APPLICATION CONFIG FILE GUIDE

ELIXIR TECHNOLOGY PTE LTD

2 Kallang Ave, #05-12/13/14 CT Hub, Singapore 339407

Table of Contents

Application Config File Guide

1. Introduction

This guide describes the application configuration file (*application.conf*) in Ambience/Repertoire software suite (hereby known as the software). This file is location in the "/etc" folder in the software.

The application configuration file is used to configure the parameters and initial settings for the software.

When upgrading the software suite, it is advisable to compare the new *application.conf* file with the old one to see whether any new items have been added.

The list of parameters that can be configured are listed in the table below.

The following sections uses Ambience as example and some sections are only applicable to Ambience only.

Refer to the following websites for more information:

- https://docs.elixirtech.com/Ambience/2022.1/index.html
- https://docs.elixirtech.com/Repertoire/2022.1/index.html
- www.elixirtech.com

2. Security

2.1. Configure and Test Mail Server

When identity is added, an email is sent with randomly generated password to the user. When a user wants to change the email or password, a verification is sent via email as well.

If you have not set up an email server, the default behaviour is to store the emails in the "/mail" folder within the softare. This is usually for diagnosis or debugging purposes. It is recommended to set up a mail server at the start.

Below are two examples of how to set up a mail server.

Example 1: Uses Gmail

1. Gmail allows only OAuth2 authentication without weakening security. Visit https://console.developers.google.com/apis/credentails to set up a "*clientID*" and "*clientSecret*". Use these to generate a "*refreshToken*".
2. In the software root folder, navigate to the "/etc" folder. Open the *application.conf* file using a text editor. In the `elixir.mail` section, edit the following with the information obtained earlier accordingly.

```
elixir.mail {
  smtp = "gmail"
  gmail {
    host = "smtp.gmail.com"
    port = 587
    debug = true
    oauth2 {
      userName = "xxx@gmail.com"
      clientId = "XXXX"
      clientSecret = "YYYY"
      refreshToken = "ZZZZ"
    }
  }
}
```

3. Save the above edits in the *application.conf* file and start the server. New users with valid email addresses can now be created in the Identities module.

Example 2: Uses AWS

1. In the software root folder, navigate to the “/etc” folder. Open the *application.conf* file using a text editor. In the “*elixir.mail*” section, edit the following:

```
elixir.mail {
  smtp = "aws"
  aws {
    from = "<user@example.com>"
    host = "<hostname>"
    dnsResolver = ""
    port = 465
    user = "XXXX"
    password = "YYYY"
    connectionTimeout = 30000
    tls = true
    ssl = true
    authMechanism = ""
    debug = false
  }
}
```

2. Save the above edits in the *application.conf* file and start the server. New users with valid email addresses can now be created in the Identities module.

2.2. Use GitLab As Authentication

The Identities module in Ambience software provides a simple mechanism for authentication (determining who is logging in). If you already have an authentication system, such as an SSO, LDAP or Active Directory, then it is possible to use that as the authentication mechanism. This identity management system is built upon OAuth2, which is what makes it possible to plug in alternate authentication providers.

If an external authentication system is used, the Identities module is not needed and should be removed to avoid confusion.

This section describes the steps to set up GitLab as the authentication method to log into the software.

The steps are as follows:

1. Create an account in GitLab.
2. In GitLab, add the software as an application under your user.

   Note: The URL callback should be http://hostname:1740/authclient for Ambience. Use port 1730 for Repertoire. This is consistent with the setting in the *application.conf* file.
3. Change the hostname on your machine to point to the proper endpoint (i.e., the added application).
4. Add the user into Users module with the same name that was created in the GitLab server.
5. Go to the software root directory and go to the "/etc" folder. Open the *application.conf* file using a text editor.
6. Make the following changes in the `elixir.sso.client` section.

```
elixir.sso.client {
  cookie-name = "elx-amb"

  cookie-same-site = "Lax"
  openid-field = "name"
  openid-scope = "openid email"
  service-definition {
    elxsso {
      authorization = "https://<gitlab-host>/oauth/authorize"
      token = "https://<gitlab-host>/oauth/token"
      userinfo = "https://<gitlab-host>/oauth/userinfo"
      logout = ${sso-server-baseurl}"/simple-sso/logout"
      debug = false
      client {
        id = "[Your Application ID]"
        secret = "[Your secret]"
        endpoint = ${sso-client-baseurl}"/authclient"
      }
    }
  }
}
```

7. Save the *application.conf* file.
8. Restart the software server. Open a browser and key in "Localhost 1740" in the address bar and hit the enter key.

   For Repertoire, ley in "localhost: 1730" in the address bar.
9. Log into the software using the GitLab account.

2.3. Two-factor Authentication

Ambience/Repertoire softwaresupports Time-based One-time Password (TOTP) Two-factor Authentication (2FA). By default, 2FA is disabled in the *application.conf* file. To enable 2FA, edit the *application.conf* file in two areas:

1. Under the *simple-server* section, change `show-totp = false` to `true`. This is to allow the login dialog to include 2FA.

```
simple-server {
  clients {
    ambience {
      secret = "171ccf22-670a-43c2-ac79-05c44bf305e3"
      redirect = ${sso-client-baseurl}"/authclient"
      #login-page = "" # set resource file here to use a custom
login page for this client
      landing-page = "http://"${host}":"${port}"/"
      name = "Elixir Ambience"
      show-totp = true
    }
  }
}
```

2. Add a new line in the *application.conf* file. This will allow User Settings module to include 2FA setup, in which users can set up their own 2FA.

```
ambience.user-settings.enable-panel.totp = true
```

2.4. Passwords

2.4.1. For First Login

When the user logs in with the randomly generated password (i.e., first login), they will be forced to change the password immediately. This can be disabled by editing the setting in the *application.conf* file. In the `elixir.identity` section, edit the `changePassword: true` to `false`.

```
elixir.identity {
      ...
      on-reset {
        changePassword: true
      }
}
```

2.4.2. Using Login Dialog

Any user can reset their own password using the login dialog. The new password will be sent to the user's email.

2.4.3. Password Policy

Strong password protects your system from hackers and malicious software. Some passwords may appear strong but are relatively easily guessed. Key aspects of a strong password are length (the longer the better), a mix of letters (upper and lower case), numbers and symbols, and no ties to your personal or corporate information.

The `elixir.identity.password-policy` section in *application.conf* file allows administrators to define the policy of the password to be used in the software suite.

```
elixir.identity {
      ...
      password-policy {
        minLength: 1
        maxLength: 0
        ...
        mustHaveSymbolSet: ""
        ...
        disallowedRegex: []
      }
}
```

The minimum and maximum length of the password, whether to use upper and/or lower case characters, whether to include digit or symbol, and disallow certain regular expressions, etc. can be defined in this section.

It may be advisable to disallow common passwords based on regular expressions, so that users are not able to set common passwords. A common password for one organisation may be very different from the next. Rather than a fixed list of regular expressions, administrators can decide in the list of regular expressions that should be disallowed.

The `disallowedRegex: []` field is empty by default. You can add your desired regular expression to disallow in the brackets.

Below is an example to disallow both "password" and "passw0rd" (with a zero instead if an o), as well as any password that starts with "e1" (a common elixir easily guessed example).

```
disallowedRegex: ["passw[o0]rd","e1.*"]
```

2.4.4. Customise Password Email

You can customise the content of the email to be sent to the user when the user resets his/her password or when new user is added.

In the *application.conf* file, add the following:

```
elixir.simple-identity {
  email (
    subject = "Elixir Password"
    add-account = ""<html><p>An account has been created for you
in toolTitle at
    <a href=`${landingPage}`>localhost:1740</a>.</p>
    <p>Your user name is ${name}, your password is ${password}</p>
    <p>Please login and change it.</p></html>""
    reset-account = ""<html><p>Your password has been reset in
toolTitle at
    <a href=`${landingPage}`>localhost:1740</a>.</p>
    <p>Your user name is ${name}, your password is ${password}</p>
    <p>Please login and change it.</p></html>""
  }
}
```

For Repertoire, change the port to 1730.

2.4.5. Customise Password Verification Email

You can customise the content of the email to be sent to the user when the user changes his/her password or email address.

In the *application.conf* file, add the following:

```
ambience.user-settings.email.password-change {
  subject = "Customised Password Change Verification"
  body = ""<h2>Elixir Ambience Password Change</h2>
    <p>A change of password has been requested by ${name}.</p>
    <p>Please enter this approval code on the change password
page:</p>
    <p>${code}</p>
    <p>If you are an Elixir user and did not request this change,
please contact your administrator.</p>
    <p>If you take no further action, the password change will be
rejected.</p>""
}
ambience.user-settings.email.password-change {
  subject = "Customised Address Verification"
  body = ""<h2>Elixir Ambience Email Change</h2>
    <p>A change of email address has been requested by
${name}.</p>
    <p>Changed from ${oldEmail} to ${newEmail}.</p>
    <p>Please enter this approval code on the change emai
page:</p>
    <p>${code}</p>
    <p>If you are an Elixir user and did not request this change,
please contact your administrator.</p>
    <p>If you take no further action, the password change will be
rejected.</p>""
}
```

2.4.6. Customise Password Reset Email

You can customise the content of the email to be sent to the user when the user resets his/her password.

In the *application.conf* file, add the following:

```
elixir.sso.server {
  email.password-reset {
    subject = "Elixir Password Reset Request: ${clientTitle} user: ${name}"
  body = """<html><p>${clientTitle} Password Rest Request for user ${name}</p>
    <p>If you have requested a password reset, please click the following link:</p>
    <p><a href=`${resetPage}`>${resetPage}</a></p>
    <p>If you have not requested a password reset, you can choose to ignore this mail, no change has been made yet,
    or report to your administrator as someone has requested a reset for an account with this email address.</p></html>"""
  }
}
```

3. Database and File Size

3.1. Add Database

When MongoDB is installed, by default the database is named “eno”. You can add another database into MongoDB and include it the *application.conf* file so that the Ambience software is able to access it.

Two areas need to be edited.

1. The first area allows user to read and write to the database via datasets and import/export modules. In the location indicated in bold, replace it with your new database name.

```
#Where datasets can be read from and written to via datasets and
import/export modules
ambience.datasets.databases = ["eno","NewDatabaseName"]
ambience.import.databases = ["eno","NewDatabaseName"]
ambience.export.databases = ["eno","NewDatabaseName"]
```

2. The other area allows the user to include the new database so that Ambience can access it in MongoDB. In the `elixir.data.mongodb` section, add a new line as indicated in bold, replacing the database name.

```
elixir.data.mongodb {
  ...
  default {
    connectionString = "mongodb://"${mongodb}":27017"
    ...
    database {
      eno = "eno"
      NewDatabaseName = "NewDatabaseName"
    }
  }
}
```

This section is for Ambience only and does not apply to Repertoire.

3.2. Upload/Import/Export File Size

The current file size limit is set to 150 MB in the *application.conf* file. This is to prevent user from slowing down the server by uploading or exporting huge files.

You can increase or decrease the size file limit by editing the value in bold.

```
#  Used by UploadDownload and ImportExport Modules
akka.http.server.parsing.max-to-strict-bytes = 150m

#  Used by UploadDownload Module and ImportExport Modules
akka.http.server.parsing.max-content-length = 150m
```

This section is for Ambience only and does not apply to Repertoire.

3.3. Session Timeout Duration

Ambience software has inactivity session timeout of 15 minutes. This timeout duration can be changed in the *application.conf* file. In the `ambience.web` section, edit the `session-timeout = 15 minutes` to the desired duration.

```
ambience.web {
  ...
  session-timeout = 15 minutes
  enforce-single-session = false
  ...
}
```

To disable the timeout, use the value `never`.

3.4. Session Cleaner

The software keep the sessions for seven days by default. A session will typically expire due to inactivity of 15 minutes. The seven days will begin from the last activity. So you can still keep a session alive for a month, as long as the mouse is moved at least once every 15 minutes.

You can change the duration for the sessions to be kept in the system in the *application.conf* file. You can add following in the *application.conf* file.

```
ambience.session-cleaner {
  expiry: 2 days
}
```

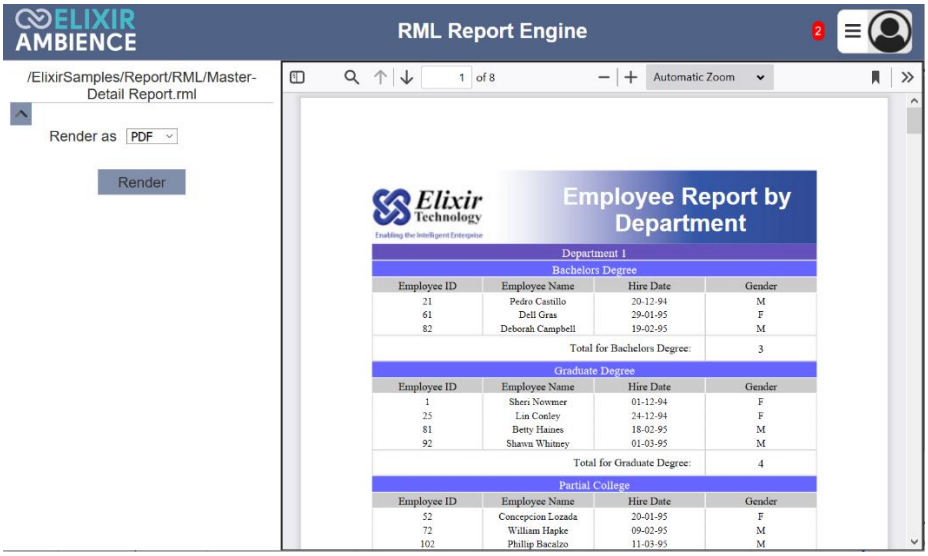
The configuration above allows the software to remove sessions that are more than two days from the last activity.

4. RML

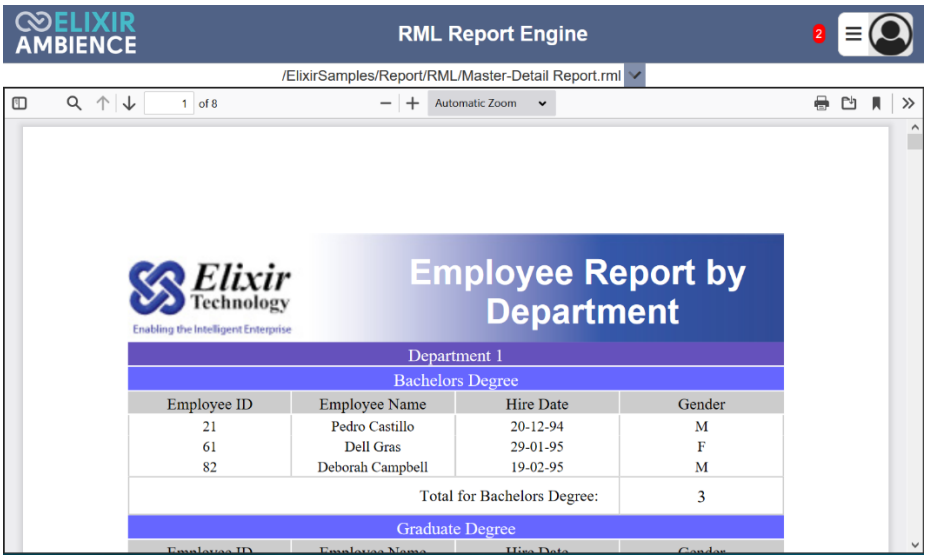
4.1. RML Report Engine Left Panel

When generating a RML report, the report by default fills the entire area in the RML Report Engine page. You can enable the parameters on the left of the page. To do so, add the following line in the *application.conf* file.

```
ambience.rml-engine.ui-left-parameters = ["application/pdf"]
```



You can change the `pdf` to other types if your browser supports viewing them. You can toggle the left panel off by clicking on the ^ icon.



To toggle on, click on the ⌄ icon on the top row.

4.2. Fonts In RML Reports

You can use extra fonts (non-OS) in the RML reports. To do so, add an extra line in the *application.conf* file.

```
elixir.rml.fonts.path = "/<pathname>/Fonts"
```

5. Currency Symbol In XLSX Output

By default, the currency symbol used is (¤) is used when an XLSX output is generated. This may not be the desired symbol to use.

You can use the currency symbol "$" XLSX output for the RML reports and for dashboard export to XLSX.

To do so, add an extra line in the *application.conf* file:

- For RML reports

```
elixir.rml.xlsx.currency-symbol = "[$$-4809]"
```

- For dashboard export to XLSX format (for Ambience only)

```
ambience.dashboard.pivot { xlsx-export { currency-code: "[$$-409]" } }
```

6. ETL

6.1. ETL Logs

By default, the ETL job logs is created as the job runs. These logs allow you to observe the behaviour of the jobs. But these logs may not be useful for production systems as they may run the ETL jobs 24 hours a day, thus generating huge volumes of log. This may slow down the system.

You can turn off the job log by adding the following into the *application.conf* file.

```
ambience.etl.logging.write-to-server-log: false
ambience.etl.logging.write-to-job-log: false
```

6.2. Apose Extensions for ETL Steps

To enable the Apose extensions for ETL steps, you require a separate license. Add the following into the *application.conf* file.

```
ambience.docx-engine.aspose {
  enabled = true
  words-license = "/<pathname>/Aspose.Words.Java.lic"
}
```

7. Config Editor

The Config Editor module in Ambience allows you to edit the configuration and save the changes onto MongoDB. This allows you to edit at one location and let other servers share the same configuration without the need to duplicate the *application.conf* file.

Before that can be done, the following need to be added into the *application.conf* file to allow the configuration in the Config Editor to be loaded.

```
ambience.config-loader.mongodb.enabled = true
```

This section is for Ambience only and does not apply to Repertoire.

8. Audit Log

Audit log will log any actions taken in the modules. Workflow module stores data either in `elxPublic` (server only) or `elxPrivate` (shared within browser). It does not know whether any data it is logging is sensitive. You can turn on or off the logging by adding the following into the *application.conf* file.

```
ambience.modules.workflow.audit {
  public: true
  private: false
}
```

The default will only show the public changes. Set `private: true` to include `elxPrivate` changes.

This section is for Ambience only and does not apply to Repertoire.

9. Redirect Using ETL

To use ETL chainset to redirect a URL, there needs to be an ETL user which has a role that is added to the appropriate chainset. This is needed as the user being redirected may not have permission to run ETL. So, this user acts as a proxy to run the chain the redirect uses.

In the *application.conf* file, add the following:

```
ambience.modules.redirect-edit.etl-user = "<username>"
```

This section is for Ambience only and does not apply to Repertoire.