# Software Developer’s Guide to the ECC Configuration API

## Introduction

The ECC Configuration API is a library of method designed to aid in the creation of project configuration software clients. The API contains two separate interfaces: IECCDirectoryConfig which initialises a project repository and provides basic file and directory management methods. IECCProjectConfig provides more complex project specific methods such as management of project components and features. The interfaces provide methods which allow the developer to set up a document repository on a WebDAV enabled Apache 2 server and on local computer, and manage Experimedia project components and features. The API also provides a range of document and directory management methods. This guide instructs the developer on how to get started, the API’s repository management capabilities, and how to use the API interfaces and methods. A Javadoc is also available in the ECC documentation.

## Getting Started

To begin using the API a WebDAV service must be enabled on a web server, the WebDAV service must use either basic or digest authentication. The URL of what will be the root of the configuration repository must be declared in the ECCConfigAPIFactory. Experimedia provides a secure configuration repository using digest authentication, but if the developer chooses to use an alternative repository: a guide to setting up a secure repository is available in the ECC documentation. Figure 1 shows the declaration of the repository URL.

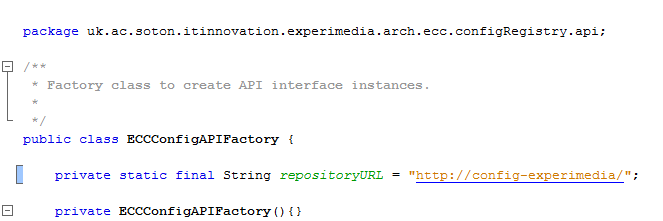


Figure : Declaring the configuration repository

To start using the API interface methods the developer must create interface instances using the factory methods with the parameters: project name, repository username and repository password. These are shown below:

IECCDirectoryConfig dc = ECCConfigAPIFactory.getDirectoryConfigAccessor( projectName, repoUsername, repoPassword );

IECCProjectConfig pcf = ECCConfigAPIFactory.getProjectConfigAccessor( projectName, repoUsername, repoPassword );

Once created the developer does not need to use the authentication parameters for any additional methods. It is recommended to create an IECCDirectoryConfig instance first before using any project configuration methods to insure that the project is correctly initialised. The API method calls are used as in the example below:

pcf.createComponentFeature( component, feature );

## Repository Management

When an instance of IECCDirectoryConfig is created a project directory is automatically created in the configuration repository using the project name parameter as a name if it does not already exist. This project directory will serve as the root directory and the API will route any project configuration operation to this directory. Any configuration data added to a component feature will also be routed accordingly as would any other method related to a specified component feature. Experimedia provides a default configuration repository to provide configuration data files for ECC Dashboard component features. When creating component features using the API will create directories and sub directories for them. The directory structure is shown below:

Figure : Configuration repository directory structure

The API also provides methods to use a locally stored repository, this feature would require the developer to declare the local repository root directory and use as a parameter, for example:

pcf.createLocalComponentFeature ( localDirectoryPath, component, feature )

## Interfaces and Methods

### IECCDirectoryConfig

This interface provides methods for initialising a new project and general document and directory management. It is recommended to create an instance of this interface before creating an instance of the IECCProjectConfig.

Create an instance of this interface using the following code:

IECCDirectoryConfig dc = ECCConfigAPIFactory.getDirectoryConfigAccessor( projectName, repoUsername, repoPassword );

### Methods

|  |  |  |
| --- | --- | --- |
| Return Type | Name and Parameters | Description |
| void | addDirectory(String directoryName) | Adds a new directory to the project repository. |
| void | copyDocument(String sourceFilePath, String destinationFilePath) | Copies a document to another location in the project repository. |
| void | createLocalDirectory(String directoryName) | Creates a directory on the local computer. |
| void | createLocalRepository(String directoryPath) | Creates multiple local directories using a complete path including drive letter. |
| void | deleteDirectory(String directoryName) | Deletes a directory in the project repository. |
| void | deleteDocument(String filePath) | Deletes a document in the project repository. |
| boolean | documentExists(String filePath) | Checks that a document or directory exists in the project repository and return a true or false value. |
| String | getDocument(String filePath) | Retrieves a document a specified document from the project repository and returns a string. |
| String | getLocalConfigPath() | Returns the location of the configuration repository if one exists. |
| void | moveDocument(String sourceFilePath, String destinationFilePath) | Moves a document from one location to another in the project repository. |
| void | putDocument(String sourceFilePath, String destinationFilePath) | Enables a document to be uploaded to the project repository. |
| void | setLocalConfigPath(String localConfigPath) | Sets the location of a configuration repository on a local computer. |

### IECCProjectConfig

This interface provided more complex methods for managing project features, components and configuration data.

Create an instance of this interface using the following code:

IECCProjectConfig pcf = ECCConfigAPIFactory.getProjectConfigAccessor( projectName, repoUsername, repoPassword );

### Methods

|  |  |  |
| --- | --- | --- |
| Return Type | Name and Parameters | Description |
| boolean | componentFeatureConfigExists(String component, String feature) | Determines whether a configuration file exists for a specified component feature. |
| void | createComponentFeature(String component, String feature) | Creates both directories for a component and a feature if they do not exist in the project directory. |
| void | createDefaultComponentFeature(String component, String feature) | Creates both directories for a component and a feature if they do not exist in the default directory. |
| void | createLocalComponentFeature(String localDirectoryPath, String component, String feature) | Creates locally stored directories for a component and feature in the specified local repository |
| void | deleteComponentFeatureConfig(String component, String feature) | Method to delete a configuration file from a specified component feature |
| String | getConfigData(String component, String feature) | Gets the configuration data for named EXPERIMEDIA component feature. |
| String | getDefaultConfigData(String component, String feature) | Gets the default configuration data for a named EXPERIMEDIA component feature. |
| String | getLocalComponentFeature(String localDirectoryPath, String component, String feature) | Retrieves file stored in a local repository and convert that file to a string. |
| String | getLocalConfigFile(String filePath) | Retrieves a file stored on the local computer and convert that file to a string |
| String | getProjectUrl() | Return the current project URL. |
| boolean | projectExists(String projectUrl) | Checks that a project exists in the repository. |
| void | putComponentFeatureConfig(String component, String feature, String jsonConfig) | Writes configuration data to ECC specific feature directory. |
| void | putDefaultComponentFeature(String component, String feature, String jsonConfig) | Use this method to write configuration data for ECC specific feature in the default configuration directory. |
| void | putLocalComponentFeature(String localDirectoryPath, String component, String feature, String data) | Method to put a configuration file into the specified locally store component feature directory. |