

---

# **Gawati Docs Documentation**

*Release 1.0*

**The Gawati Community**

**Nov 16, 2018**



---

# Contents

---

<b>1</b>	<b>What is Gawati</b>	<b>3</b>
1.1	What is Gawati and what problem is it trying to solve ? . . . . .	3
1.2	Here is how Gawati works . . . . .	3
<b>2</b>	<b>About Gawati</b>	<b>7</b>
2.1	What is Gawati . . . . .	7
2.2	What Gawati is not . . . . .	7
2.3	Gawati description . . . . .	7
2.4	Licensing and Contribution . . . . .	7
<b>3</b>	<b>Installation</b>	<b>9</b>
3.1	Server Setup . . . . .	9
3.2	Code Deployment and Build . . . . .	13
3.3	Release Versions . . . . .	13
<b>4</b>	<b>System Overview</b>	<b>15</b>
4.1	Gawati Architecture . . . . .	15
4.2	Gawati Data Server . . . . .	19
4.3	How Trust & Security Works in Gawati . . . . .	20
<b>5</b>	<b>Development</b>	<b>27</b>
5.1	Introduction for Developers . . . . .	27
5.2	Developing Gawati . . . . .	28
5.3	Coding Guidelines . . . . .	50
5.4	Using VSCode for Development . . . . .	53
5.5	Data Server APIs . . . . .	55
5.6	Package Versions . . . . .	65
5.7	Jenkins Setup . . . . .	65
5.8	Installer . . . . .	66
<b>6</b>	<b>Indices and tables</b>	<b>79</b>



This is the documentation site for the [Gawati Project](#). The current development (unstable) version of Gwati is showcased on the [Portal demo](#) and [Editor demo](#) pages. For feedback or open questions, you can contact us on [our Google Group](#).



### 1.1 What is Gawati and what problem is it trying to solve ?

Gawati is a Legal Data Exchange platform that allows Legal documents to be Authenticated, Shared and Published. If you are a custodian of legal documents, and want to publish them online, Gawati may be ideally suited to fulfil your requirements.

Gawati lets you *own* your legal data and *authenticate* it via digital signatures, and also *control* how you want it published, while at the same time making it *searchable and accessible* using open standards.

Conventional platforms like DSpace and Fedora Commons have very rigid formats for storing documents and are spread across various relational database tables, making it difficult to access your data in external systems, or to even migrate your data out if you want to move to a different system.

Gawati makes use of [Akoma Ntoso XML](#) an open standard for Legal and Legislative documents to capture all content and metadata. This allows easy portability of data if ever you want to take your data out of gawati and move to a different system.

We do this quite differently. Our distributed / federated architecture allows syncing legal data across servers.

You can run gawati as standalone as described above, but you can also run it in a federated setup, where multiple gawati instances running independently push data onto a central aggregator:

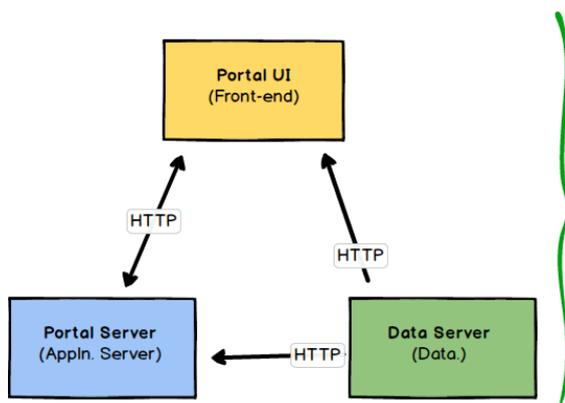
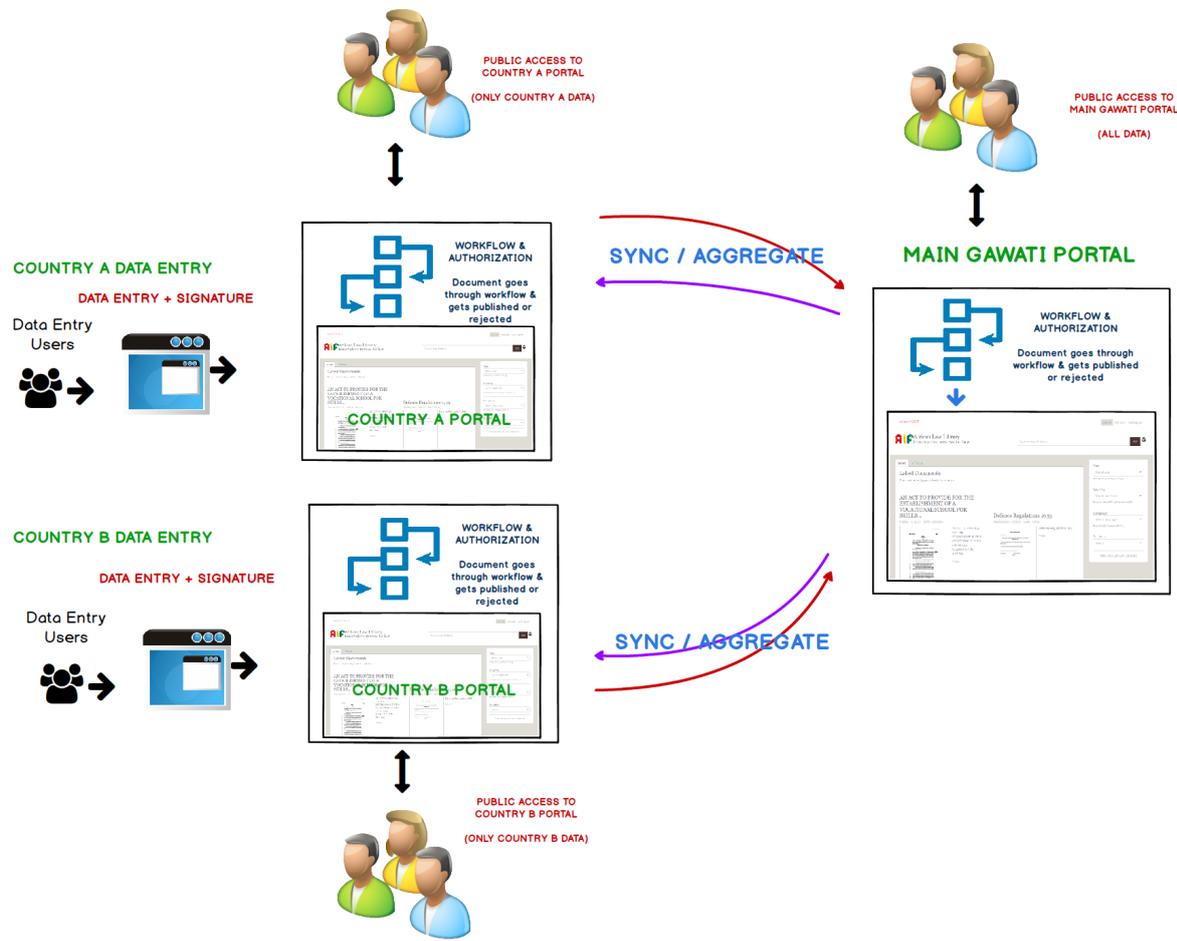
### 1.2 Here is how Gawati works

Your typical way to access Gawati would be via the online Portal - where you would search and view legal data.

(for a more Technically oriented explanation click [here](#) )

The Portal follows a conventional architecture which is popular nowadays of having different components which integrate via HTTP REST services. The portal is primarily three components:

- Data Server component: this serves all the Legal documents in Akoma Ntoso XML format. It provides to search and browse for legislation.
- Application Server component - the application server does some processing (for e.g. summarization) of the legislative data to make it easier for the front-end to process large volumes of data.



- UI Front-end: this shows the web-page being accessed by the user, and is the primary interface by which a user interacts with the system.

Adding documents to the Portal is done via a web-based Client interface, which has a similar architecture.



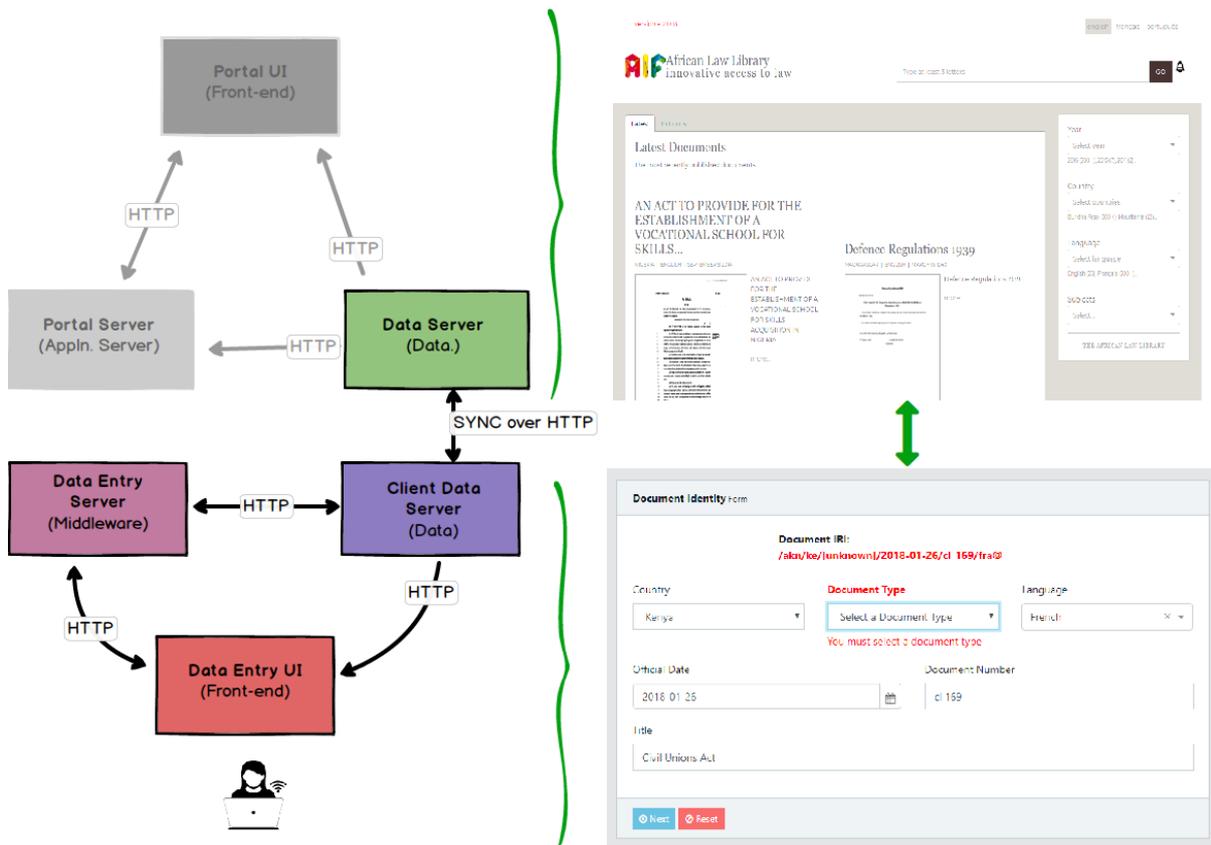
(for a more Technically oriented explanation click here )

**The data entry client is composed of three components:**

- Client Data Server component: this stores all the documents in AKoma Ntoso XML format.
- Application Server component - this provides access to services like Workflow, Data conversion and other processing which needs to be done on the server.
- UI Front-end: this provides the data entry forms and validates user input.

So how does the Data entry client interface with Portal data ?

Since it is only data that is interfaced between the data entry system and the portal, this interfacing is also done over ... HTTP !





## 2.1 What is Gawati

Gawati is an open source document library in the sense of a document and metadata repository with web based access. In its initial version 1.0 its interfaces will concentrate on legal documentation as it is implemented as an open source project for the [African Law Library](#).

## 2.2 What Gawati is not

Gawati in version 1.0 is not going to be equipped with lend/borrow or customer management components. However we think it's well suited to extend it in that regard.

## 2.3 Gawati description

For the enduser, Gawati allows to search a large library of documents in an intuitive and flexible way. It will support the users providing context to search results so they can identify document relationships, for example different translations, editions, versions or validity over time.

For data maintenance it provides workflows for professional data entry services, individuals as well as community involvement using role based permissioning. Using synchronisation capabilities, different parties can join their efforts and build upon each others work.

For the librarian Gawati is an entity-relationship based [FRBR](#) metadata and document storage, search engine and browser. The metadata is extended with full [Akoma Ntoso](#) support for legal document management.

## 2.4 Licensing and Contribution

Gawati source code is licensed under [AGPLv3](#). For including your source code contribution we will have to ask you to for a contribution agreement which you can find in the [Introduction for Developers](#).



### 3.1 Server Setup

#### Table of Contents

- *Installation script*
  - *Quickstart*
- *Firewall with SSL inspection*
- *Common configurations*
  - *monitoring email address*
  - *Gawati server URL*
  - *SSL server certificate*
    - \* *locally signed certificate*
    - \* *letsencrypt signed certificate*
  - *builduser*
- *Installation targets*
- *Components overview*
  - *Jetty*
  - *eXistdb*
  - *Downloads*
  - *Uninstalling*
- *References*

### 3.1.1 Installation script

The server setup will install all required components in the OS from [CentOS](#) (RedHat compatible) repositories or respective project sources where no CentOS packages are available. It will configure the OS and all services as a ready to run system.

If you want to try Gawati on a local virtual machine, please follow the related instructions in [Gawati on a server / VM](#) to install Virtualbox and download / import our preinstalled CentOS 7 image, then continue here.

The installer is written for [CentOS 7](#) (RedHat 7 compatible). CentOS / RedHat “Minimal installation” type is sufficient.

To download the [installation script](#), switch to user root and execute:

```
cd
curl http://dl.gawati.org/dev/setup -o setup
chmod 755 setup
```

### Quickstart

The installer **must be run as user root directly** (no sudo). It will switch accounts as needed.

To run the installer simply run:

```
./setup
```

The installer works in two steps. On first invocation it downloads a Gawati configuration template with our default settings. On second run the installation is executed according to the settings in this configuration file.

Running the installer twice will complete our default installation, assuming you will access your server using the base URL <https://my.gawati.local>. Additional services will be installed on subdomains thereof. In total, you will need to configure all of the following names resolve to your server IP locally (ie using your hosts file):

```
my.gawati.local
data.my.gawati.local
edit.my.gawati.local
media.my.gawati.local
```

---

**Note:** The Installer will automatically set the Admin password for eXist and display it to you in a summary when the installation completed. You will need to copy and paste this from the screen or note it down somewhere as it is the only time when the password is shown to the user.

---

If that’s all you need, you may finish reading here. Below you find more information for customising common configuration items and the key information of what is going to be installed.

### 3.1.2 Firewall with SSL inspection

If you run a firewall that does SSL interception replacing server certificates, you must add your firewalls CA to the Gawati server as a trusted CA. To do so, retrieve your firewall CA in pem format from your firewall, copy it onto the Gawati server (SSLinterceptCA.crt in below example) and execute the following commands to add it as a trusted CA:

```
yum install ca-certificates update-ca-trust force-enable cp SSLinterceptCA.crt /etc/pki/ca-
trust/source/anchors/ update-ca-trust extract
```

### 3.1.3 Common configurations

#### monitoring email address

In the section [fail2ban], change the variables *mailsender* (default: `from@sender.domain`) and *mailrecipient* (default: `root@localhost`).

#### Gawati server URL

In the ini file that has been downloaded to your home folder after the first run of the installer, find the section [gawati-portal] and change the *GAWATI\_URL\_ROOT* variable (default: `my.gawati.org`).

#### SSL server certificate

Gawati works with 2 distinct URLs. One as the main website URL (eg `my.gawati.local`) and one for the media / file repository providing static content derived from the main URL (`media.my.gawati.local`).

Your SSL certificate names must match your Gawati server URL. The installer offers two mutually exclusive options for creating a certificate.

- signed locally (use for internal server / testing)
- signed by letsencrypt (use for public servers)

Between both options there is a shared set of information identifying the owner of the generated certificates. Please adapt the respective section in [options] to your case:

```
[options]
...
organisation=ACME Installation Corp Ltd
country=CH
state=Zug
city=Zug
```

#### locally signed certificate

Creating such a certificate can be done without any external dependencies. It's meant for running internal or testing servers. In section [acme] make sure to configure *type=disabled*. In section [localcerts] set *type=install* and set variable *certs* identical to your *GAWATI\_URL\_ROOT* and add a whitespace followed by the equivalent of media. *GAWATI\_URL\_ROOT*.

#### letsencrypt signed certificate

For this, your Gawati server URL and certificate name must be resolvable via public DNS and public HTTP requests for it must arrive at your Gawati server on port 80. If those conditions are met and you intend to make your server publicly available, this is the preferred option.

In section [localcerts] make sure to configure *type=disabled*. In section [acme] set *type=install* and set variable *certs* identical to your *GAWATI\_URL\_ROOT* and add a whitespace followed by the equivalent of media. *GAWATI\_URL\_ROOT*.

#### builduser

After installing eXist application servers, the installer will retrieve code from github, compile and deploy it into these eXist instances. To do this, the installer creates a user dedicated for compiling Gawati components from source. This avoids compiling as root and interfering with existing user environments. The name of this user account is defined by the *builduser* user item in the [gawati-portal] section.

### 3.1.4 Installation targets

When you run the installer for the first time, it will download an additional file “dev.ini” into your home folder. The ini file defines the details of the installation. We call this an installation target.

With the second execution of the installer, installation commences according to the configuration in the ini file.

To choose a different profile to install, provide it as a commandline parameter, for example:

```
./setup prod
```

At this time, the default target “dev” is the only installation target provided by us.

You can change ours, or create your own ini files if you need to deviate from our defaults.

### 3.1.5 Components overview

The Gawati reference server is based on CentOS 7, Minimal Install. For hosting the application, we use eXistdb as XML/document database and jetty as Java web application server.

A production installation of Gawati will be installed with (2) instances of eXistdb

1. Gawati-Editor, internal management of the Gawati data
2. Gawati-Portal, data copy for public access

A development installation will serv both function off a single installation.

All services except for a (1) frontend Apache instance will be listening on 127.0.0.1 only.

#### Jetty

jetty binaries will be installed into /opt for shared use. It will be configured with configuration files in “start.d” directory.

The Gawati jetty-base environment will be installed into a separate user account. A JETTY\_BASE folder will be created in that users ~/apps/ folder. A link to its jetty installation in /opt will be created inside JETTY\_BASE called “jettyserver”. JETTY\_HOME will be configured as JETTY\_BASE/jettyserver.

Jetty will be installed as a system service starting with the boot process.

#### eXistdb

eXistdb will be installed using a dedicated user account. The name of the user account is defined in the setup configuration (eg dev.ini). eXistdb will be installed in folder ~/apps/existdb with data in ~/apps/existdata. A random generated password will be configured for existdb user “admin” and is displayed during installation.

eXistdb will be installed as a system service starting with the boot process.

#### Downloads

Installation Resources will be downloaded into “/opt/Download”

#### Uninstalling

There is no proper uninstaller yet, but if you installed the system with our default installation paths and service names, you can use the script at /opt/Download/installer/uninstall.sh to remove all files related to Gawati.

### 3.1.6 References

- [setup-installationsystem](#).

## 3.2 Code Deployment and Build

### 3.2.1 Introduction

Once you have completed the *setup essentials*, you will need to deploy the code from github into the system. The steps below describe the complete installation process and the tools. Note that - if your intention is just develop / contribute on a particular module, you **don't need** to do these following steps, you can as well clone the specific repositories and *contribute to Gawati* .

## 3.3 Release Versions

---

**Note:**

---



### 4.1 Gawati Architecture

The Gawati architecture is described in this document.

Gawati is composed of 2 different applications which have different audiences in mind.

1. The Gawati Portal - this is a public facing portal system that allows searching and accessing legal documents
2. The Gawati Data Input System - which is a back-office application that allows managing and inputting legal documents which are presented by the Portal.

#### 4.1.1 Gawati Portal System

The Gawati Portal is composed of different application components which are indicated below in the diagram.

The Portal System has its own dependencies which need to be installed before it can be used:

- gawati-portal-ui - the UI web front-end
- gawati-portal-fe - the back-end service for the UI
- gawati profiles system - see below *Gawati Profiles System* which has its own dependencies
- gawati-data - the Database service component
- gawati-portal-publisher - this is a subsystem component that allows publishing data onto the portal that has been received from the *Gawati Editor System*.
- gawati-portal-qprocessor - this is a subsystem component that allows receiving data from the *Gawati Editor System*.

#### 4.1.2 Gawati Editor System

While the *Gawati Portal System* provided access to documents, the documents are published onto the Portal via the Gawati Editor System described here.

The Editor System is composed of different application components which are indicated below in the diagram.

Functionally, the data input system looks like as below:

## PORTAL SYSTEM

UI (User Interface) <https://github.com/gawati/gawati-portal-ui>

Portal FE <https://github.com/gawati/gawati-portal-fe>

Portal Data <https://github.com/gawati/gawati-data>

Portal Data <https://github.com/gawati/gawati-data>

Portal QProcessor <https://github.com/gawati/gawati-portal-qprocessor>

Portal Publisher <https://github.com/gawati/gawati-portal-publisher>

## EDITOR SYSTEM

UI (User Interface) <https://github.com/gawati/gawati-portal-ui>

Editor FE <https://github.com/gawati/gawati-editor-fe>

Editor Data <https://github.com/gawati/gawati-client-data>

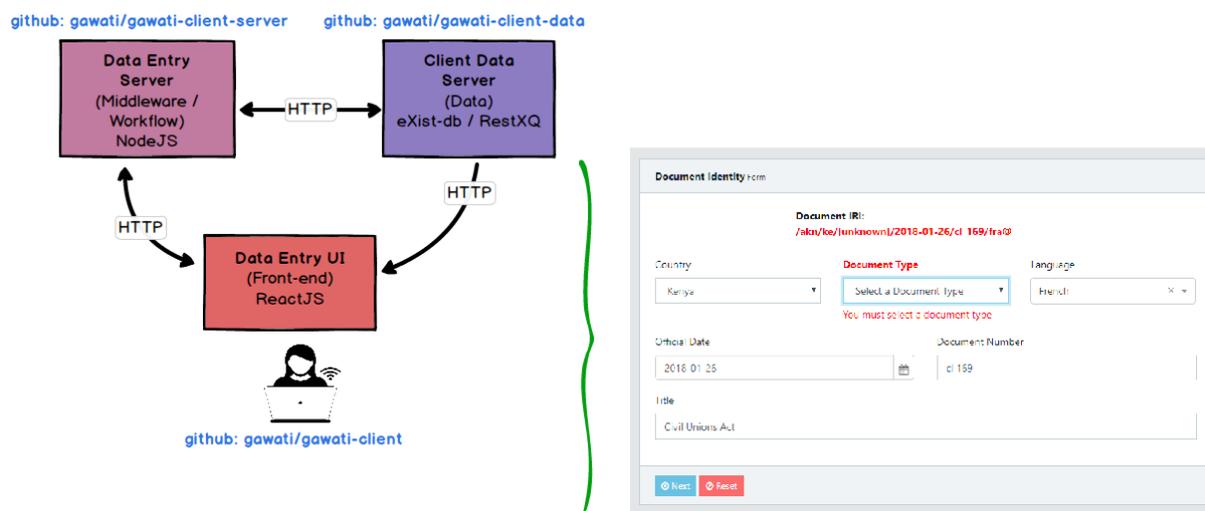
Digital Signature FE <https://github.com/gawati/gawati-digisign-fe>

Package Signature <https://github.com/gawati/gawati-package-sign>

Automated Tagging <https://github.com/gawati/gawati-tagit>

PDF to Xml conversion <https://github.com/gawati/pdf2xml-service>

Editor QProcessor <https://github.com/gawati/gawati-editor-qprocessor>



The Gawati Editor system has been conceived as an offline / online data entry system which can be used to publish information onto the *Gawati Portal System* even if offline. This is to cater for scenarios where data input is being done from places where internet connectivity is not always stable.

The Gawati Editor system uses a [message-queue](#) based architecture to support such a functionality.

## Architecture in Depth

The Editor System follows the same basic architecture as that of the Portal system, and provides a browser based experience for adding and managing documents:

- `gawati-editor-ui` - which is the front-end client which provides the UI and UX for the editor system. It encompasses a dashboard, a workflow and a form based system to manage content (see `gawati-editor-ui`).
- `gawati-editor-fe` - which provides front-end services that the `gawati-editor-ui` can interact with.
- `gawati-client-data` - which provides back-end services that are primarily used for getting data in and out of the XML database. The front-end `gawati-editor-ui` client does not access this directly but accesses this service via `gawati-editor-fe` which acts an authentication proxy for these services.

In addition to these there are some supporting services which are required by the editor system, that provide functional support for various other services:

- **Full text extraction** - a lot of content in Gawati is PDF based. We extract the full text of the PDF document as XML and make that searchable by integrating it with the main metadata search. This full-text extraction is provided by a separated full-text extraction service. (See `pd2xml-service`)
- **Automatic Tagging** - a document whose full text has been extract can be automatically tagged by a specialized service which uses a trained legal model to identify and generate tags based on the document's content. (see `gawati-tagit`).
- **Digital Signatures** - Document packages in Gawati can be digitally signed and published. This digital signature functionality is provided on the back of 2 services:
  1. **package signature service**: this accepts a gawati document as a gawati package and a private key to sign the document, and returns the signed document. (See `gawati-package-sign`). This service also allows validating a document based on its public key.
  2. **digital signature interaction service**: this is intended (and recommended) to be run on the `host computer` signing the document.(See `gawati-digisign-fe`). For more information how digital signatures work in Gawati see About Digital Signatures.
- **Asynchronous Publication** - The gawati editor system uses an asynchronous publication mechanism to move documents

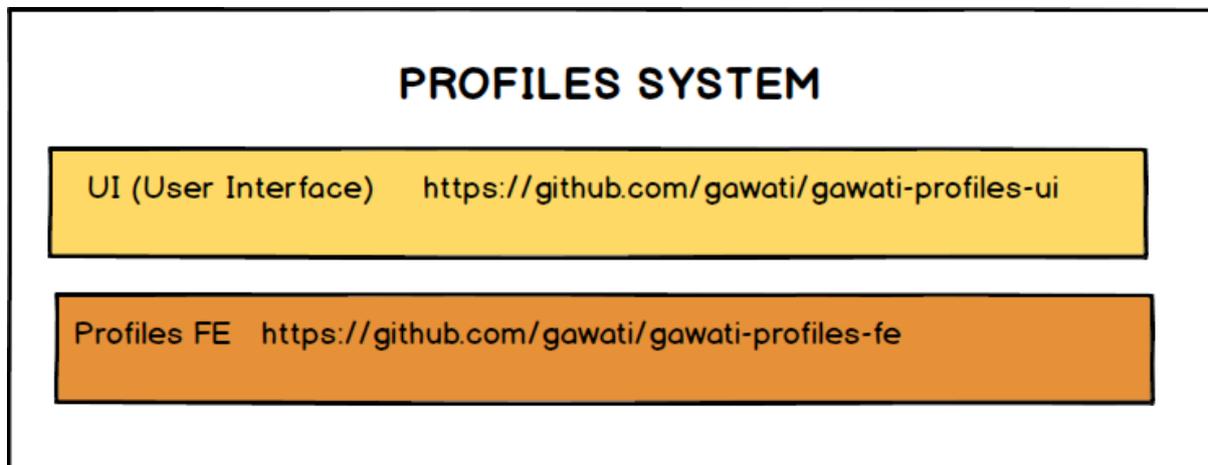
1. RabbitMQ - this is an enterprise grade Message Queue system which we just use out of the box for its message queue capabilities.
2. Editor QProcessor - this is a specialized service that runs as a background process in the Editor system and periodically scans the message queue for publication events. It transmits documents which are to be published onto the portal system. (See [Editor QProcessor](#)).

### 4.1.3 Gawati Profiles System

The Gawati Profiles System allows authenticated users in the Gawati system to have user profiles.

The profiles system is technically composed of two main components:

- Front-end : [gawati-profiles-ui](#)
- Back-end service: [gawati-profiles-fe](#) which in turn requires a MongoDB database back-end (see [inst-prerequisites](#)).



The profiles system does not provide authentication or a user directory on its own, for that it simply integrates with authentication and user directory services provided by KeyCloak (see [inst-prerequisites](#)).

The profiles system integrates with the portal system (see [Gawati Portal System](#)) using KeyCloak single-sign-on (SSO) if the user logs into the profiles system, they are transparently logged into the portal. The Profiles system allows storing extended user information e.g. in the portal users can save their favourite searches. In the user interface it may appear as if it is the portal that is providing this information in reality this served by the profiles service.

## 4.2 Gawati Data Server

[Gawati Data Server](#) is the document repository for Gawati documents.

It provides REST services to provide access to the data from other applications, the principal application being the Portal.

The document repository can reside on the same app-server installation as the Portal, or on a different application server or an entirely different physical server.

The only interface provided to read and write data to the document repository is as REST services.

The diagram below expresses this –

This allows having multiple consumers and producers of documents independent of the Portal.

The Gawati portal is also a native eXist application, but we don't connect to the data natively to keep our system flexible. That way we can deploy the data independent of the portal or any other application, and we can do things like failover for just the data server component.

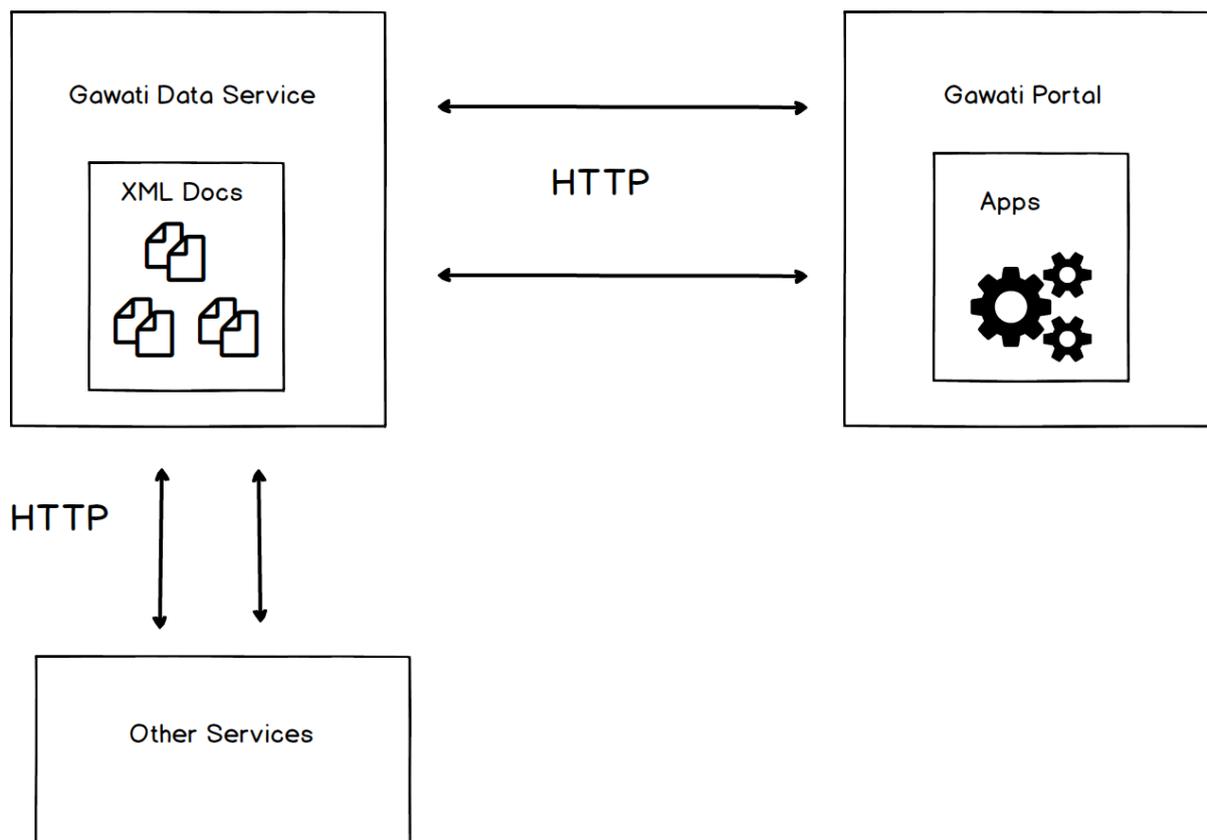


Fig. 1: Gawati Data Server

## 4.3 How Trust & Security Works in Gawati

### Table of Contents

- *Introduction*
- *Realms, Users, Applications*
- *Login & Authentication*

### 4.3.1 Introduction

This document focuses on how trust and security works in Gawati, what are the components we use, how they interact with each other, and what are the specific configuration choices we have made in Gawati. We cover aspects governing both authentication and authorization in Gawati.

In Gawati we use a third-party open source server called [Key Cloak](#). KeyCloak uses JWT (JSON Web Tokens) to implement security. For a quick introduction to JWT see [Understanding JSON Web Tokens](#) .

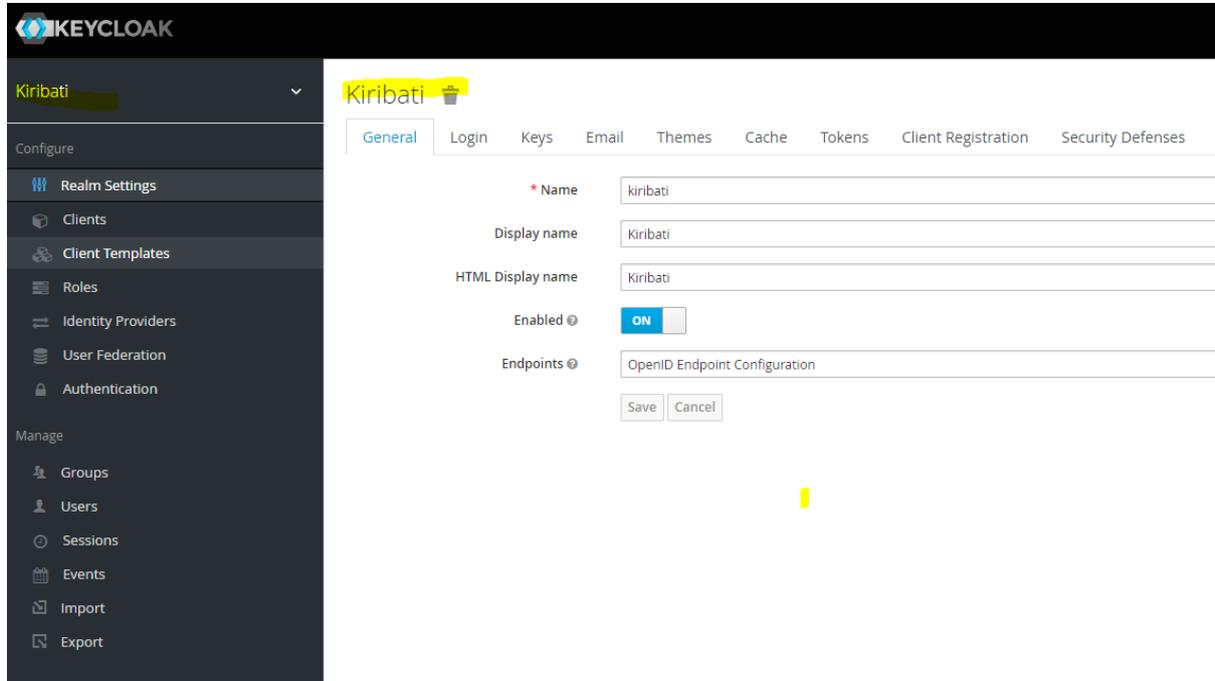
---

**Note:** To install KeyCloak for Gawati use see the `../development/authentication` .

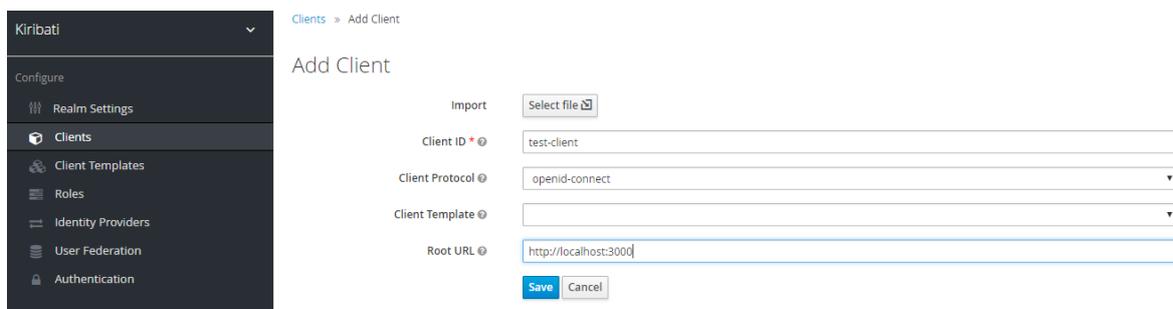
---

### 4.3.2 Realms, Users, Applications

Users in KeyCloak are defined within authentication realms. Applications wanting to authenticate with a Realm, need to be registered and configured on the Realm, a registered Application in KeyCloak is called a `Client`. Applications can define specific roles which can be assigned to users on the Realm. In the image below we have added a realm called `kiribati` which will hold all the registered users.



Client applications wishing to connect need to register, by `Add Client`, see image below, where we add a `test-client`. The `root url` is the URL of the application authenticating with the KeyCloak instance.



The `test-client` can be edited. For use in Gawati we need to change some defaults. Most importantly, we need to change the default `Access Type` from `public` to `confidential`. We do this because, the `public` method does not provide access to authentication tokens; `confidential` also requires passing a `secret` to KeyCloak when initiating any kind of request, and is more secure; finally, in `confidential` mode, tokens can be introspected on the server side (this is possible when `Service Accounts Enabled` is switched on).

Enable `Service Accounts`, `Authorization` and `Direct Access Grants`. For application use you may also set `Direct Access Grants` to off, but some admin tools may not work with the client. Set the rest of the URL parameters as per your application. When `Access Type` is changed to `confidential`, a new `credentials` tab is added which has the `secret` which has to be passed by clients making requests to KeyCloak.

Configure

- Realm Settings
- Clients**
- Client Templates
- Roles
- Identity Providers
- User Federation
- Authentication

Manage

- Groups
- Users
- Sessions
- Events
- Import
- Export

### Test-client

Settings Roles Mappers Scope Revocation Sessions Offline Access Installation Permissions

Client ID

Name

Description

Enabled

Consent Required

Client Protocol

Client Template

Access Type

Standard Flow Enabled

Implicit Flow Enabled

Direct Access Grants Enabled

Service Accounts Enabled

Authorization Enabled

Root URL

\* Valid Redirect URIs

Base URL

Admin URL

Web Origins

Clients » test-client

### Test-client

Settings **Credentials** Roles Mappers Scope Authorization Revocation Sessions Offline Access Clusteri

Client Authenticator

Secret

Registration access token



Clients > test-client

Test-client

Settings Credentials Roles Mappers Scope Authorization Revocation Sessions Offline Access Clustering **Installation** Service Account Roles Permissions

Format Option Keycloak OIDC JSON

Download

```
{
  "realm": "kiribati",
  "auth-server-url": "http://auth.gawati.local/auth",
  "ssl-required": "external",
  "resource": "test-client",
  "credentials": {
    "secret": "4b5021d2-abce-4e1d-bf80-4ae791dcb09c"
  },
  "use-resource-role-mappings": true,
  "confidential-port": 0,
  "policy-enforcer": {}
}
```

(continued from previous page)

3  
4  
5  
6  
7  
8  
9

```

"expires_in": 60,
"refresh_expires_in": 1800,
"refresh_token":
↪ eyJhbGciOiJSUzI1NiIsInR5cCI6IkpXLTQ5MTc5MTcxNzksIm5iOiI6MCw
↪ 189VEcNhrpQ70SnZ711dQ92mVpNhqpX_KGOWcRQxykODv9cVbfckMAbQ3JIsvkq7GazuJuUh3VL-
↪ f2ib7dwPoFsIBNGcQXwdG8Wn-EjIvKDAAZZ0xjIBGJjRr5P24u_
↪ U3JPTWM1rWT3XAYQGdk4PLPJlI5RUVJRPn8Gly9fOKbPbd3dwAB6grKBdy9jW3mbAtF6bCEz1zByuK3n17kj5jGz1xsZt21
↪ 8r1Mo4Gy2Zutt_Oq7M75cOEtMDBBmsBBREELBJ0K5tz47L8IspQ_e92bh-
↪ ZSkxRjZdj2SylCLnhB5CI3z2ifpl0rjLp6RFucj8HJCAEI-FA",
"token_type": "bearer",
"not-before-policy": 0,
"session_state": "699e953f-15e3-4ddc-bd5e-f38ca40f0ce9"
}
```

The `access_token` has only a short life-span, and needs to be periodically updated by the Application, by making a refresh token API call to Keycloak, to indicate that the user is still active. The `access_token` contains all the information associated with the Application, that we had configured on the Keycloak client earlier. The client can make authenticated and unauthenticated API calls to a server side API, but for authenticated Server Side apis, the `access_token` is passed, and validated at the server end. Validation happens at the server end, by passing the token back to the Keycloak server, to an introspecting API, which returns a status of `active = false` if the token is invalidated, or if it is valid returns the full decoded content of the token:

1  
2  
3  
4  
5  
6  
7  
8  
9  
10  
11  
12  
13  
14  
15  
16  
17

```
{
  "jti": "0a9c614a-1120-438b-8aa3-db69792df89a",
  "exp": 1522085986,
  "nbf": 0,
  "iat": 1522085686,
  "iss": "http://auth.gawati.local/auth/realms/kiribati",
  "aud": "test-client",
  "sub": "fa3dceab-dd19-4bd2-9f2a-e4aeb9557bfe",
  "typ": "Bearer",
  "azp": "test-client",
  "nonce": "d0de0910-4184-4497-8510-73294ff03cbb",
  "auth_time": 1522085418,
  "session_state": "a448a83c-79c1-49eb-bc86-d6b672a6973f",
  "name": "test kumar",
  "given_name": "test",
  "family_name": "kumar",
  "preferred_username": "test",
}
```

(continues on next page)

(continued from previous page)

```
18  "email": "test@blah.in",
19  "acr": "0",
20  "allowed-origins": [
21    "http://localhost:3000"
22  ],
23  "realm_access": {
24    "roles": [
25      "uma_authorization"
26    ]
27  },
28  "resource_access": {
29    "test-client": {
30      "roles": [
31        "test-client.Admin"
32      ]
33    },
34    "gawati-client": {
35      "roles": [
36        "client.Editor",
37        "client.Admin"
38      ]
39    },
40    "account": {
41      "roles": [
42        "manage-account",
43        "manage-account-links",
44        "view-profile"
45      ]
46    }
47  },
48  "client_id": "test-client",
49  "username": "test",
50  "active": true
51 }
```

The decoded information(for e.g. the roles) can be used by the server API to apply it to business logic - for instance, filter queried data based on the role of the user, and send it back to the client in the API response. Effectively there is an overhead of querying KeyCloak to introspect the token for every request made to the the server.

---

**Note:** There is an alternative approach which avoids the overhead of the request, by making use of signed JWT, where the signature passed by the client is validated by the server API using standard JWT libraries, without querying KeyCloak. We have not implemented this in Gawati yet.

---



## 5.1 Introduction for Developers

### 5.1.1 Start Here

*Developing Gawati*

### 5.1.2 GitHub

All our code is on Github ([Gawati on Github](#)), under Open Source licenses. Our Github Project folder is folder: [Gawati Project on Github](#)

### 5.1.3 Mailing Lists

- [Google Group - General](#)
- [Google Group - Development](#)

### 5.1.4 Contributor Agreement

We appreciate your consideration to contribute to our project. Please contact us via our [Google Group - Development](#).

To avoid potential legal problems over the course of time, we want to prepare to avoid problems that some open source projects have faced in the past when the need arises to [change the license](#) of the project. This contributor agreement is very light, and shall simply ensure that the project can continue using all of its code base, should there be a need to relicense.

Like our source code license ([AGPLv3](#)), we use existing community standards ([Harmony Agreements](#)). As we want to retain the wording 1:1, we chose to not remove what becomes a duplicate reference to this page in the document in our case, our apologies for the confusion this may cause.

**As per Harmony Agreements we provide a:**

- Contributor Agreement download for individual contributors, and a separate

- Contributor Agreement download for entities

## 5.2 Developing Gawati

### Table of Contents

- *Introduction*
- *Getting Started*
- *Development Workflow*
- *Building code from Github*
- *Customizing Gawati*

### 5.2.1 Introduction

Gawati is made of different components, you don't need to install the full set to begin developing on the Gawati platform. This section provides an outline of how you can use and individual components.

---

**Note:** If you only wish to install and test the system, See *Setup*.

---

### 5.2.2 Getting Started

You will need to setup the basics first.

Prepare your development environment installing and running Gawati as local services on your development system (this document provides you with additional insight). Alternatively, use a prebuilt VM and Gawati installer to run Gawati as a virtual machine (this document gets you a standardised environment fast).

#### Gawati as a local installation

If you haven't yet, please review the Gawati Architecture page: *Gawati Architecture*.

The below instructions are not specific to an operating system, the components will run on different operating systems.

---

**Note:**

#### Pre-requisites (runtime):

1. **JDK 1.8 LTS** - On Linux operating systems you can install [OpenJDK8](#); For Windows and for Mac OSX and [Using OS X Homebrew](#)
2. **NodeJS LTS 8.11.x** - See [NodeJS LTS 8.11.x](#). Alternatively you can install NVM (for Linux , Windows ) which lets you easily install parallel versions of NodeJS.
3. **eXist-db 4.3.0**: Download and install eXist-db 4.3.0, see [eXist-db](#) Remember to note down the admin password of the eXist-db installation, you will need that later. If you are installing eXist-db on Mac OS X, install it within the User folder, installing it in `/Applications` causes problems sometimes as the permissions required for eXist-db to write to the file system are for a super user.

4. **Apache HTTPD 2.4.x:** Install Apache, on Cent OS, Ubuntu and OS X this will likely be installed by default, on Windows you will have to download and install, see [Apache for Windows](#); enable `mod_alias`, `mod_rewrite`, `mod_proxy`, `mod_proxy_http` and enable htaccess.
5. **KeyCloak 3.4.1:** Authentication server, provides single-sign-on, see [./authentication](#)
6. **RabbitMQ 3.7.6:** Message Queue server, used by the content sync engine that moves data between the *Gawati Editor System* and the *Gawati Portal System* see [installing rabbitmq server, version 3.7.6](#) you will need to install Erlang 20.3 before that, see [installing erlang](#).
7. **MongoDB 3.6.5 Community Edition:** Used by the Gawati User Profiles system, see [Download MongoDB](#) , [Install MongoDB](#)
8. **Jetty 9.4.6.v20170531:** Used by *Gawati Portal System* (NOTE: required, only if you are saving legal documents in XML format), [Download](#) , [Installing](#).
9. **Python 3.6:** Used by *Gawati Editor System* for automatically tagging text (the sub-service is used `gawati-tagit`).
10. **Python 2.7:** Used by *Gawati Editor System* for extracting text from PDF files (the sub-service is used `pdf2xml-service`).

**Pre-requisites (development):**

1. *Ant*: Download and [install Ant](#)
2. *Visual Studio Code*: This is if you want play around with gawati code. Download, and setup Visual Studio Code (there are versions for Windows, OS X and Linux) for development, see [VS Code Setup](#)

## Table of Contents

- *Installing Gawati Portal*
  - *Installing Gawati Data*
    - \* *Load Sample Data*
    - \* *Download the data sets*
    - \* *Setup the PDF data set*
    - \* *Setup the XML data set*
    - \* *Finally Rebuild the Database Index*
    - \* *Apache Config*
    - \* *Development*
    - \* *Accessing eXist-db via SSH tunnelling*
  - *Installing Gawati Portal UI*
    - \* *Development and Production mode*
  - *Installing Gawati Portal FE*
    - \* *Apache Config*
    - \* *Environment Variables*
  - *Configuring KeyCloak Auth*
  - *Starting up Services*
  - *Installing Gawati Viewer Service*
    - \* *Running the REST service*

- *Installing Gawati Profiles*
  - *Setting up profiles-fe*
  - *Installing Gawati Profiles UI*
- *Installing Gawati Data Editor*
  - *Setting up the Editor UI*
  - *Setting Up the Editor FE (Server Component)*
  - *Installing Gawati Editor BE (Data services Component)*
  - *Configuring KeyCloak Auth*
  - *Run Gawati Editor*

You can either build the source from github for each component, or you can install a released version of a component. For getting familiar with the system we recommend starting by installing a released version.

---

### Note: Current Version

- Gawati 1.0.18 [download link](#) , released on: 05 July 2018
  - portal-ui : 2.0.33
  - portal-fe : 1.0.15
  - gawati-data : 1.19
  - gawati-editor-ui : 1.0.14
  - gawati-editor-fe : 1.0.11
  - gawati-client-data : 1.11
  - gawati-editor-qprocessor : 1.0.0
  - gawati-portal-publisher : 1.0.0
  - gawati-portal-qprocessor : 1.0.0
  - gawati-profiles-ui : 1.0.2
  - gawati-profiles-fe : 1.0.1

Older releases, see: `./version-compat`

---

## Installing Gawati Portal

Gawati Portal provides access to all legal and legislative data in the system. See *Gawati Portal System* for an architecture overview.

**IMPORTANT:** In Gawati Component interaction is purely via REST services, we use Apache HTTP as a reverse proxy to bind all the services together under one domain and user interface for the user. You may want to read up and get an idea of the Apache configuration before starting the installation. `./dev-and-prod-testing`

## Installing Gawati Data

---

### Note: What is gawati-data about ?:

The *Gawati Data* package provides REST based services to access the XML documents which are stored in the XML database.

---

For development environments, you should clone the project from Git, and build the package:

```
1 git clone https://github.com/gawati/gawati-data.git
2 ant xar
```

Install the `gawati-data` XAR file into eXist using the eXist package manager in the eXist-db admin dashboard to manually select and install the package (see [Installing Packages in eXist-db](#)).

## Load Sample Data

---

**Note:** The sample data is currently at version 1.14

---

To understand better how `gawati` works, we provide you with sample data, which can be loaded into the system and tested. Sample data is provided in two specific parts:

- Xml Documents - which get loaded into the XML database
- PDF and other binary Documents - which are referred to by the XML documents, but served from the *file system*

We serve PDF and other binary documents from the filesystem to ensure optimal performance.

## Download the data sets

Download the XML data set, which is in 2 parts: [XML Data set](#) + [Full Text Data set](#) (the full text data set is the full text extraction of the PDFs) and the corresponding [PDF Data set](#)

## Setup the PDF data set

To setup the PDF data-set, you just need to extract the files into a folder, e.g if you extract the PDF files into `/home/data/akn_pdf`, and add a Apache configuration to serve the folder contents (See [conf-binary](#))

## Setup the XML data set

To setup the XML data-set, extract the archives into separate folders (e.g. `/home/data/akn_xml/akn` and `/home/data/akn_xml/akn_ft`). On Linux and MacOS you can run the following command to get the data input password:

```
1 <path_to_exist>/bin/client.sh -ouri=xmldb:exist://localhost:8080/exist/xmlrpc -u_
↵admin -P <exist_admin_password> -x "data(doc('/db/apps/gawati-data/_auth/_pw.xml
↵')/users/user[@name = 'gawatidata']/@pw) "
```

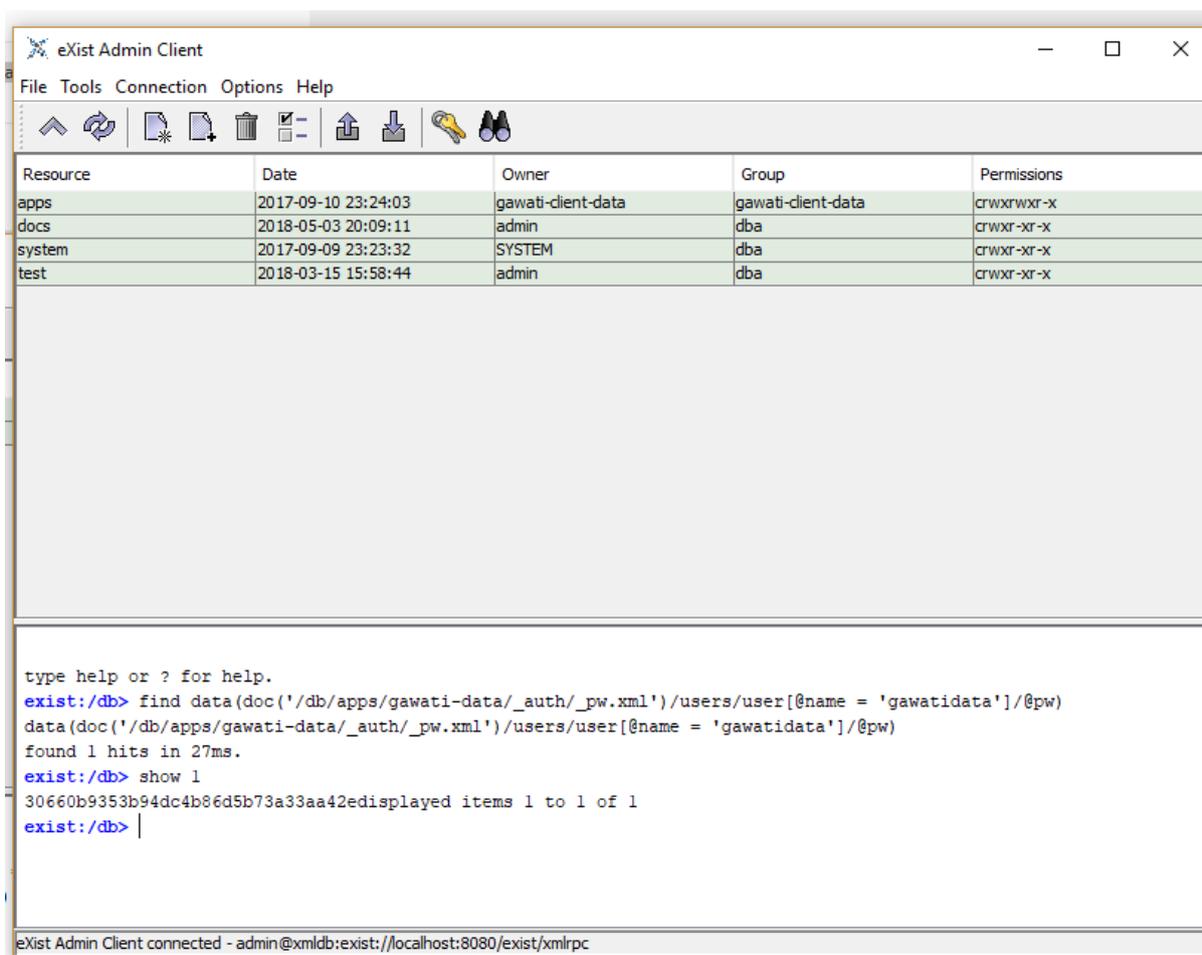
Where `<path_to_exist>` is the path to the eXist-db installation, and `<exist_admin_password>` is the eXist-db admin password. If you installed eXist on a different port change that in the `-ouri` setting.

On Windows do the following; Start the eXist-db Client(`<path_to_exist>/bin/client.bat`). In the command window of the eXist-db client run the following commands:

```
1 find data(doc('/db/apps/gawati-data/_auth/_pw.xml')/users/user[@name = 'gawatidata
↵']/@pw)
2 show 1
```

Copy the output password hash as shown below.

Now upload the data using the following command run from the eXist-db folder:



```

1 ./bin/client.sh -u gawatidata -P <copied_password_hash> -d -m /db/docs/gawati-data_
↪-p /home/data/akn_xml_docs_sample
2 ./bin/client.sh -u gawatidata -P <copied_password_hash> -d -m /db/docs/gawati-data_
↪-p /home/data/akn_xml_ft_sample

```

On Windows you will run it as :samp:.\bin\client.bat instead:

```

1 .\bin\client.bat -u gawatidata -P <copied_password_hash> -d -m /db/docs/gawati-
↪data -p d:\data\akn_xml_docs_sample
2 .\bin\client.bat -u gawatidata -P <copied_password_hash> -d -m /db/docs/gawati-
↪data -p d:\data\akn_xml_ft_sample

```

#### Note:

1. the current folder structure expected within /db/docs/gawati-data is /db/docs/gawati-data/akn for the XML files and /db/docs/gawati-data/akn\_ft for the full text files.
2. if you get a password failure, log in to eXist-db as admin, and reset the password for gwdata user manually, and then use that password.

### Finally Rebuild the Database Index

```

1 $curl http://localhost:8080/exist/apps/gawati-data/post-data-load.xml
2 <success>Build Sort index</success>

```

### Apache Config

There are Apache HTTP configs required for both serving XML and PDF documents. See conf-gawati-data and conf-binary

### Development

We recommend using Oxygen XML for developing on eXist-db. VSCode can also be used (see *Using VS Code with eXist DB*).

### Accessing eXist-db via SSH tunnelling

If eXist-db is installed in a remote server, by default the server starts on port 8080 and listens only to localhost. To access the web-based dashboard from a remote computer, you need to use ssh tunneling. For example, if your remote server is on the I.P. Address *101.102.103.104*, and eXist-db is on port *8080*, running the following command, will give you access to the eXist-db dashboard on *http://localhost:9999* :

```

1 ssh -vv -i <path to private key> -p 22 -L 9999:127.0.0.1:8080 server_user@101.102.
↪103.104

```

### Installing Gawati Portal UI

Extract the contents of the zip file onto a directory served by Apache.

But, if installing for development, clone from git and build:

```
1 git clone https://github.com/gawati/gawati-portal-ui.git
2 npm install
```

And add the corresponding Apache Server configuration entry (See `conf-portal-ui`).

The React app in the <https://gitlab.com/bungenicom/gawati/gawati-portal-ui> repository can be used to run different clients. The configuration and set up scripts for the presently available clients: *ecowas* and *gawati* are available in separate git repositories.

```
1 git clone https://gitlab.com/bungenicom/gawati/portal-ui-config
2 git clone https://gitlab.com/bungenicom/ecowas/portal-ui-config
```

### Switching clients

In order to switch clients during development:

- Checkout to the `<client>/portal-ui-config` repository and pull latest.
- **Check proxy setting in**
  - `package.json`
  - `portal-ui-config/proxy.js`.
- Change following values in `.env` and `.env.development` files accordingly:

```
REACT_APP_THEME=default
REACT_APP_CLIENT=gawati
REACT_APP_CONFIG_PATH=../../gawati-portal-config
```

- Restart portal

---

**Note:** `REACT_APP_CONFIG_PATH` points to the client config repository. The path is relative to the `scripts` folder in `gawati-portal-ui` repository.

---

---

**Note:** A theme can only be associated with a specific client. A client may have a choice of multiple themes.

---

### Development and Production mode

See our detailed guide on setting up your environment for production and development mode testing `./dev-and-prod-testing`.

For setting up Authentication, click here: [Authentication](#)

### Installing Gawati Portal FE

Extract the contents of the zip file into any directory.

For development environments, clone from git and install it:

```
1 git clone https://github.com/gawati/gawati-portal-fe.git
2 npm install
```

The Gawati Portal has two runnable components, the portal http server which provides access to REST services, and a cron component that runs scheduled tasks periodically.

## Apache Config

See conf-portal-server.

## Environment Variables

The server can be customized with various environment variables which can be specified as prefixes to the service startup.

- `WITH_CRON` - setting `WITH_CRON=1` starts the server with the cron, so there is no separate process for the cron. *This is not recommended for production use.*
- `WITH_CLIENT` - setting `WITH_CLIENT=1`, the server provides the portal-ui client on the `/v2` virtual directory (instead of Apache doing it). The client is expected to be in the `client/build` sub-directory.
- `HOST` - allows setting the host name or address which the server binds to, default is `127.0.0.1`.
- `PORT` - allows setting the port on which the server listens to, default is `9001`.
- `API_HOST` - allows setting the host address to the `gawati-data` server, default is `localhost`
- `API_PORT` - allows setting the port number to the `gawati-data` server, default is `8080`

## Configuring KeyCloak Auth

1. Follow the installation steps from [Installing Keycloak](#).
2. Within the `auth.gawati.local` realm, navigate to the `Clients` tab. Click on `gawati-portal-ui`. Set the other parameters as shown below. In this case we have set the root url, valid url etc to `http://localhost:3000` which is the dev mode host and port for Gawati Editor UI. If you are deploying on a domain e.g. `http://www.domain.org` you can set it to that domain. Note the Redirect URLs can be set to multiple urls, this is because, the profiles services typically runs on a different url base .e.g. a different domain than the portal, so the authentication headers need to support such redirects, and unless each of the redirect domains ( for the portal-ui and for the profiles system) are set here, then the authenticated redirect will fail. You can set `Web Origins` to `+` which tells KeyCloak to set valids CORS headers `Redirect Urls`.

The screenshot shows the Keycloak Admin Console interface. The left sidebar is dark with the Keycloak logo at the top. The main content area is light gray and shows the configuration for the 'gawati-portal-ui' client. The 'Settings' tab is selected, and the 'Enabled' toggle is turned on. The 'Client ID' is 'gawati-portal-ui', and the 'Access Type' is 'confidential'. The 'Standard Flow Enabled' toggle is also turned on.

Field	Value
Client ID	gawati-portal-ui
Name	
Description	
Enabled	ON
Consent Required	OFF
Client Protocol	openid-connect
Client Template	
Access Type	confidential
Standard Flow Enabled	ON

Authorization Enabled ?  ON

Root URL ?

\* Valid Redirect URIs ?

<input type="text" value="http://localhost:3000/*"/>	-
<input type="text" value="http://localhost:9004/*"/>	-
<input type="text" value="http://localhost:3001/*"/>	-
	+

Base URL ?

Admin URL ?

Web Origins ?

<input type="text" value="+"/>	-
	+

3. Within the client config, switch to the `Credentials` tab and regenerate the secret.

Clients > gawati-portal-ui

Gawati-portal-ui

Settings **Credentials** Roles Mappers ? Scope ? Authorization Revocation Sessions ? Offline Access ?

Client Authenticator ?

Secret

Registration access token ?

4. Switch to the `Installation` tab in the client section, and choose the format as `KeyCloak OIDC JSON`. Download the json file.
5. Open the downloaded json file using your preferred text editor. Copy the variables `auth-server-url` to `url` and `resource` to `clientId`. It should look similar to the json shown below.

```

1  {
2  "realm": "auth.gawati.local",
3  "auth-server-url": "http://localhost:11080/auth",
4  "url": "http://localhost:11080/auth",
5  "ssl-required": "external",
6  "resource": "gawati-portal-ui",
7  "clientId": "gawati-portal-ui",
8  "credentials": {
9    "secret": "b344caaa-7341-479f-81b7-9d47aa3128dc"
10 }
11 "use-resource-role-mappings": true,
12 "confidential-port": 0,
13 "policy-enforcer": {}
14 }

```

6. Copy the downloaded `keycloak.json` contents into the `gawati-portal-fe/configs/auth.json` file on the portal-fe installation (see *Installing Gawati Portal FE*).

7. Finally, login as admin into KeyCloak and create some users. You can create a test users like *portal-admin*, *portaleditor*, *portaluser* and associate them with the groups *portalui.Admins*, *portalui.Eeditors* and *portalui.Eeditors* .

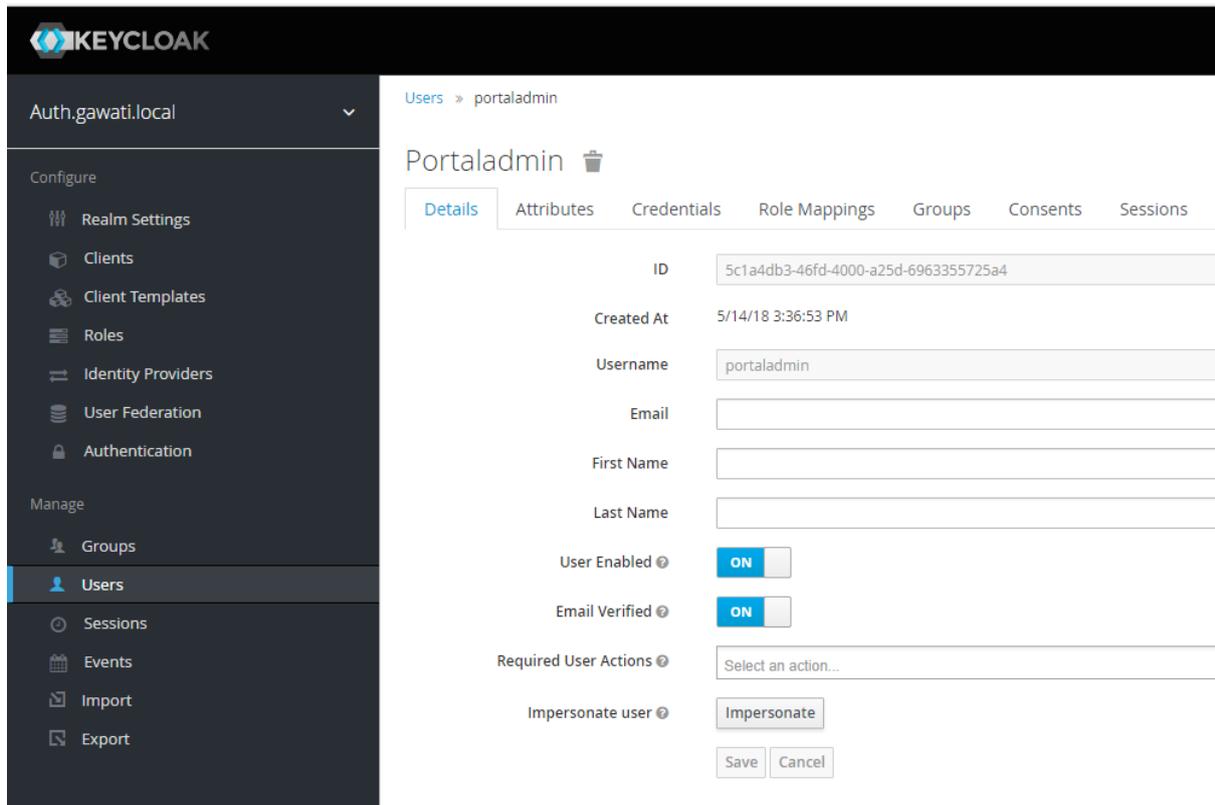


Fig. 1: Above: a user called `portaladmin` has been added.

## Starting up Services

All the primary components and services need to be started in a specific order because of interconnected dependencies.

The recommended order is as follows:

- Base Services
  1. KeyCloak
  2. eXist-db
  3. MongoDB
  4. RabbitMQ
- Component Services
  1. *Installing Gawati Portal FE*
  2. *Setting up profiles-fe*
  3. *Installing Gawati Profiles UI*
  4. *Installing Gawati Viewer Service*
  5. *Installing Gawati Portal UI*

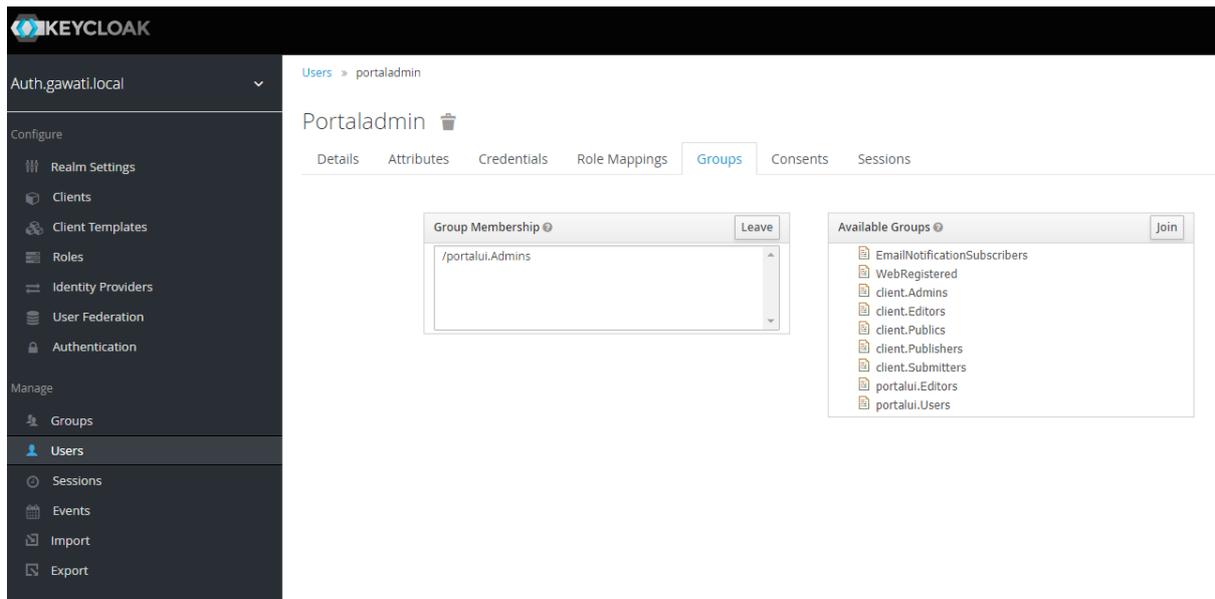


Fig. 2: Above: the user has been added to the `portalui.Admins` group to give it the `portalui.Admin` role.

## Installing Gawati Viewer Service

Extract the contents of the zip file into any directory.

For development environments, clone from git and install it:

```
1 git clone https://github.com/gawati/gawati-viewer-service.git
2 npm install
```

The Gawati Viewer services are invoked by the `gawati-viewer` npm package. The service can convert XML/DOCX to HTML

## Running the REST service

Run the following in the extracted folder to setup the server:

Assuming you extracted the portal server into : `/home/web/gawati-viewer-service`, from that folder, run :

```
1 node ./bin/www
```

## Installing Gawati Profiles

Gawati Profiles allows authenticated users in system to have a profile with their personal information. The Profiles system supports other functionality in the system, like allowing logged in users to save their searches. The profiles system is made up of 3 different components:

- MongoDB (as mentioned earlier as a pre-requisite)
- `profiles-ui` - front-end component
- `profiles-fe` - back-end component

## Setting up profiles-fe

Extract the contents of the zip file into any directory.

For development environments, clone from git and install it:

```
1 git clone https://github.com/gawati/gawati-profiles-fe.git
2 npm install
```

Run the following in the app folder to setup the server:

```
1 npm install
```

From that folder, run... :

```
1 npm start
```

...to start up the web-service. By default it starts on PORT 9003. You can change that by running it as:

```
1 PORT=11003 npm start
```

## Installing Gawati Profiles UI

Extract the contents of the zip file onto a directory served by Apache.

But, if installing for development, clone from git and build:

```
1 git clone https://github.com/gawati/gawati-profiles-ui.git
2 npm install
```

And add the corresponding Apache Server configuration entry (See conf-profiles-ui).

## Installing Gawati Data Editor

Gawati Data Editor (or Gawati Editor in short), is a tool that allows inputting managing documents in the portal. Gawati is a suite of distributed applications, and the same model applies here for data entry. The *Gawati Portal System* has been conceived has a public facing system to access and search for data. The Gawati Editor is a back-office systeme that allows managing the process of entering data and publishing it online.

Gawati Editor can be used independent of the *Gawati Portal System*, as it has its own working-data store and workflow, and information is published onto the *Gawati Portal System* via a asynchronous message queue.

The Gawati Editor is composed of different components: Editor UI, Editor Server component, Editor Data services(an eXist-db component), and authentication integration component.

See *Gawati Editor System* for an architecture overview.

## Setting up the Editor UI

To install the Editor UI Component in development environments:

1. Clone <https://github.com/gawati/gawati-editor-ui.git>
2. Install packages

```
1 npm install
```

In case of error during the above step, please refer to <https://stackoverflow.com/a/39648550> on instructions on how to resolve the issue

### Setting Up the Editor FE (Server Component)

To install the Editor Server Component in development environments:

1. Clone <https://github.com/gawati/gawati-editor-fe.git>
2. Install packages

```
1 npm install
```

### Installing Gawati Editor BE (Data services Component)

1. Download sample data from here: [Client Sample data \(XML\)](#), [Client Sample data \(PDF\)](#)
2. Clone <https://github.com/gawati/gawati-client-data.git>
3. Build to get the package.

```
1 cd gawati-client-data
2 ant xar
```

The above generates *gawati-client-data-1.x.xar* package in the `build` folder. Install it using the Package Manager in the eXist-db admin dashboard to manually select and install the package (see [Installing Packages in eXist-db](#)). Alternatively, here is a video that shows how to install a package in eXist-db:

4. Extract and load the [Client Sample data \(XML\)](#). In eXist's dashboard -> Collections, create the path `/db/docs/gawati-client-data`.

Now upload the data using the following command run from the eXist-db folder:

```
1 ./bin/client.sh -u gawati-client-data -P <gawati-client-data_password>
  ↪ -d -m /db/docs/gawati-client-data -p <path_to_extracted_data>/gawati-
  ↪ client-data
```

or on windows:

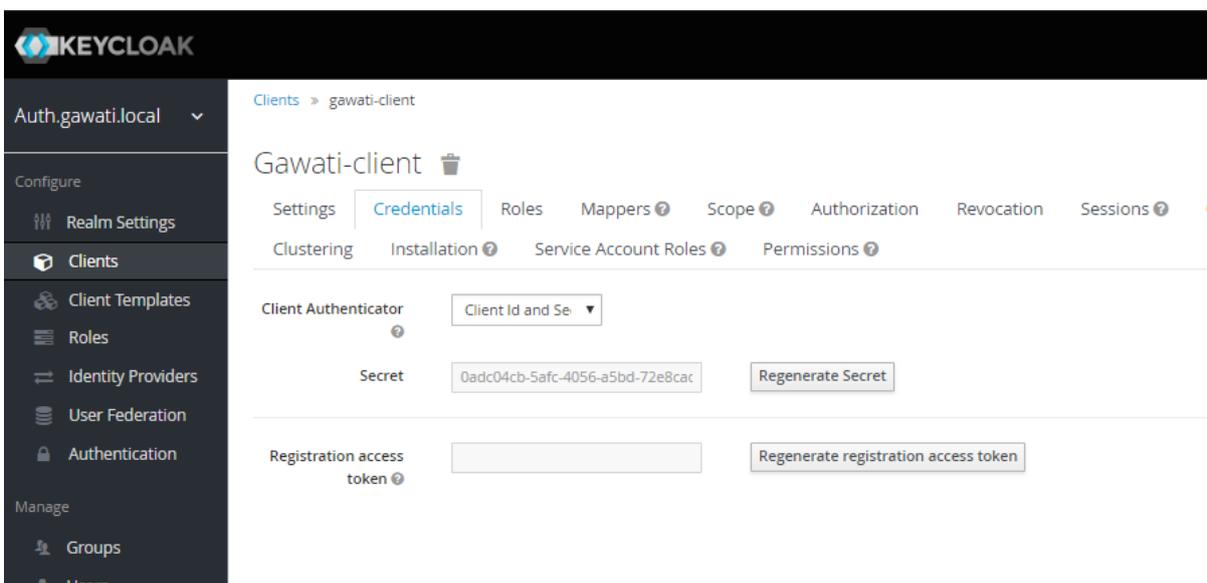
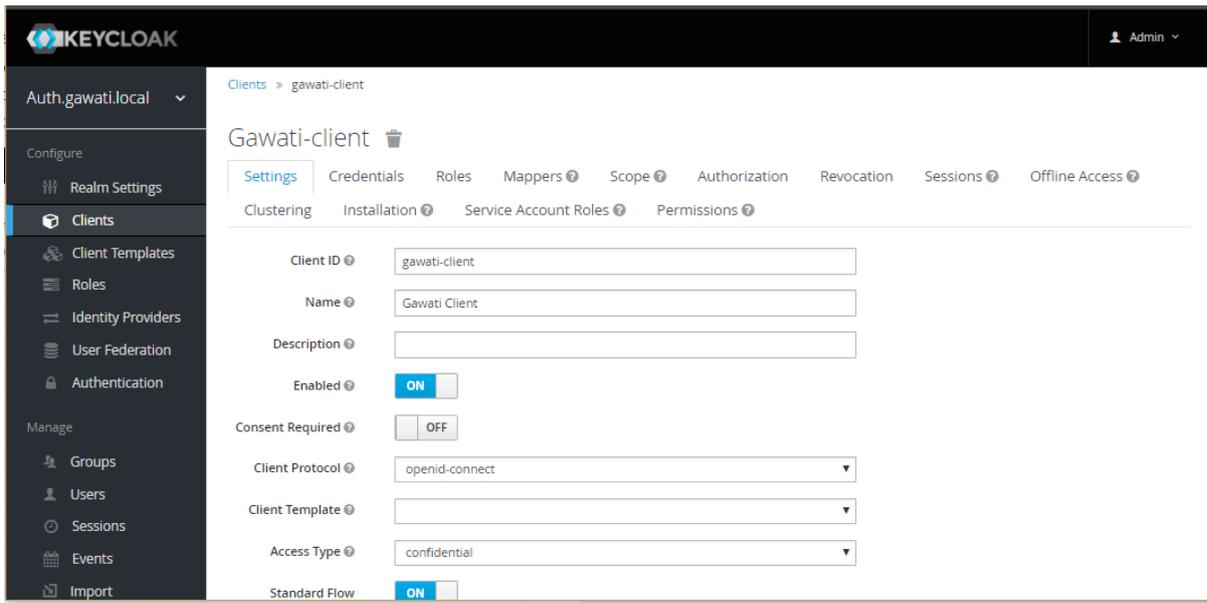
```
1 .\bin\client.bat -u gawati-client-data -P <gawati-client-data_password>
  ↪ -d -m /db/docs/gawati-client-data -p <path_to_extracted_data>
  ↪ \gawati-client-data
```

the user here is `gawati-client-data` which is the user with permissions over the `/db/docs/gawati-client-data` collection where we are storing the xml documents. The password for this user is generated during installation and stored in the `/db/apps/gawati-client-data/_auth/_pw.xml` file. The same instructions are shown in the video below.

1. Make the necessary Apache conf entries. See `conf-client`.

### Configuring KeyCloak Auth

1. Follow the installation steps from [Installing Keycloak](#).
2. Within the `auth.gawati.local` realm, navigate to the `Clients` tab. Click on `gawati-client`. Set the other parameters as shown below. In this case we have set the root url, valid url etc to <http://localhost:3000> which is the dev mode host and port for Gawati Editor UI. If you are deploying on a domain e.g. <http://www.domain.org> you can set it to that domain.
3. Within the client, switch to the `Credentials` tab and regenerate the secret.



4. Switch to the Installation tab in the client section, and choose the format as KeyCloak OIDC JSON. Download the json file.
5. Open the downloaded json file using your preferred text editor. Copy the variables `auth-server-url` to `url` and `resource` to `clientId`. It should look similar to the json shown below.

```
1 {
2   "realm": "auth.gawati.local",
3   "auth-server-url": "http://localhost:11080/auth",
4   "url": "http://localhost:11080/auth",
5   "ssl-required": "external",
6   "resource": "gawati-client",
7   "clientId": "gawati-client",
8   "credentials": {
9     "secret": "b344caaa-7341-479f-81b7-9d47aa3128dc"
10  },
11  "use-resource-role-mappings": true,
12  "confidential-port": 0,
13  "policy-enforcer": {}
14 }
```

6. Copy the downloaded `keycloak.json` contents into the `gawati-editor-fe/auth.json` file on the editor-fe installation (see *Setting Up the Editor FE (Server Component)*).
7. Finally, login as admin into KeyCloak and create some users. You can create test users like *submitter*, *editor*, *admin* and associate them with the groups *client.Submitters*, *client.Editors* and *client.Admins*.

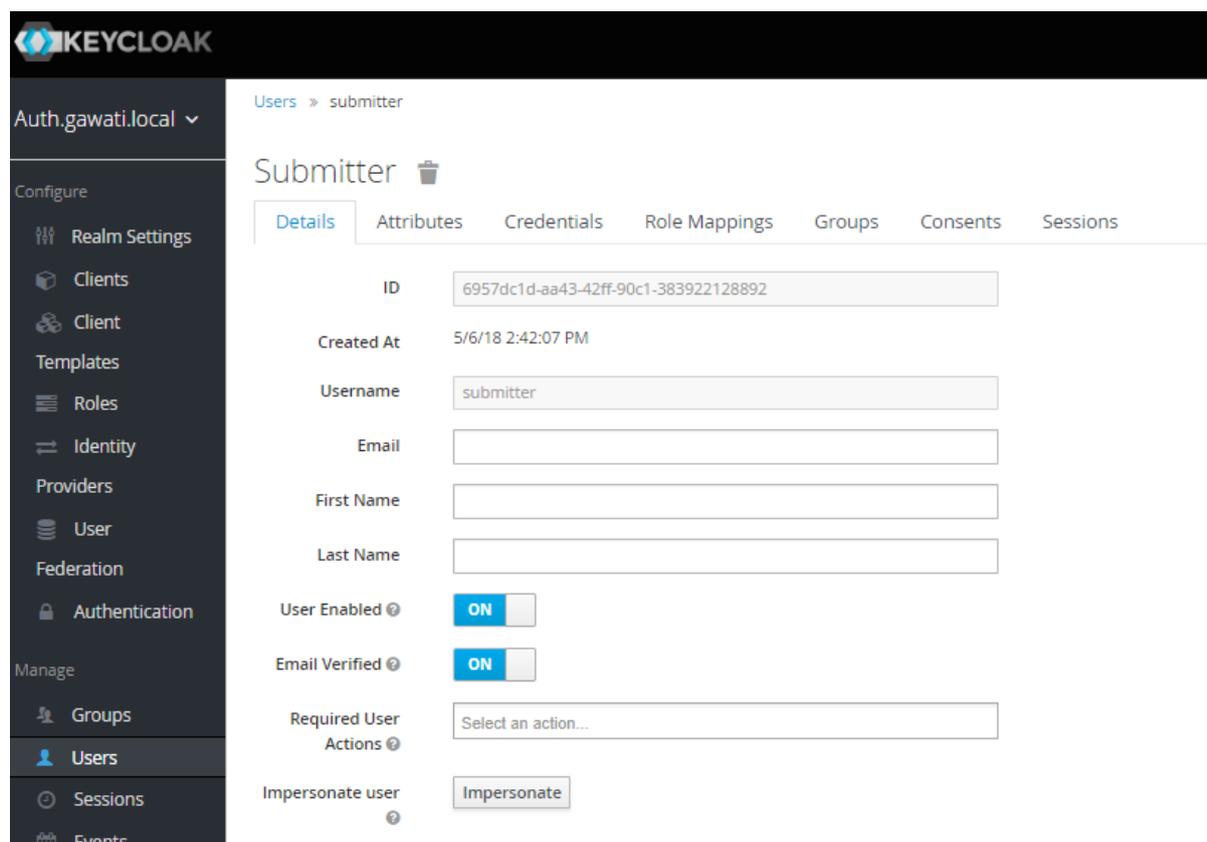


Fig. 3: Above: a user called `submitter` has been added.

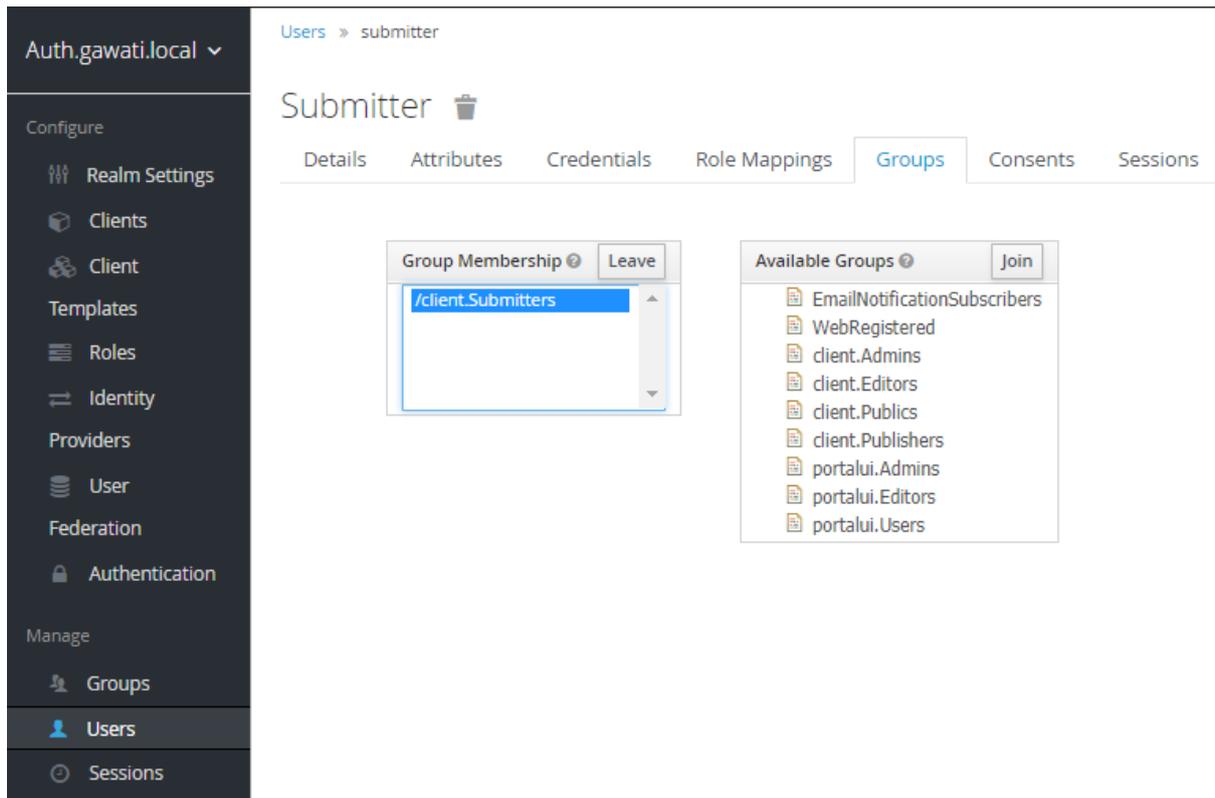


Fig. 4: Above: the user has been added to the client.Submitters group to give it the client.Submitter role.

## Run Gawati Editor

1. Start eXist
2. Start keycloak

```
1 cd keycloak-3.4.3.Final
2 ./bin/standalone.sh
```

3. Start gawati-editor-fe service. Use the dev\_npm\_start scripts to start the service in development mode.

```
1 cd gawati-editor-fe
2 ./dev_npm_start.sh # .\dev_npm_start.bat on windows
```

4. Start gawati-editor-ui

```
1 cd gawati-editor-ui
2 npm start
```

5. Load <http://localhost:3000> in the browser. You should see a login screen. Login with any of the users you created.
6. After logging in, you should be able to see the dashboard with some sample documents.

## Gawati on a server / VM

This will have your development consist of 2 components:

### 5.2. Developing Gawati

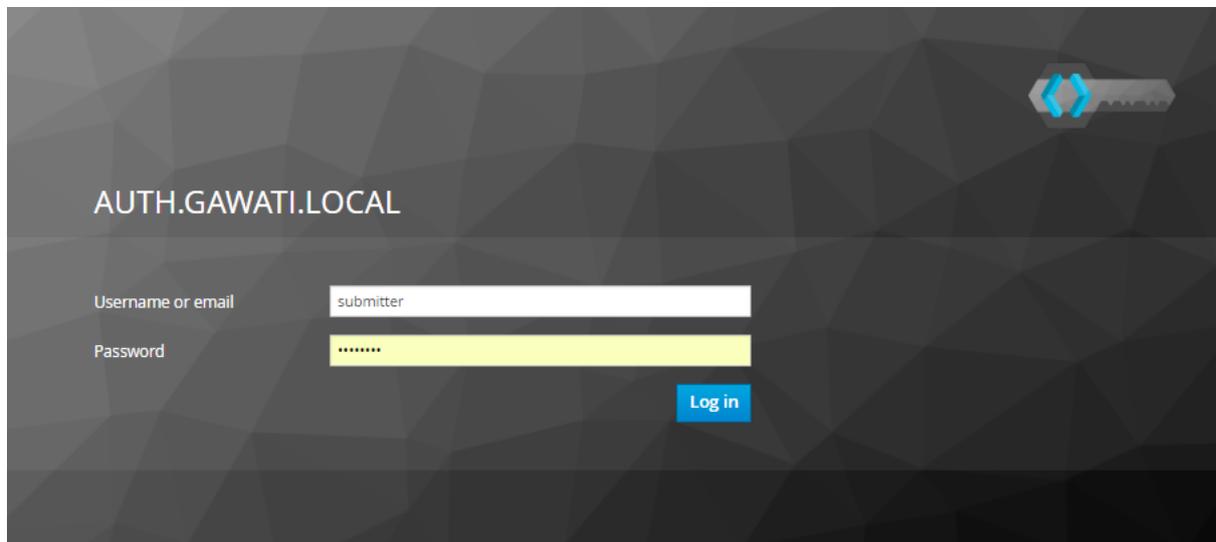
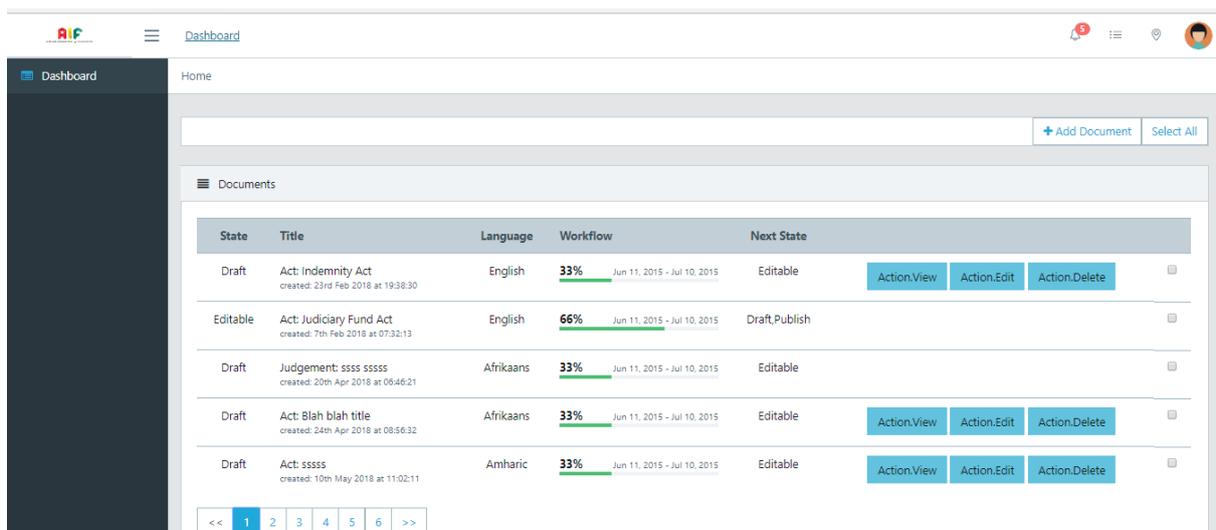


Fig. 5: Above: Login screen for gawati-editor



- Gawati on a CentOS server or virtual machine
- Your desktop machine for development

You will use SSH to securely connect your desktop to the server.

### Table of Contents

- *Using Gawati scripts*
- *Chocolatey installation system*
  - *For MS Windows*
    - \* *Install Chocolatey*
- *Install Gawati server*
  - *Prepare a CentOS minimal server*
    - \* *Gawati Server on local VM*
- *Install your desktop development tools*
  - *Install development applications*
  - *Configure Visual Studio Code*
  - *Map a drive to Gawati server*

## Using Gawati scripts

The installation has been prepared for you in the form of [scripts to download](#) and run. After the script based installation completes, you may skip some steps below and continue at [Configure Visual Studio Code](#) . The individual steps described below are for reference and troubleshooting.

## Chocolatey installation system

### For MS Windows

#### Install Chocolatey

Install [Chocolatey](#) from their website or open administrative cmd.exe and execute

```
@"%SystemRoot%\System32\WindowsPowerShell\v1.0\powershell.exe" -NoProfile -
↵InputFormat None -ExecutionPolicy Bypass -Command "iex ((New-Object System.Net.
↵WebClient).DownloadString('https://chocolatey.org/install.ps1'))" && SET "PATH=
↵%PATH%;%ALLUSERSPROFILE%\chocolatey\bin"
```

Close and reopen administrative cmd.exe The [Chocolatey](#) installation commands below, are to be executed in this administrative cmd.exe

#### Install Gawati server

#### Prepare a CentOS minimal server

You can do this as a VM on your local machine, or use a remote installation. Below we will run on a preconfigured VM image that you can download.

### Gawati Server on local VM

We recommend [Virtualbox](#). Install using the command

```
choco install kitty 7zip virtualbox -y
```

We provide a Centos 7 Minimal Install [Virtualbox Image](#) using LVM and seprate filesystems mounted where Gawati takes up space:

<http://gawati.org/GawatiVM.7z>

The VM is 7zip packed. Unpack it into folder “%USERPROFILE%\VirtualBox VMs”. Add the VM to VirtualBox Manage using Machine -> Add , browse into the “Gawati” Folder and select the Gawati (.vbox) file.

To run it for development we recommend to not start this instance, instead create a linked clone and run that. To do so, highlight the Gawati VM, right click and “Clone”, select “Expert Mode”, activate “Linked Clone” and name it “Gawati Clean”. This clone will keep a fresh installation of Gawati. Start the “Gawati Clean” VM.

The VM is configured with dynamic IP (if its your first VM, tends to be 192.168.56.101). Log in to the VM console:

Log in credentials:

```
User root
Password MyGawatiLocal
```

Check IP addr:

```
ip addr show dev eth0
```

Add an entry to your hosts file at %WINDIR%\system32\drivers\etc using an administrative instance of notepad and add an entry equivalent to this, using the IP of your VM:

```
192.168.56.101 my.gawati.local
```

You can connect to it using ssh:

```
kitty -pw MyGawatiLocal -ssh root@my.gawati.local
```

Allow all traffic from your PC to your VM (dont do this for internet facing servers)

```
firewall-cmd --zone=trusted --change-interface=eth0 --permanent
```

From Gawati installer documentation, just download the installer as described, run it twice and reboot the system after installation to activate kernel configurations and have services bind to IPs correctly

```
curl "https://gawati.org/setup" -o setup
chmod 755 setup
./setup
./setup
echo Take note of the admin credentials, then press >ENTER<
read
poweroff
```

Create another linked clone as above, but name it “Gawati Dev”. You can then run this clone headless, and when you are done with it or broke it, delete it (with deleting files) and create a new clone to restart on a clean slate.

### Install your desktop development tools

#### Install development applications

in administrative cmd.exe run

```
choco install git jdk8 ant visualstudiocode -y
```

## Configure Visual Studio Code

Go to File -> Preferences -> Settings (Ctrl+),. Paste into rightmost tab titled 'Place your settings here...'

```
{
  "telemetry.enableTelemetry": false,
  "telemetry.enableCrashReporter": false,
  "files.autoGuessEncoding": true
}
```

Go to View -> Extensions (Ctrl+Shift+X)

Install the following plugins from there:

- XML Tools
- XML Formatter

For writing documentation install:

- reStructuredText

## Map a drive to Gawati server

Exist DB server allows WebDav access from localhost only, so we will use SSH forwarding to make our connection appear local.

Open a new cmd shell and connect to your VM using

```
kitty -pw MyGawatiLocal -ssh root@my.gawati.local -L 10443:localhost:10443
```

This will tunnel localhost:10443 to your server:10443 and encrypt the communication on its path. You can lower this shell, leaving it running in the background. This forwarding allows you to access the exist instance as a local service. For example you can now browse <https://localhost:10443> where you can log in as admin user (credentials received in server installation) to the (remote) server.

In a new cmd shell, replace 'youradminpassword' with the password retrieved above and run

```
net use x: "https://localhost:10443/exist/webdav/db/apps" /user:admin_
↪youradminpassword
```

You can close this cmd window.

Open the new X: drive in Visual Studio Code in File -> Open Folder (CTRL+K -> CTRL+O)

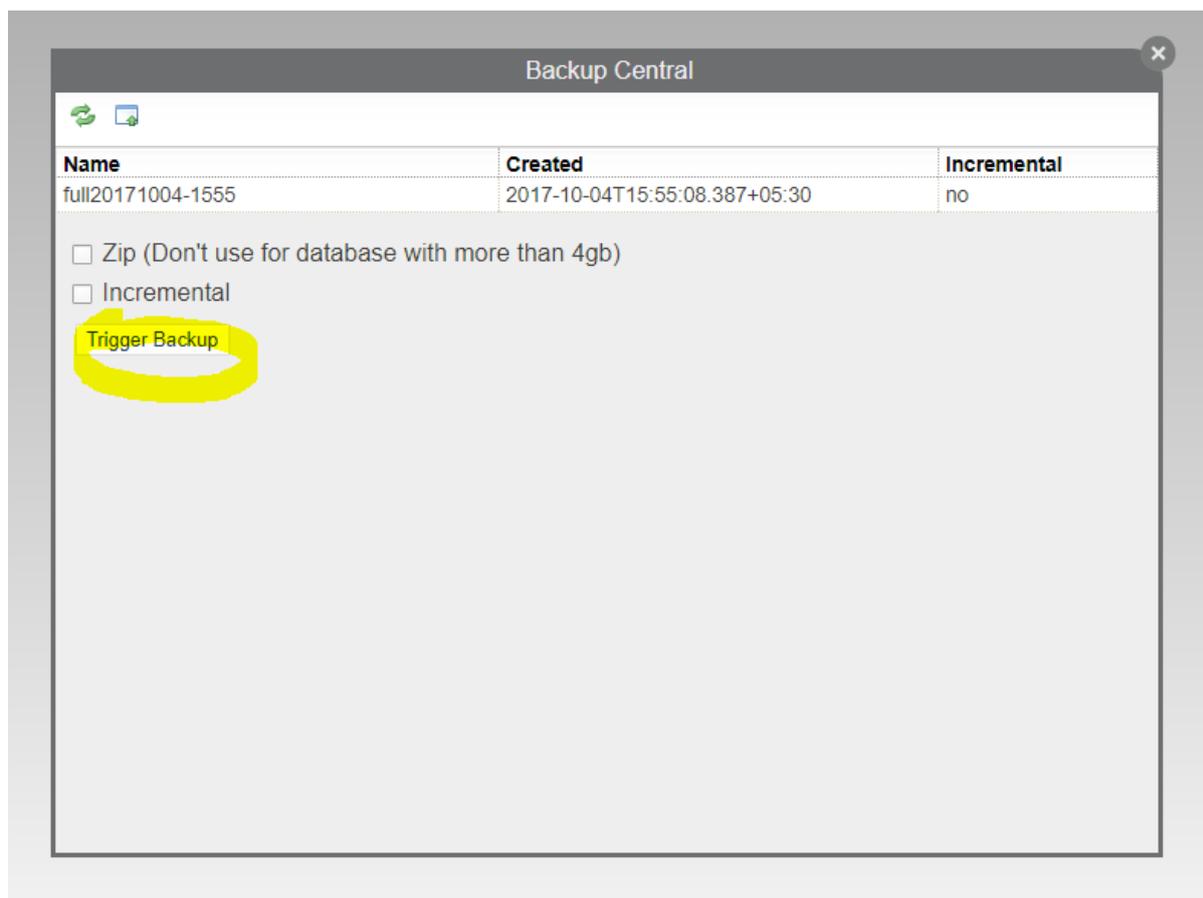
## 5.2.3 Development Workflow

The standard development cycle is as follows:

1. clone the projects from github
2. build the projects where necessary ('gawati portal', 'gawati data', 'gawati data xml')
3. deploy onto eXist-db ('gawati portal', 'gawati data', 'gawati data xml')

Code for eXist-db packages requires an additional step. You will need to export the database onto the file-system and then merge it into your github clone folder:

The database contents get exported to the file system:



In the image the exported **'gawati portal'\_** folder is selected. You will need to compare this folder with the git cloned folder on your file system using a tool like **'WinMerge'\_(on Windows)** or **'Meld'\_(on Linux)** or **'Meld OS X'\_,** and merge the changed files. After which you can commit your changes.

## 5.2.4 Building code from Github

We are going to look at 2 components of Gawati:

- the Gawati-Portal component: Provides a web portal interface to Legal data on Gawati
- the Gawati-Data component: Provides a REST API to access Gawati documents from the XML database.

The Portal accesses data and documents from the XML database via the Gawati-Data server's REST APIs.

The build process for these components is a trivial one. It merely packages the files into a format expected by eXist-db, and then the packages are deployed on eXist-db.

For example, to deploy Gawati-Data on the eXist-db server, do the following:

```
https://github.com/gawati/gawati-data.git
cd gawati-data
```

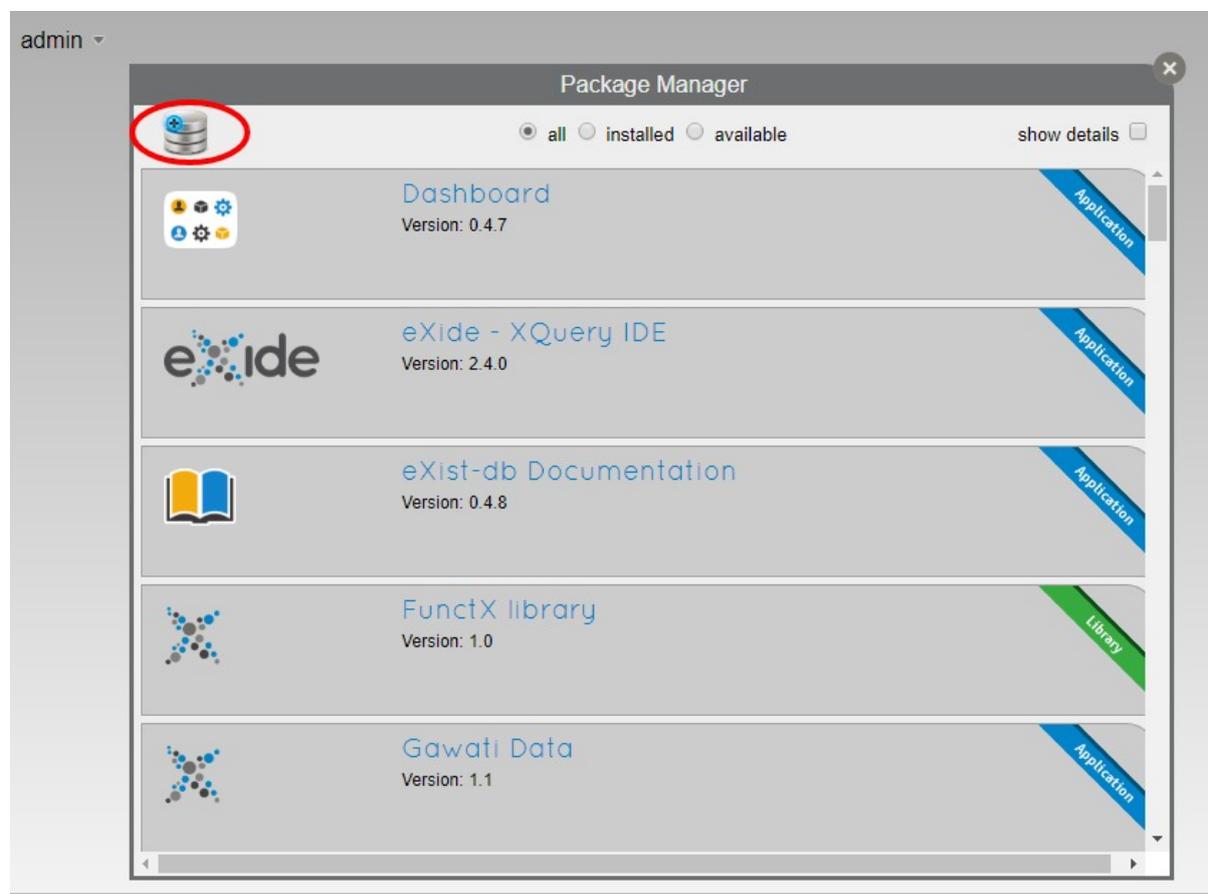
The source code for the Gawati-Data server is in the *gawati-data* folder, you can make code changes there. Finally package your code:

```
ant xar
```

exist-341	Name	Date modified
└─ autodeploy	└─ _auth	26-Oct-17 12:33 PM
└─ bin	└─ _cache	26-Oct-17 12:33 PM
└─ build	└─ _configs	26-Oct-17 12:33 PM
└─ extensions	└─ _cron	26-Oct-17 12:33 PM
└─ installer	└─ _pages	26-Oct-17 12:33 PM
└─ lib	└─ modules	26-Oct-17 12:33 PM
└─ samples	└─ services	26-Oct-17 12:33 PM
└─ schema	└─ __contents__.xml	26-Oct-17 12:33 PM
└─ src	└─ about.html	26-Oct-17 12:33 PM
└─ test	└─ build.xml	26-Oct-17 12:33 PM
└─ tools	└─ collection.xconf	26-Oct-17 12:33 PM
└─ webapp	└─ content.html	26-Oct-17 12:33 PM
└─ resources	└─ controller.xql	26-Oct-17 12:33 PM
└─ WEB-INF	└─ docs-summary.html	26-Oct-17 12:33 PM
└─ classes	└─ document.html	26-Oct-17 12:33 PM
└─ data	└─ error-page.html	26-Oct-17 12:33 PM
└─ expathrepo	└─ expath-pkg.xml	26-Oct-17 12:33 PM
└─ export	└─ index.html	26-Oct-17 12:33 PM
└─ full20171020-2246	└─ LICENSE	26-Oct-17 12:33 PM
└─ full20171024-2332	└─ post-install.xql	26-Oct-17 12:33 PM
└─ full20171026-1233	└─ pre-install.xql	26-Oct-17 12:33 PM
└─ db	└─ repo.xml	26-Oct-17 12:33 PM
└─ __lost_and_found__	└─ search.html	26-Oct-17 12:33 PM
└─ apps	└─ test.xql	26-Oct-17 12:33 PM
└─ dashboard	└─ themes-summary.html	26-Oct-17 12:33 PM
└─ dc3	└─ xml.html	26-Oct-17 12:33 PM
└─ doc		
└─ eXide		
└─ fundocs		
└─ gawati-data		
└─ gawati-portal		

This will generate a file *gawati-data-X.X.X.xar* in the *./build* folder, which you will install into eXist-db via the Dashboard.

If you have a stock installation of eXist-db, it will be running on port 8080. Access eXist-db on that port via the web-browser. Login as admin, and that should bring you to the page <http://localhost:8080/exist/apps/dashboard/index.html>. In the dashboard click on *Package Manager*:



Click the + icon, and select the package you just built in the *build* folder and install it into eXist-db. You will find the package accessible via the URL: *eXist gawati data* <<http://localhost:8080/exist/apps/gawati-data>>

### 5.2.5 Customizing Gawati

See Theming Guide .

## 5.3 Coding Guidelines

### 5.3.1 General

This provides general coding guidelines for Gawati code. This document covers different programming languages used in Gawati: XQuery (.xql, .xqm), JavaScript (.js), Java (.java). The objective of having guidelines is to keep the code-base consistent and readable which is a critical part keeping the code maintainable and manageable.

## 5.3.2 Basic Rules across all languages

### No mixing of tabs and spaces

The code must use 4 spaces (instead of a tab) for indentation, except for third party libraries (e.g. XSLTforms, YUI). Most modern code editing tools support mapping a tab to 4 spaces. Every logical depth level in the code needs to have an indent. There should be no whitespace at the end of the line.

### Use common sense

To make code readable on the screen.

### Line length

Must be kept to a maximum of 80 characters (or less). This allows reading of the code on laptops without side-scrolling.

### Unix style line-breaks

Use [unix style line-breaks](#) rather windows line breaks.

### Comments

Comments need to talk about the “why”. This is useful not just for others but for yourself when you read the code at a later time. They need to be well written and clear and need to state a date and an author if it is important commentary worth revisiting. For such revisit comments, a format has been specified below.

### *Revisit* Comments

Use *Revisit* comments for code that is temporary, a short-term solution to be reworked, or for code on which there is still questions to be resolved or understood. Multiple related *Revisit* comments (that use same *label*) may be used across the project source, and across different languages.

A *Revisit* comments should have a:

- `!+ :` must explicitly start with these 2 characters
- *label* : for easier grepping, and to be able to relate multiple comments across the project source (even in different languages)
- *author identifier* : so others know who to follow-up with if needed, this can be the github handle of the committer.
- *date* : so it is easier to judge relevance and status at a later time
- *description* : should also indicate what conditions/events would render the comment obsolete.

The following is an example of a revisit comment:

```

1  /**
2   * !+LINK_ROUTE(kohsah, 2017-12-31) Switching to using linkRoute instead of hard-
3   * ↔coded link URL
4   *
   */

```

### Formal Documentation comments

These formal documentation comments allow running documentation generation tools to extract documentation from code. For this we recommend the following for different programming languages

- Javascript - [SphinxJS](#) style comments for functions and modules
- XQuery - [XQDoc](#) style comments for functions and modules
- Java - [JavaDoc](#) style comments for functions and classes.

### 5.3.3 Language Specific Coding Guidelines

The below are coding guidelines of how the code is to be structured and written. The reason for doing this is to have consistent and readable code. Many programming languages and frameworks provide multiple ways of doing the same thing - for e.g. in React JS you can declare components either declaring a class directly, or by calling a specific `createClass` api, both approaches may look correct, but both result in different looking code – which reduces readability. These coding guidelines below for different languages and frameworks encourage uniformity and consistency.

#### React JS

Our front-end is primarily using React JS, and we recommend following the [Airbnb React JS coding style guide](#) . Additionally we recommend the following guidelines for our React code-base.

#### ES 2015

We recommend using [ES 2015](#) for all front-end code.

so

```
import { getLangs } from "../utils/i18nhelper";
```

is good,

```
require("../utils/i18nhelper");
```

is not good.

#### AJAX Http Client

Use [axios](#) (not `fetch` or `XMLHttpRequest`) for both client and server (Node) usage.

#### Reading/Writing Files

Use only async apis.

#### Callbacks vs Promises

Use Promises where possible. For APIs where there is no promise based version available, use an API promisifier (like [bluebird](#))

## Structuring Component includes

Provide a line of white space between different kinds of imports. for e.g. :

```

1  // external component includes
2  import React from 'react';
3  import axios from 'axios';
4
5  // utility function includes
6  import {apiGetCall} from '../api';
7  import {coerceIntoArray} from '../utils/generalhelper';
8
9  // component & container includes
10 import DivFeed from '../components/DivFeed';
11 import DivListing from '../components/DivListing';
12 import ExprAbstract from './ExprAbstract';
13 import SearchListPaginator from '../components/SearchListPaginator';
14 import GwSpinner from '../components/GwSpinner';
15
16 // css & image includes
17 import '../css/ListingContentColumn.css';

```

## 5.4 Using VSCode for Development

### 5.4.1 Introduction

We recommend using Visual Studio Code for Gawati Development. It is available as a free download (<https://code.visualstudio.com/>) and works on most major OS Platforms.

### 5.4.2 Setting up defaults

Go to *File-> Preferences -> Settings*. Add / Edit entry “*files.encoding*” to “*utf8*”. You can also set “*files.autoGuessEncoding*” to *true*.

### 5.4.3 Using VS Code with eXist DB

eXist-DB runs XQuery code from within an eXist-db database collection. You can use VSCode to directly edit XQUery, XSLT or XML within eXist-db. Assumption is you have installed eXist-db and VSCode on the same machine.

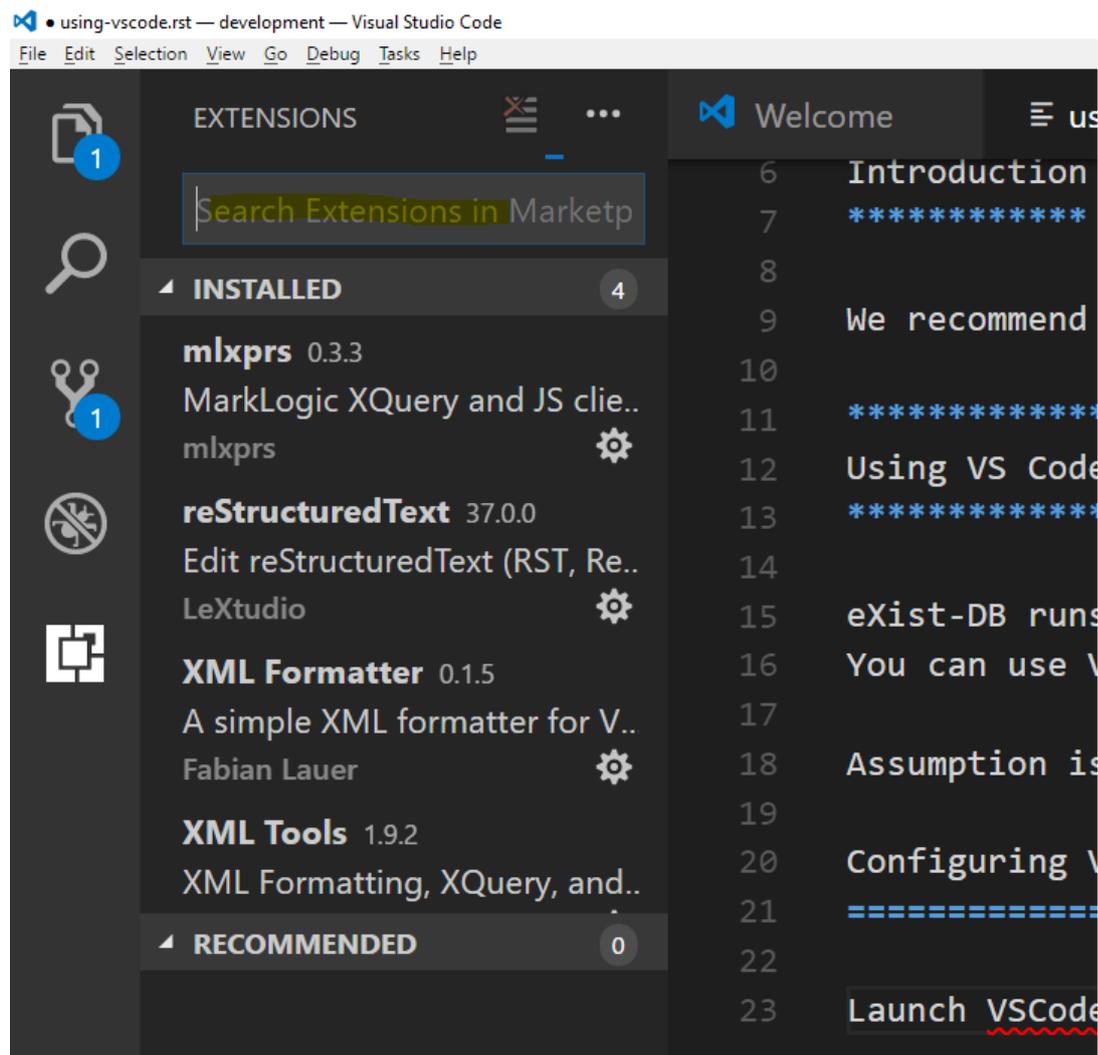
#### Configuring VSCode

Launch VSCode, and then type *Ctrl+Shift+X*, and you will get the extension marketplace search box –

**Install the following plugins from there:**

- *XML Tools*
- *XML Formatter*

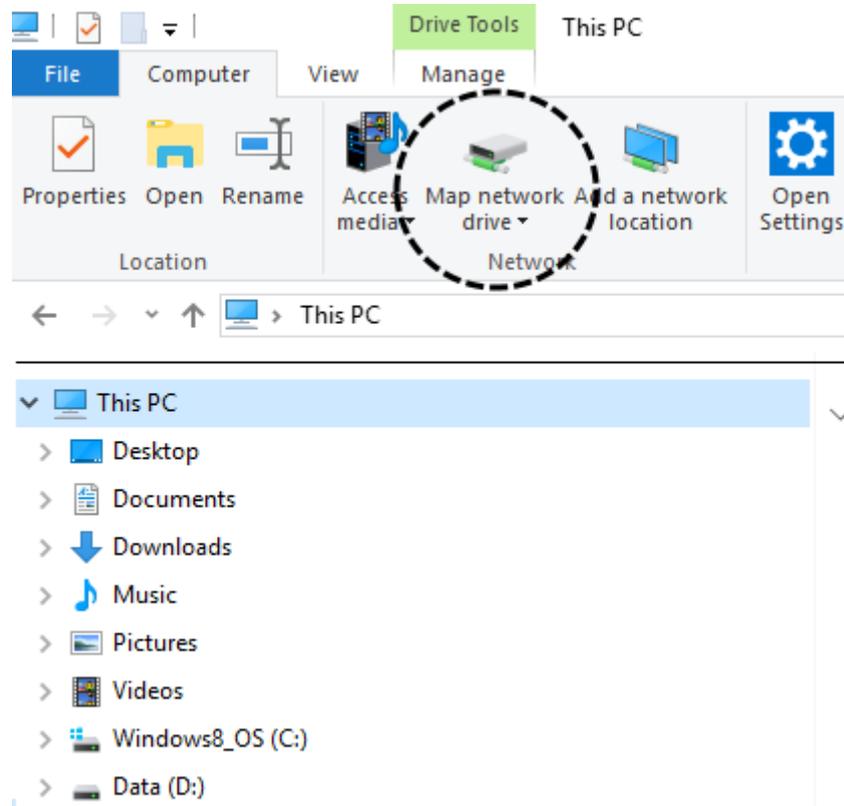
You can also install other plugins that you may need for CSS / JS development.



## Connecting to eXist-db via WebDav

eXist-db provides an inbuilt Web-Dav server (<https://exist-db.org/exist/apps/doc/webdav.xml>) by default, this allows us to map the eXist collection as a folder.

Launch windows explorer, click on “This-PC” and then click “Map Network Drive”:



You will need to connect using SSL, for example the *gawati-portal* project installs in a collection in eXist-db with the following path: `/db/apps/gawati-portal` to access that you will need to enter: `https://localhost:8443/exist/webdav/db/apps/gawati-portal` (as shown below) –

Remember to check the “Reconnect at Sign-in” option, that way your network mapping to the eXist dav folder will be persistent between reboots of your computer.

It will prompt you for a user name and password, enter the user name and password credentials for the collection / application. For the *gawati-portal* application for example, the default user name is *gawatiportal*. Once that is complete you will be able to open the ‘Z:’ folder in VSCode, and edit the XQuery, XSLT code.

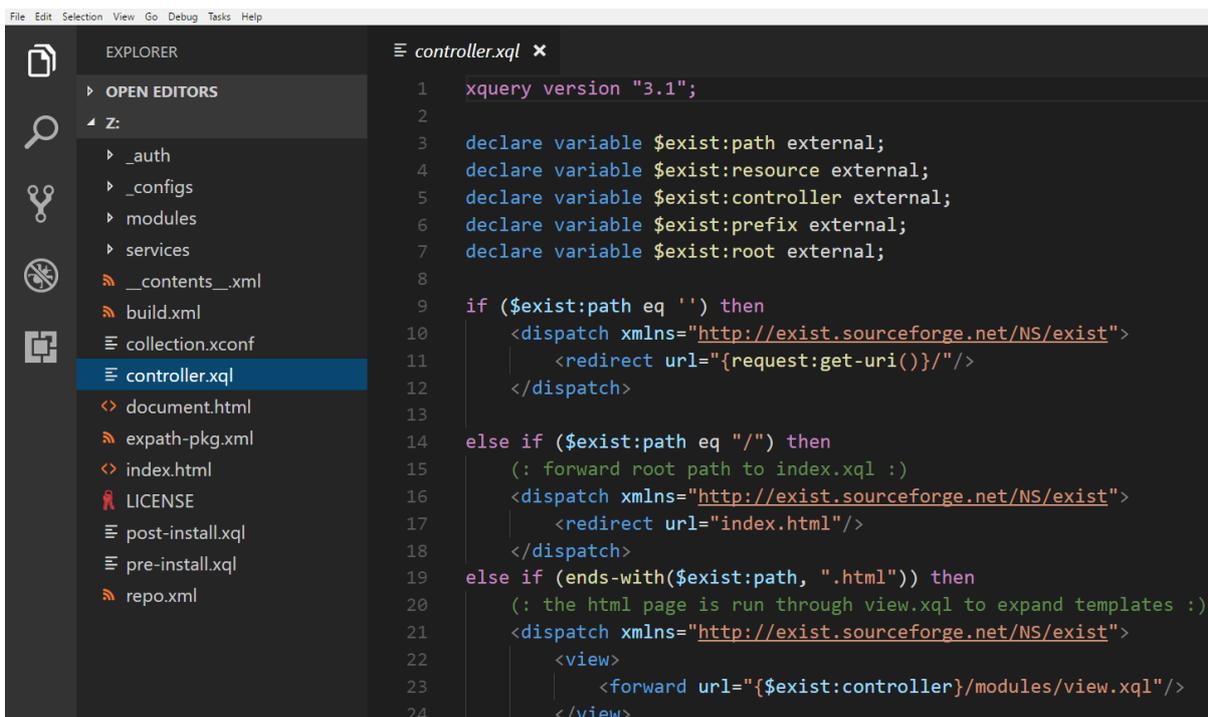
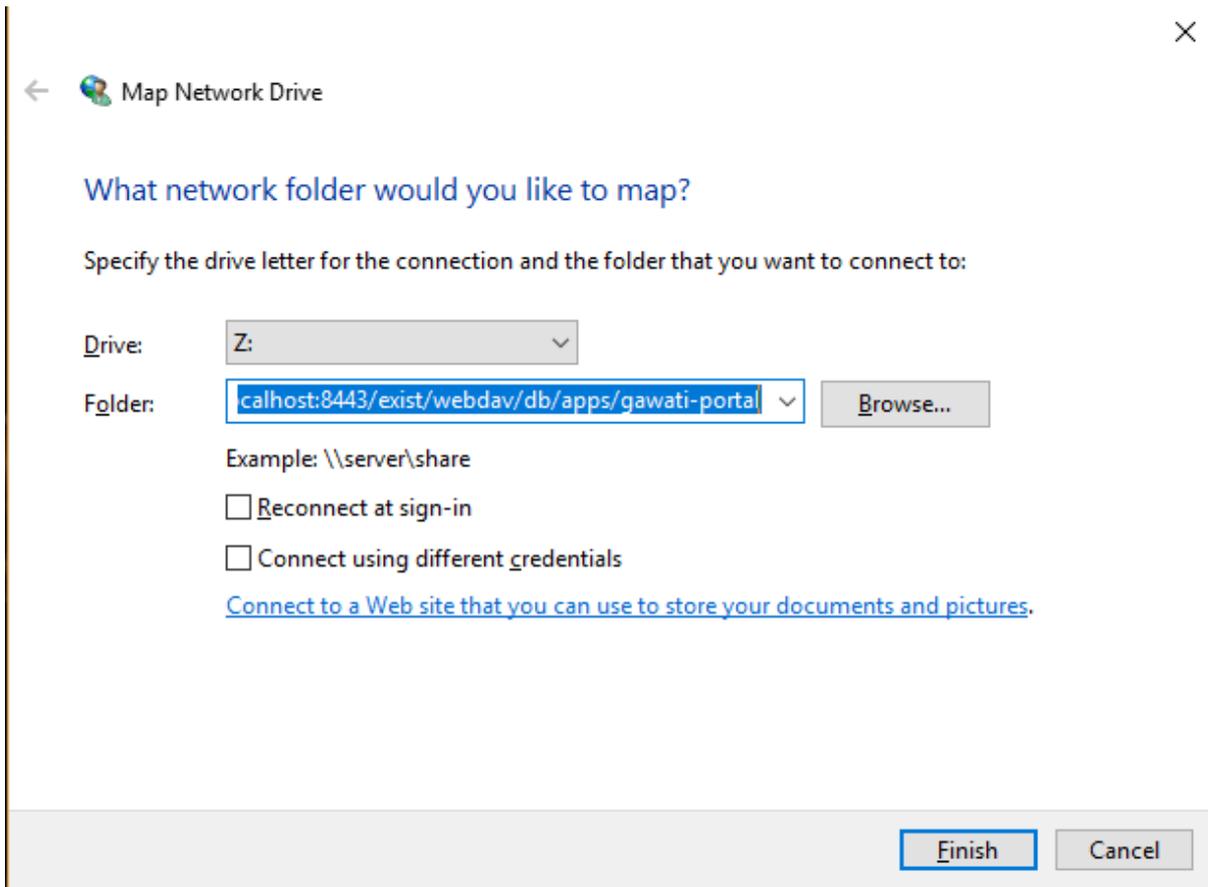
You can save, edit and create new files in the eXist collection via VSCode now.

## 5.5 Data Server APIs

### 5.5.1 Introduction

The Data Server has various APIs to access the Data from the Gawati System. The Data is accessible via REST APIs, which have been documented here. Full documents are served as *Akoma Ntoso* Documents.

These APIs allow accessing a list of documents or the document itself using different kinds of queries. Each API has 2 end points, an XML endpoint that produces outputs in XML (*text/xml*), and a JSON (*application/json*) end point that produces outputs in JSON.



**Table of Contents**

- *Listing of documents filtered by language*
- *Listing of documents filtered by year*
- *Listing of documents filtered by keywords*
- *Listing of documents filtered by countries*
- *Listing of recent documents*
- *Return a complete document based on its IRI*
- *Advanced Document Filtering API*

**Listing of documents filtered by language**

Returns a list of document abstracts (summaries) filtered by a specific language.

- Method: GET
- End-points:
  - XML endpoint - `/gwd/search/languages/summary`
  - JSON endpoint - `/gwd/search/languages/summary/json`
- Parameters:
  - `doclang` - ISO639-2 Alpha 3 language code
  - `count` - the number of results to return
  - `from` - the paging point where to return results from.

**Example:**

The following makes a request for 1 english document in json format.

```
1 $ curl "http://localhost/gwd/search/languages/summary/json?doclang=eng&count=1&
   ↪from=1"
```

returns:

```
1 {
2   "timestamp": "2018-02-21T10:36:34.371+05:30",
3   "exprAbstracts": {
4     "orderedby": "dt-updated-desc",
5     "records": "23",
6     "pagesize": "1",
7     "itemsfrom": "1",
8     "totalpages": "23",
9     "currentpage": "2",
10    "exprAbstract": {
11      "expr-iri": "/akn/mr/act/2011-01-01/gn_no_86-2011/eng@/!main",
12      "work-iri": "/akn/mr/act/2011-01-01/gn_no_86-2011/!main",
13      "date": [
14        {
15          "name": "work",
16          "value": "2011-01-01"
17        },
18        {
19          "name": "expression",
20          "value": "2011-01-01"
21        }
13      ]
14    }
15  }
16 }
```

(continues on next page)

(continued from previous page)

```

22     ],
23     "type": {
24         "name": "legislation",
25         "aknType": "act"
26     },
27     "country": {
28         "value": "mr",
29         "showAs": "Mauritania"
30     },
31     "language": {
32         "value": "eng",
33         "showAs": "English"
34     },
35     "publishedAs": "Distributive Trades (Remuneration Order) (Amendment)
↵ Regulations 2011 (Amended)",
36     "number": {
37         "value": "gn_no_86-2011",
38         "showAs": "GN No. 86/2011"
39     },
40     "componentLink": {
41         "src": "/akn/mr/act/2011-01-01/gn_no_86-2011/eng@/!main.pdf",
42         "value": "akn_mr_act_2011-01-01_gn_no_86-2011_eng_main.pdf"
43     },
44     "thumbnail": {"src": "th_akn_mr_act_2011-01-01_gn_no_86-2011_eng_main.
↵ png"}
45     }
46 }
47 }

```

The following makes a request for 1 english document in XML format.

```

1 $ curl "http://localhost/gwd/search/languages/summary?doclang=eng&count=1&from=1"

```

returns:

```

1 <gwd:package xmlns:gwd="http://gawati.org/ns/1.0/data" timestamp="2018-02-
↵ 21T10:37:55.612+05:30">
2   <gwd:exprAbstracts orderedby="dt-updated-desc" records="23" pagesize="1"
↵ itemsfrom="1" totalpages="23" currentpage="2">
3     <gwd:exprAbstract expr-iri="/akn/mr/act/2011-01-01/gn_no_86-2011/eng@/!main
↵ " work-iri="/akn/mr/act/2011-01-01/gn_no_86-2011/!main">
4       <gwd:date name="work" value="2011-01-01"/>
5       <gwd:date name="expression" value="2011-01-01"/>
6       <gwd:type name="legislation" aknType="act"/>
7       <gwd:country value="mr" showAs="Mauritania"/>
8       <gwd:language value="eng" showAs="English"/>
9       <gwd:publishedAs>Distributive Trades (Remuneration Order) (Amendment)
↵ Regulations 2011 (Amended)</gwd:publishedAs>
10      <gwd:number value="gn_no_86-2011" showAs="GN No. 86/2011"/>
11      <gwd:componentLink src="/akn/mr/act/2011-01-01/gn_no_86-2011/eng@/!
↵ main.pdf" value="akn_mr_act_2011-01-01_gn_no_86-2011_eng_main.pdf"/>
12      <gwd:thumbnail src="th_akn_mr_act_2011-01-01_gn_no_86-2011_eng_main.png
↵ "/>
13    </gwd:exprAbstract>
14  </gwd:exprAbstracts>
15 </gwd:package>

```

The outputs are exactly the same in terms of content, only the format differs.

## Listing of documents filtered by year

Returns an abstracted list of documents, filtered by the year.

- Method: GET
- End-points:
  - XML endpoint - `/gwd/search/years/summary`
  - JSON endpoint - `/gwd/search/years/summary/json`
- Parameters:
  - year - four digit year
  - count - the number of results to return
  - from - the paging point where to return results from.

## Listing of documents filtered by keywords

Returns an abstracted list of documents, filtered by one or more keywords.

- Method: GET
- End-points:
  - XML endpoint - `/gwd/search/keywords/summary`
  - JSON endpoint - `/gwd/search/keywords/summary/json`
- Parameters:
  - kw(1+) - keyword, this parameter can be specified multiple times
  - count - the number of results to return
  - from - the paging point where to return results from.

Note here that the *kw* parameter can be specified multiple times.

For example, the following returns 2 documents when the keyword *Finance* is specified:

```
1 $ curl "http://localhost/gwd/search/keywords/summary?kw=Finance&count=10&from=1"
```

returns abstracts for 2 documents:

```
1 <gwd:package xmlns:gwd="http://gawati.org/ns/1.0/data" timestamp="2018-02-
  ↳21T10:55:33.755+05:30">
2   <gwd:exprAbstracts orderedby="dt-updated-desc" records="2" pagesize="10"
  ↳itemsfrom="1" totalpages="1" currentpage="1">
3     <gwd:exprAbstract expr-iri="/akn/bf/judgment/2016-04-22/arrêt_no003_-2003-
  ↳2004/fra@/!main" work-iri="/akn/bf/judgment/2016-04-22/arrêt_no003_-2003-2004/!
  ↳main">
4       <gwd:date name="work" value="2016-04-22"/>
5       <gwd:date name="expression" value="2016-04-22"/>
6       <gwd:type name="legislation" aknType="act"/>
7       <gwd:country value="bf" showAs="Burkina Faso"/>
8       <gwd:language value="fra" showAs="Français"/>
9       <gwd:publishedAs>Conseil d'Etat , Chambre du contentieux , Madame S.F.
  ↳ et 14 autres Magistrats c. ETAT Burkinabé ( MJ) , 28 novembre 2003 ,Arrêt n°003_
  ↳/2003-2004</gwd:publishedAs>
10      <gwd:number value="arrêt_no003_-2003-2004" showAs="Arrêt n°003 /2003-
  ↳2004"/>
11      <gwd:componentLink src="/akn/bf/judgment/2016-04-22/arrêt_no003_-2003-
  ↳2004/fra@/!main.pdf" value="akn_bf_judgment_2016-04-22_arrêt_no003_-2003-2004_
  ↳fra_main.pdf"/>
```

(continues on next page)

(continued from previous page)

```

12     <gwd:thumbnail src="th_akn_bf_judgment_2016-04-22_arrêt_no003_-2003-
↳2004_fra_main.png"/>
13     </gwd:exprAbstract>
14     <gwd:exprAbstract expr-iri="/akn/bf/judgment/2016-05-12/jugement_no_088/
↳fra@/!main" work-iri="/akn/bf/judgment/2016-05-12/jugement_no_088/!main">
15         <gwd:date name="work" value="2016-05-12"/>
16         <gwd:date name="expression" value="2016-05-12"/>
17         <gwd:type name="legislation" aknType="act"/>
18         <gwd:country value="bf" showAs="Burkina Faso"/>
19         <gwd:language value="fra" showAs="Français"/>
20         <gwd:publishedAs>Tribunal du travail de Ouagadougou( Burkina Faso),
↳Monsieur K.J.M.c Monsieur B.A. , 20 MAI 2005 , JUGEMENT N° 088</
↳gwd:publishedAs>
21         <gwd:number value="jugement_no_088" showAs="JUGEMENT N° 088"/>
22         <gwd:componentLink src="/akn/bf/judgment/2016-05-12/jugement_no_088/
↳fra@/!main.pdf" value="akn_bf_judgment_2016-05-12_jugement_no_088_fra_main.pdf
↳"/>
23         <gwd:thumbnail src="th_akn_bf_judgment_2016-05-12_jugement_no_088_
↳fra_main.png"/>
24     </gwd:exprAbstract>
25 </gwd:exprAbstracts>
26 </gwd:package>

```

When queried for 2 keywords, like below:

```

1 $ curl "http://localhost/gwd/search/keywords/summary?kw=Finance&kw=Tax&count=10&
↳from=1"

```

returns 3 documents (i.e. documents having either of the 2 keywords) :

```

1 <gwd:package xmlns:gwd="http://gawati.org/ns/1.0/data" timestamp="2018-02-
↳21T11:00:45.176+05:30">
2     <gwd:exprAbstracts orderedby="dt-updated-desc" records="3" pagesize="10"
↳
↳itemsfrom="1" totalpages="1" currentpage="1">
3         <gwd:exprAbstract expr-iri="/akn/bf/judgment/2016-04-22/arrêt_no003_-2003-
↳2004/fra@/!main" work-iri="/akn/bf/judgment/2016-04-22/arrêt_no003_-2003-2004/!
↳main">
4             <gwd:date name="work" value="2016-04-22"/>
5             <gwd:date name="expression" value="2016-04-22"/>
6             <gwd:type name="legislation" aknType="act"/>
7             <gwd:country value="bf" showAs="Burkina Faso"/>
8             <gwd:language value="fra" showAs="Français"/>
9             <gwd:publishedAs>Conseil d'Etat , Chambre du contentieux , Madame S.F.
↳ et 14 autres Magistrats c. ETAT Burkinabé ( MJ) , 28 novembre 2003 ,Arrêt n°003,
↳/2003-2004</gwd:publishedAs>
10             <gwd:number value="arrêt_no003_-2003-2004" showAs="Arrêt n°003 /2003-
↳2004"/>
11             <gwd:componentLink src="/akn/bf/judgment/2016-04-22/arrêt_no003_-2003-
↳2004/fra@/!main.pdf" value="akn_bf_judgment_2016-04-22_arrêt_no003_-2003-2004_
↳fra_main.pdf"/>
12             <gwd:thumbnail src="th_akn_bf_judgment_2016-04-22_arrêt_no003_-2003-
↳2004_fra_main.png"/>
13         </gwd:exprAbstract>
14         <gwd:exprAbstract expr-iri="/akn/bf/judgment/2016-05-12/jugement_no_088/
↳fra@/!main" work-iri="/akn/bf/judgment/2016-05-12/jugement_no_088/!main">
15             <gwd:date name="work" value="2016-05-12"/>
16             <gwd:date name="expression" value="2016-05-12"/>
17             <gwd:type name="legislation" aknType="act"/>
18             <gwd:country value="bf" showAs="Burkina Faso"/>
19             <gwd:language value="fra" showAs="Français"/>
20             <gwd:publishedAs>Tribunal du travail de Ouagadougou( Burkina Faso),
↳
↳Monsieur K.J.M.c Monsieur B.A. , 20 MAI 2005 , JUGEMENT N° 088</
↳gwd:publishedAs>

```

(continues on next page)

(continued from previous page)

```

21     <gwd:number value="jugement_no_088" showAs="JUGEMENT N° 088"/>
22     <gwd:componentLink src="/akn/bf/judgment/2016-05-12/jugement_no_088/
↵fra@/!main.pdf" value="akn_bf_judgment_2016-05-12_jugement_no_088_fra_main.pdf
↵"/>
23     <gwd:thumbnail src="th_akn_bf_judgment_2016-05-12_jugement_no_088_
↵fra_main.png"/>
24     </gwd:exprAbstract>
25     <gwd:exprAbstract expr-iri="/akn/mr/act/1963-10-12/gn_no_150-1983/eng@/!
↵main" work-iri="/akn/mr/act/1963-10-12/gn_no_150-1983!/main">
26         <gwd:date name="work" value="1963-10-12"/>
27         <gwd:date name="expression" value="1963-10-12"/>
28         <gwd:type name="legislation" aknType="act"/>
29         <gwd:country value="mr" showAs="Mauritania"/>
30         <gwd:language value="eng" showAs="English"/>
31         <gwd:publishedAs>Sales Tax (Amendment of schedule) Regulations 1983_
↵(Amended) </gwd:publishedAs>
32         <gwd:number value="gn_no_150-1983" showAs="GN No. 150/1983"/>
33         <gwd:componentLink src="/akn/mr/act/1963-10-12/gn_no_150-1983/eng@/!
↵main.pdf" value="akn_mr_act_1963-10-12_gn_no_150-1983_eng_main.pdf"/>
34         <gwd:thumbnail src="th_akn_mr_act_1963-10-12_gn_no_150-1983_eng_main.
↵png"/>
35     </gwd:exprAbstract>
36 </gwd:exprAbstracts>
37 </gwd:package>

```

## Listing of documents filtered by countries

Returns an abstracted list of documents, filtered by one or more countries.

- Method: GET
- End-points:
  - XML endpoint - `/gwd/search/countries/summary`
  - JSON endpoint - `/gwd/search/countries/summary/json`
- Parameters:
  - `country(1+)` - country, this parameter can be specified multiple times. Countries are specified as [ISO ALPH-2 country codes](#).
  - `count` - the number of results to return
  - `from` - the paging point where to return results from.

Note here that the `country` parameter can be specified multiple times.

## Listing of recent documents

Returns documents based on recency. Most recent appear first. Recency is established based on the updated date metadata of the document :

## Return a complete document based on its IRI

Returns documents based on recency. Most recent appear first. Recency is established based on the updated date metadata of the document :

- Method: GET
- End-points:

- XML endpoint - `/gwd/doc`
- JSON endpoint - `/gwd/doc/json`
- Parameters:
- `iri` - The `FRBRthis/@value` of the document to be retrieved

For example, to retrieve the document with the IRI `/akn/mr/act/1963-10-12/gn_no_150-1983/eng@/!main` the following:

```
$ curl "http://localhost/gwd/doc?iri=/akn/mr/act/1963-10-12/gn_no_150-1983/eng@/!main"
↪%21main"
```

Note that to run this with curl we need to escape the `!` character in the IRI as `%21`. returns, the XML document:

```
1 <an:akomaNtoso xmlns:gw="http://gawati.org/ns/1.0" xmlns:gxsl="http://gawati.org/
↪ns/1.0/xsl" xmlns:an="http://docs.oasis-open.org/legaldocml/ns/akn/3.0">
2   <an:act name="act">
3     <an:meta>
4       <an:identification source="#gawati">
5         <an:FRBRWork>
6           <an:FRBRthis value="/akn/mr/act/1963-10-12/gn_no_150-1983/!main
↪"/>
7           <an:FRBRuri value="/akn/mr/act/1963-10-12/gn_no_150-1983"/>
8           <an:FRBRdate name="Work Date" date="1963-10-12"/>
9           <an:FRBRauthor href="#author"/>
10          <an:FRBRcountry value="mr" showAs="Mauritania"/>
11          <an:FRBRnumber value="gn_no_150-1983" showAs="GN No. 150/1983"/
↪>
12          <an:FRBRprescriptive value="false"/>
13          <an:FRBRauthoritative value="false"/>
14        </an:FRBRWork>
15        <an:FRBRExpression>
16          <an:FRBRthis value="/akn/mr/act/1963-10-12/gn_no_150-1983/eng@/
↪!main"/>
17          <an:FRBRuri value="/akn/mr/act/1963-10-12/gn_no_150-1983/eng@"/
↪>
18          <an:FRBRdate name="Expression Date" date="1963-10-12"/>
19          <an:FRBRauthor href="#author"/>
20          <an:FRBRlanguage language="eng"/>
21        </an:FRBRExpression>
22        <an:FRBRManifestation>
23          <an:FRBRthis value="/akn/mr/act/1963-10-12/gn_no_150-1983/eng@/
↪!main.xml"/>
24          <an:FRBRuri value="/akn/mr/act/1963-10-12/gn_no_150-1983/eng@/.
↪akn"/>
25          <an:FRBRdate name="Manifestation Date" date="2016-05-09"/>
26          <an:FRBRauthor href="#author"/>
27          <an:FRBRformat value="xml"/>
28        </an:FRBRManifestation>
29      </an:identification>
30      <an:publication date="1963-10-12" showAs="Sales Tax (Amendment of
↪schedule) Regulations 1983 (Amended)" name="Act" number="GN No. 150/1983"/>
31      <an:classification source="#legacy">
32        <an:keyword eId="ontology.dictionary.gawati.legacy.Regulation"
↪value="Regulation" showAs="Regulation" dictionary="#gawati-legacy"/>
33        <an:keyword eId="ontology.dictionary.gawati.legacy.Sale" value=
↪"Sale" showAs="Sale" dictionary="#gawati-legacy"/>
34        <an:keyword eId="ontology.dictionary.gawati.legacy.Tax" value="Tax
↪" showAs="Tax" dictionary="#gawati-legacy"/>
35        <an:keyword eId="ontology.dictionary.gawati.legacy.Act" value="Act
↪" showAs="Act" dictionary="#gawati-legacy"/>
36        <an:keyword eId="ontology.dictionary.gawati.legacy.Schedule" value=
↪"Schedule" showAs="Schedule" dictionary="#gawati-legacy"/>
```

(continues on next page)

(continued from previous page)

```

37     </an:classification>
38     <an:lifecycle source="#all">
39         <an:eventRef date="1963-10-12" source="#original" type="generation
↪"/>
40         <an:eventRef date="1983-10-12" source="#original" type="generatio" ↪
↪refersTo="#dtInForce"/>
41     </an:lifecycle>
42     <an:references source="#source">
43         <an:original eId="original" href="/akn/mr/act/1963-10-12/gn_no_150-
↪1983/eng@/!main" showAs="GN No. 150/1983"/>
44         <an:TLCOrganization eId="all" href="/ontology/Organization/
↪AfricanLawLibrary" showAs="African Law Library"/>
45         <an:TLCEvent eId="dtInForce" href="/ontology/Event/ALL/InForce" ↪
↪showAs="Entry into Force Date"/>
46         <an:TLCConcept eId="ct-Legal-Finance" href="/ontology/Concept/
↪Legacy/Legal/Finance" showAs="Finance"/>
47         <an:TLCConcept eId="ct-Legal-Tax" href="/ontology/Concept/Legacy/
↪Legal/Tax" showAs="Tax"/>
48         <an:TLCConcept eId="ct-Legal-TradeAndIndustry" href="/ontology/
↪Concept/Legacy/Legal/TradeAndIndustry" showAs="Trade and Industry"/>
49         <an:TLCConcept eId="ct-Legal-TradeLaw" href="/ontology/Concept/
↪Legacy/Legal/TradeLaw" showAs="Trade Law"/>
50     </an:references>
51     <an:proprietary source="#all">
52         <gw:gawati>
53             <gw:legacyIdentifier>africanlawlib:6741344</
↪gw:legacyIdentifier>
54             <gw:legacyOwner>16956378</gw:legacyOwner>
55             <gw:legacyCollection>L03033 Mauritius</gw:legacyCollection>
56             <gw:languages>
57                 <gw:language code="eng"/>
58             </gw:languages>
59             <gw:embeddedContents>
60                 <gw:embeddedContent eId="embedded-doc-1" type="pdf" file=
↪"MUSCM_1983GN150.pdf" state="true"/>
61             </gw:embeddedContents>
62             <gw:dateTime refersTo="#dtCreated" datetime="2016-03-
↪18T11:12:51.964Z"/>
63             <gw:dateTime refersTo="#dtModified" datetime="2016-05-
↪09T11:07:51.725Z"/>
64             <gw:date refersTo="#dtInForce" date="1983-10-12"/>
65             <gw:themes source="#legacy">
66                 <gw:theme refersTo="#ct-Legal-Finance"/>
67                 <gw:theme refersTo="#ct-Legal-Tax"/>
68                 <gw:theme refersTo="#ct-Legal-TradeAndIndustry"/>
69                 <gw:theme refersTo="#ct-Legal-TradeLaw"/>
70             </gw:themes>
71         </gw:gawati>
72     </an:proprietary>
73 </an:meta>
74 <an:body>
75     <an:book refersTo="#mainDocument">
76         <an:componentRef src="/akn/mr/act/1963-10-12/gn_no_150-1983/eng@/!
↪main.pdf" alt="akn_mr_act_1963-10-12_gn_no_150-1983_eng_main.pdf" GUID="
↪#embedded-doc-1" showAs="Sales Tax (Amendment of schedule) Regulations 1983 ↪
↪(Amended)"/>
77     </an:book>
78 </an:body>
79 </an:act>
80 </an:akomaNtoso>

```

## Advanced Document Filtering API

This API allows a more complex combination of filters, and conducting searches on the data server.

- Method: GET
- End-points:
  - XML endpoint - `/gwd/search/filter`
  - JSON endpoint - `/gwd/search/filter/json`
- Parameters:
  - `q` - filter query, based on filter XQuery syntax (see below).
  - `count` - the number of results to return
  - `from` - the paging point where to return results from.

The query syntax for the `q` parameter runs pseudo XQuery:

For example, when `q` is set to the below, it returns documents only from the year 2016:

```
1 [./an:FRBRdate[ year-from-date(@date) eq 2016 ]]
```

The below instead returns documents from both 2014 and 2016:

```
1 [./an:FRBRdate[ year-from-date(@date) eq 2016 or year-from-date(@date) eq 2014]]
```

Note that the Portal front-end does not directly compose this query, there is an intermediate query translation api that lets the server make the request using a JSON based syntax. The below is the JSON query of the above :

```
1 {"year": [2014, 2016]}
```

These filters can be stacked, the below searches for documents from 2014 and 2016 which are from “Burkina Faso”:

```
1 [./an:FRBRdate[ year-from-date(@date) eq 2016 or year-from-date(@date) eq 2014_
↪]] [./an:FRBRcountry[ @value eq 'bf' ]]
```

The Json query of the same would look like:

```
1 {"year": [2014, 2016], "countries": ["bf"]}
```

At this point the data server does not support JSON queries yet, but eventually the XQuery based API will be migrated to support only the JSON based API.

Another stacked filter supported is the language of the document:

```
1 [./an:FRBRdate[ year-from-date(@date) eq 2016 or year-from-date(@date) eq 2014 ]]
2 [./an:FRBRcountry[ @value eq 'bf' ]]
3 [./an:FRBRlanguage[ @language eq 'eng' ]] <===
```

And finally Keywords:

```
1 [./an:FRBRdate[ year-from-date(@date) eq 2016 or year-from-date(@date) eq 2014 ]]
2 [./an:FRBRcountry[ @value eq 'bf' ]]
3 [./an:FRBRlanguage[ @language eq 'eng' ]]
4 [./an:classification/an:keyword[ @value eq 'Trade' or @value eq 'FinancialLaw' ]]
↪<===
```

## 5.6 Package Versions

This page lists the most upto-date set of packages which can be used to install Gawati.

### Current Version

- Gawati 1.0.18 download link , released on: 05 July 2018
  - portal-ui : 2.0.33
  - portal-fe : 1.0.15
  - gawati-data : 1.19
  - gawati-editor-ui : 1.0.14
  - gawati-editor-fe : 1.0.11
  - gawati-client-data : 1.11
  - gawati-editor-qprocessor : 1.0.0
  - gawati-portal-publisher : 1.0.0
  - gawati-portal-qprocessor : 1.0.0
  - gawati-profiles-ui : 1.0.2
  - gawati-profiles-fe : 1.0.1

Older releases, see: ./version-compat

### Download links:

1. Download [Gawati Portal UI release](#) , [Gawati Portal UI latest](#)
2. Download [Gawati Portal Server release](#), [Gawati Portal Server latest](#)
3. Download [Gawati Data release](#), [Gawati Data latest](#)
4. Download [Gawati Data Xml release](#), [Gawati Data Xml latest](#)

## 5.7 Jenkins Setup

### 5.7.1 Installation

Run:

```
yum groupinstall 'Development Tools'  
yum install jenkins
```

### 5.7.2 Repository configuration

- Gawati repositories are added as a single resource of type “Github Organization”.
- Only branches “master” and “dev” are displayed by configuring Project Behaviour

“Filter by name (with wildcards) is configured as “master dev”. - “Automatic branch project triggering” is configured as “master|dev”

Individual repositories are included by adding a “Jenkinsfile” in the sourcecode root folder.

### 5.7.3 Jenkinsfile and Jenkins library

Packaging, upload to package repository and deployment are implemented using shell script. As individual shell calls in Jenkinsfile reinstatiate with a new environment on each call and the need for extensive escaping from within Jenkinsfile for shell script, we collect Jenkins operations in a single library limiting commands from within Jenkins to a few lines.

The Jenkins library resides in the “library” folder within the github [Jenkins repository](#).

### 5.7.4 Building packages and copy to downloadserver

For more detailed information about Jenkins library see [Jenkins library](#)

if you need the package file name body different from the package name you may defined it as variable PKF in Jenkinsfile:

```
environment {  
    PKF="portal-ui"  
}
```

Jenkinslib uses git branch name from Jenkins environment to designate target environment (dev or prod).

#### using npm

See example at <https://github.com/gawati/gawati-portal-ui/blob/dev/Jenkinsfile>

Parameters are loaded from npm environment defined in “package.json” (Package version).

#### using ant

See example at <https://github.com/gawati/gawati-data/blob/dev/Jenkinsfile>

Parameters are loaded from ant environment defined in “build.xml” (Package version).

## 5.8 Installer

### 5.8.1 System

#### Gawati Installation System

If you are looking for how to run a standard Gawati installation, please see [Server Setup](#).

#### Installation definiton file

The format is standard ini format. Section names and item names within a section must be unique.

#### options

The “options” section defines parameters that are valid for all installation components.

example content of [options] in “dev.ini”:

```
[options]
installPackages=java-1.8.0-openjdk
downloadFolder=/opt/Download
deploymentFolder=/opt
debug=0
```

**installPackages** defines Packages to be installed from the Linux Distributions own repositories

**downloadFolder** common folder for downloads

**deploymentFolder** where applicable, common folder for deploying shared components

**debug** defines debuglevel, higher number means more noise during installation (0-3)

## Installer

Installers define what to install and parameters to be used for installation. The content of demonstration installer instance in “dev.ini”:

```
[demo]
type=install
installer=template
version=1.0
user=root
instanceFolder=~
options=demooption
postinstall=postdemo
```

This installer is made for demonstration purposes. It does not execute any installation task. The instance of this installation is called *demo*. If You run multiple installations with the same installer, each one must have a different, unique instance name as its header. This name will also be used as the service / daemon name if installed as such.

**type** Installer sections are marked by the line:

```
type=install
```

**installer** Installer sections reference an installer script. The name of the installer is denoted in the installer parameter:

```
installer=template
```

This name references the folder name inside the *installers* folder containing all components of this installer including the script itself

**version** Different versions may be available for installation. The chosen version must be denoted for the installation:

```
version=1.0
```

The *installers* folder is to be found on the git repository in the folder *setup-scripts/gawati* and will be copied to the users *downloadFolder* as defined in the [options] ini files, which is in */opt/Download* by default.

In total these together:

```
[demo]
type=install
installer=template
version=1.0
```

make the installer run the script *installer/\${installer}/\${version}*. With *downloadFolder* set as */opt/Download*, in this example expand to:

```
/opt/Download/installer/template/1.0
```

which is a link to the actual script inside the folder `/opt/Download/installer/template/scripts`:

```
/opt/Download
├── installer
│   └── template
│       ├── 1.0 -> scripts/template.sh
│       └── scripts
│           ├── postdemo.sh
│           └── template.sh
```

This allows to make a single script serve multiple versions of a product if such is feasible for the given product.

**user** OS user name. The installation will typically execute as this user and/or run its service as this user:

```
user=root
```

**instanceFolder** filesystem folder into which the installation will be deployed:

```
instanceFolder=~
```

**options** options for the execution of the installation:

```
options=daemon
```

**daemon** make the installation a boot time system service / daemon

which options are supported is defined by the installer in the given section

**postinstall** additional, optional installation steps where available with the given installer can be activated by listing in postinstall

There can be any number of additional items added and defined by the installer script called in the section.

## resources

A dedicated resources section is used in special cases only. Typically installers define their requirements themselves.

Resources define additional files used for installation. They are identified by the line:

```
type=resource
```

The section header defines the name of the resource. Resource names currently must match the name of the installer function that uses them.

**download** defines, separated by whitespace

1. the filename as written to in local filesystem
2. the URL from which the resource is to be retrieved

**unpackfolder (optional)** for installations deploying shared components into deploymentFolder, the name of the shared folder that will be created in deploymentFolder

## Implementation considerations

### Applying eXistdb ports

We deviate with our configuration method from recommendations by eXistdb for the reasons below.

## mismatch between online documentation and installation content

Delivered in the package we have...

jetty.xml:

```
<Configure id="Server" class="org.eclipse.jetty.server.Server">
  <New id="httpConfig" class="org.eclipse.jetty.server.HttpConfiguration">
    <Set name="securePort">
      <Property name="jetty.httpConfig.securePort" deprecated="jetty.secure.port">
        <Default>
          <SystemProperty name="jetty.secure.port" default="8443"/>
        </Default>
      </Property>
    </Set>
  </New>
</Configure>
```

jetty-http.xml:

```
<Configure id="Server" class="org.eclipse.jetty.server.Server">
  <Call name="addConnector">
    <Arg>
      <New id="httpConnector" class="org.eclipse.jetty.server.ServerConnector">
        <Set name="port">
          <Property name="jetty.http.port" deprecated="jetty.port">
            <Default>
              <SystemProperty name="jetty.port" default="8080"/>
            </Default>
          </Property>
        </Set>
      </New>
    </Arg>
  </Call>
</Configure>
```

jetty-ssl.xml:

```
<Configure id="Server" class="org.eclipse.jetty.server.Server">
  <Call name="addConnector">
    <Arg>
      <New id="sslConnector" class="org.eclipse.jetty.server.ServerConnector">
        <Set name="port">
          <Property name="jetty.ssl.port" deprecated="ssl.port">
            <Default>
              <SystemProperty name="jetty.ssl.port" deprecated="ssl.port" default=
↪ "8443"/>
            </Default>
          </Property>
        </Set>
      </New>
    </Arg>
  </Call>
</Configure>
```

Compared to documentation at <http://exist-db.org/exist/apps/doc/troubleshooting.xml> which wants you to...

change this for nonSSL (which doesnt exist):

```
<Set name="port"><SystemProperty name="jetty.port" default="8080"/></Set>
```

change both of these for SSL (which dont exist):

```
<Set name="confidentialPort">8443</Set>
<Set name="Port">8443</Set>
```

## Options considered

changing jetty.xml, but doesnt produce the expected result:

```
sed -i "s%^(.*\)name=\"jetty.port\" default=\"[[:digit:]]*\"/>(.*\)$\1name=\
↪ "jetty.port\" default=\"${EXIST_PORT}\"/>\2% " "${EXIST_HOME}/tools/jetty/etc/
↪ jetty.xml"
```

changing the default for an undefined property instead of defining the property is not the right thing to do, but does work:

```
xmllstarlet ed -u '/Configure[@id="Server"]/New[@id="httpConfig"]/Set[@name=
↪ "securePort"]/Property[@name="jetty.httpConfig.securePort"]/Default/
↪ SystemProperty[@name="jetty.secure.port"]/@default' -v "8444" jetty.xml
```

Best candidate, defining probed system properties in jetty.xml:

```
<Call class="java.lang.System" name="setProperty">
  <Arg>jetty.port</Arg>
  <Arg>10083</Arg>
</Call>

<Call class="java.lang.System" name="setProperty">
  <Arg>jetty.ssl.port</Arg>
  <Arg>10443</Arg>
</Call>
```

### References

- <http://exist-db.org/exist/apps/doc/advanced-installation.xml>
- [http://exist-db.org/exist/apps/doc/production\\_good\\_practice.xml](http://exist-db.org/exist/apps/doc/production_good_practice.xml)
- <http://exist-db.org/exist/apps/doc/configuration.xml>
- <http://exist-db.org/exist/apps/doc/java-admin-client.xml>
- <http://exist-db.org/exist/apps/doc/troubleshooting.xml>

## 5.8.2 Modules

### centos

#### General

The goal of the module is to configure a newly minimal CentOS installation for production use as an internet connected server platform.

#### Software Packages

The module will perform a system update if it has never been performed or last update is older than 1 week. It will add essential administration tools.

#### Configuration

The system will be configured for minimum swapping and disable IPv6. Correct hostname and DNS domain name configuration will be ensured.

#### ini Variables

##### hostname

The “physical” (interface and IP independent) single hostname of the machine without domain name (no dot “.”).

##### DNSdomain

The DNS domain name to be used with the hostname variable.

## Postinstallers

None

## existapp

### General

The goal of the module is to install or update exist applications from internet sources.

### Software Packages

No OS packages to install

### Configuration

No OS configurations applied

### ini Variables

#### appname

The name of the app as it appears inside existdb namespace. Used to identify if an instance of the application is already present:

```
appname=http://gawati.org/portal
```

#### source\_url

The source off which the app is to be deployed as a URL:

```
source_url=https://github.com/gawati/gawati-portal/releases/download/1.4/gawati-portal-1.4-all.jar
```

#### exist\_instance

The exist installer instance name in the same installer ini file to which the app is to be deployed. This retrieves connection and if available password information to connect to exist:

```
exist_instance=eXist-st
```

## Postinstallers

No postinstaller available.

## existdb

### General

The goal of the module is to install existdb and apply essential configuration for production use.

### Software Packages

The module installs “xmlstarlet” to edit XML based configuration files of existdb.

### Configuration

No configurations applied

### ini Variables

#### instanceFolder

The filesystem folder where existdb binaries will be installed.

#### dataFolder

The filesystem folder where existdb data files will be stored.

#### port

The non SSL TCP port where existdb will answer http requests.

#### sslport

The SSL TCP port where existdb will answer https requests.

#### options

If “options” contains “daemon” existdb will be configured and enabled for start during boot.

### Postinstallers

1 postinstaller available called “selinux” to apply selinux context configuration for existdb.

#### fail2ban

##### General

The goal of the module is to protect the system from some attacks that can be detected by notifications in logfiles.

### Software Packages

“iptables” and “fail2ban” will be installed.

## Configuration

The reactive measures are configured in `/etc/fail2ban/jail.local`. By default we monitor login failures on ssh. Banning will be executed by temporarily blocking all incoming traffic from originating IPs. A new shell script “offenders” is added to the system listing the current status of banning.

### ini Variables

For email notifications, you need to configure a senders email address in “mailsender” and a recipient address in “mailrecipient” respectively.

## Postinstallers

None

## gawatidemodata

### General

This module feeds demodata into a Gawati instance. We provide a set of 200 documents of the ALL document dataset and matching XML data for trying and developing the Gawati system.

## Software Packages

unzip will be installed.

## Configuration

No OS configurations applied

### ini Variables

#### existst

Which exist instance will be used for uploading the data. The configuration of this instance must be available in the same installer ini:

```
existst=eXist-st
```

#### importFolder

Defines the folder into which the data for uploading will be unpacked:

```
importFolder=/tmp/import
```

## Postinstallers

No postinstaller available.

### **gawatiportal**

#### **General**

The goal of the module is to apply Gawati specific configurations in the operating system and webserver.

#### **Software Packages**

“unzip” will be installed.

#### **Configuration**

A Gawati website configuration “10-gawati.conf” file will be added to /etc/httpd/conf.d

#### **ini Variables**

The DNS name for the public gawati portal root URL needs to be specified in “GAWATI\_URL\_ROOT”. The internal URL for the backend eXist DB must be specified as “EXIST\_ST\_URL”. The eXist DB installer instance names for Gawati backend and frontend must be provided as “exstbe” and “existst” respectively.

#### **Postinstallers**

None

#### **Details**

In local apache instance, a virtual host directory will be created in /var/www/html and a matching log firectory in /var/log/httpd. These locations will be references in /etc/httpd/conf.d/10-gawati.conf. SSL key and certificate matching the Gawati URL with compliant naming must be present in /etc/pki/tls/private and /etc/pki/tls/certs respectively. By default this is prepared by the <localcert> or <letsencrypt> module. The Gawati default website template / theme will be deployed into its web root folder below /var/www/html

### **httpd**

#### **General**

The goal of the module is to install Apache webserver, clear default content and prepare it with a blank configuration for hosting of multiple (virtual) sites in a modular way.

#### **Software Packages**

“httpd” and “mod\_ssl” will be installed.

#### **Configuration**

The packaged configuration files in /etc/httpd/conf.d will be replaced with cleaned up versions, removing / disabling default content.

## ini Variables

None

## Postinstallers

None

## letsencrypt

### General

The goal of the module is to prepare the OS and install the tools for using letsencrypt certificate services. Using a postinstaller, the module provides SSL keys officially signed by letsencrypt ready for use with a local apache webserver. To maintain the validity of the certificate the postinstaller creates a cronjob to renew the certificate when needed.

### Software Packages

“epel-release” will be installed to enable use of the EPEL repository. “acme-tiny” will be installed for accessing the letsencrypt service.

### Configuration

No configurations applied

## ini Variables

The main module for installation of the acme tools, do not use variables. `postinstall=setupcerts certs=my.gawati.org`

## Postinstallers

1 postinstaller available called “setupcerts” to create key pairs and retrieve certificates from letsencrypt. “certs” specifies a comma separated list of DNSnames for which certificates shall be retrieved. These names must reach the local apache webserver on port 80 through public DNS to be successful.

Example:

```
postinstall=setupcerts
certs=my.gawati.org
```

## Details

The letsencrypt verification folder structure will be created at “/var/www/challenges/well-known/acme-challenge” Keys will be stored at “/etc/pki/tls/letsencrypt” Certificates will be stored at “/etc/pki/tls/letsencrypt” For compatibility and organisation links will be created at “/etc/ssl/letsencrypt”

## localcert

### General

The goal of the module is to create a local certificate authority and local certificates self signed by this CA.

### Software Packages

OS package openssl will be installed

### Configuration

Several files will be created

**/etc/pki/CA/newcerts/ca.conf**

default data for certificate creation

**/etc/pki/CA/private/cacert.pem**

CA keypair and certificate for self signing

**/etc/pki/CA/certs/ca.crt**

public certificate of self signing CA

**/etc/pki/CA/<servername>.conf**

certificate data

**/etc/pki/CA/private/<servername>.key**

per server private keys

**/etc/pki/CA/certs/<servername>.crt**

per server public certificates

### ini Variables

#### certs

comma separated list of servernames for which self signed certificates will be created:

```
certs=my.gawati.org
```

## Postinstallers

No postinstaller available.

## monit

### General

The goal of the module is to install and configure monit monitoring service for Gawati and the server its running on.

### Software Packages

monit will be installed

### Configuration

Monit will be configured to start on boot, monitoring items will be configured in /etc/monit.d , logfiles created and a monitoring target file created in Apache webroot to provide a test URL

### ini Variables

#### options

Define the monitoring targets to activate:

```
options=apache,base,chrony,email,eXist-st,fail2ban,sshd,startup,system,webinterface
```

Define a notification email address to receive alerts:

```
mailrecipient=root@my.gawati.local
```

## Postinstallers

No postinstaller available.

## template

### General

The goal of the module is to provide a demonstration of the installer environment that can be used as a reference and for introduction to the installer. Many installer commands and variables are used.

### Software Packages

No packages to install

### Configuration

No configurations applied

### **ini Variables**

None

### **Postinstallers**

1 postinstaller available called “postdemo” to show how to invoke it.

## CHAPTER 6

---

### Indices and tables

---

- [genindex](#)
- [modindex](#)
- [search](#)

This documentation is generated using [Sphinx](#) from source written in [reStructuredText](#) and maintained on [GitHub](#).