file provided by Cisco and can be downloaded from Cisco DevNet. Every application must include this file for single-sign on to work. It is recommended that you do not modify this file.

  It contains the cookie information to implement the single sign-on in an application. It contains the cookie data and the mechanism to retrieve the data from APIC. This file must contain the data for the cookies, token and challenge. The value of the cookie is sent to APIC as headers as part of each request made from app's UI to avail single sign-on. This file also includes the loading sequence for an app. It contains a message which is displayed when the app is being loaded.

  It contains information to receive the tokens from the APIC and coverts the data into a cookie. You must then get the tokens used in a cookie and use it in further requests.

  APIC regularly sends a token to the application. The app must have the mechanism to receive and update its token accordingly. You can retrieve the token using `Ext.util.Cookies.get`, each time you make a request.

```
<script type="text/javascript">
    window.addEventListener('message', function (e) {
        if (e.source === window.parent) {
            var tokenObj = Ext.decode(e.data, true);
            if (tokenObj) {
// Setting the cookie with the tokens received by the APIC
                Ext.util.Cookies.set('app_' + tokenObj.appID + '_token', tokenO
bj.token);
                Ext.util.Cookies.set('app_' + tokenObj.appID + '_urlToken', tokenO
bj.urlToken);
            }
        }
    });
  </script>
```

  Another option for implementation, is to store the tokens from the cookie in variables. In this example, the application HelloAci uses `window.APIC_DEV_COOKIE` and `window.APIC_URL_TOKEN` when sending a request.

```
<script type="text/javascript">
   window.APIC_DEV_COOKIE = Ext.util.Cookies.get("app_Cisco_HelloAciStateful_token");

   window.APIC_URL_TOKEN =  Ext.util.Cookies.get("app_Cisco_HelloAciStateful_urlToken");


    window.addEventListener('message', function (e) {
```

```
        if (e.source === window.parent) {
            var tokenObj = Ext.decode(e.data, true);
            if (tokenObj) {
                window.APIC_DEV_COOKIE = tokenObj.token;
                window.APIC_URL_TOKEN = tokenObj.urlToken;
            }
} });
</script>
```

Step 6 Create a `Image` folder.
This folder contains the required docker image, `aci_appcenter_docker_image.tgz` for the application. A docker image contains all the packages required by the app to implement the backend. The image can contain packages such as Web server to open the API, OpenSSL for security, Cisco APIC Python SDK (cobra) for querying the APIC. The execution environment for the app should be provided in this image. See Creating a Docker Image on how to create a docker image and add it to the image folder.

Cisco also provides reference docker images and this image can be downloaded from Cisco DevNet. Cisco provides the following docker images:

- Docker image containing Cobra SDK.
- Docker image containing SQLite database, Cobra SDK, and Acitoolkit.
- Docker image containing MySQL database, Cobra SDK, and Acitoolkit.

Note The size of the docker image should not exceed 1 GB. Every stateful application must have a separate docker image. Sharing of docker images is currently not supported.

If you bring your own docker image or update the Cisco's reference image, you must first unzip or untar the `docker.tgz` file, then remove the `manifest.json` file, and finally tar or zip the `docker.tgz` file.

When the docker image is mounted, it contains the following directories located in `/home/app`:

- `src` — Contains all the source files for the app.
- `credentials` — Contains the private key to query the APIC.
- `data` — Contains the data for the distributed file system in the APIC cluster.
- `logs` — Contains the logs for the app that is collected as part of tech support.

Step 7 Create a `Service` folder and the files specified in this folder for your app.
This folder contains the service files.

- `start.sh` (Required) — It contains the first script that is executed after the docker container is installed. It includes all the initializations required for the application. It also allows you to start any script specified in the docker image.
- Other files (Optional) — This folder could contain a `server.py` file that runs a Web server providing an API for the application. In this case, `start.sh` file must contain the line starting `server.py`. In this release, only python is supported as an execution environment.