# Content Security Quick Start

## Quick Start

The following exercise builds on the Attivio Quick Start Tutorial to introduce the concept of content security. We'll extend our example project by ingesting content which only specific users will have permission to see.

### Watch This Tutorial

### 1. Deploy the Attivio Factbook Project

Open the Attivio Quick Start Tutorial in a new tab in your browser, follow the instructions to deploy the Factbook project, and return to this page when complete.

### 2. Create users

When we created the Factbook project, we included the **demo** group of modules (`-g demo`) which includes the **security** module. Attivio's **security** module provides the basic framework for content security. This framework includes the ability to define users and groups, the ability to attach access control lists (ACLs) to documents during ingestion, and the ability to apply that access control information to the search user and return only those documents to which the user has access. These general abilities are provided by fully-configurable components.

In order to demonstrate the ability to return only those documents a user is entitled to see, we must first create some user credentials and configure the authentication of Attivio. For simplicity, this tutorial will utilize XML-based authentication, but a more common implementation is Active Directory and LDAP Configuration.

Execute the following steps to create users into Attivio. Afterwards, we'll be able to log into Search UI as each user and verify that each user can only access the appropriate content.

**Step 1**

Create a file named **`users.xml`** with the contents below and save it to **`<project-dir>\resources\users.xml`** .

```
<?xml version="1.0" encoding="UTF-8"?>
<principals>
    <user id="user1@test" name="user1" password="password1"/>
    <user id="user2@test" name="user2" password="password2"/>
</principals>
```

**Step 2**

Create a file named **`default-users-authentication-provider.xml`** with the contents below and save it to **`<project-dir>\conf\bean\`** .

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns="http://www.springframework.org/schema
/beans" xmlns:util="http://www.springframework.org/schema/util" xmlns:sec="http://www.springframework.org/schema
/security" xsi:schemaLocation="http://www.springframework.org/schema/beans http://www.springframework.org
/schema/beans/spring-beans.xsd http://www.springframework.org/schema/util http://www.springframework.org/schema
/security/spring-security-3.1.xsd">
    <bean name="default-users-authentication-provider" class="com.attivio.security.authentication.
XmlBasedAuthenticationProvider">
        <property name="xmlFile" value="users.xml"/>
    </bean>
</beans>
```

**Step 3**

Edit the file **`<project-dir>\conf\features\core\DeployWebapp._searchui.xml`** and replace its contents with the following:

```
<?xml version="1.0" encoding="UTF-8"?>
<ff:features xmlns:ff="http://www.attivio.com/configuration/config" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance" xmlns:fbase="http://www.attivio.com/configuration/features/base" xmlns:f="http://www.attivio.com
/configuration/features/core" xsi:schemaLocation="http://www.attivio.com/configuration/config http://www.
attivio.com/configuration/config.xsd http://www.attivio.com/configuration/features/base http://www.attivio.com
/configuration/features/baseFeatures.xsd http://www.attivio.com/configuration/features/core http://www.attivio.
com/configuration/features/coreFeatures.xsd">
  <f:deployWebapp authentication-provider-ref="default-users-authentication-provider" context-path="/searchui"
directory="webapps/searchui" enabled="true" featureNameSource="contextPath" nodeset="*" requires-
authentication="true">
    <f:menuEntry blank="true" label="Search UI" path="search" uri="/searchui"/>
  </f:deployWebapp>
</ff:features>
```

Save and close the file.

**Step 4**

Edit the file **`<project-dir>\conf\features\core\DeployWebapp._rest.xml`** and replace its contents with the following:

```
<?xml version="1.0" encoding="UTF-8"?>
<ff:features xmlns:ff="http://www.attivio.com/configuration/config" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance" xmlns:fbase="http://www.attivio.com/configuration/features/base" xmlns:f="http://www.attivio.com
/configuration/features/core" xsi:schemaLocation="http://www.attivio.com/configuration/config http://www.
attivio.com/configuration/config.xsd http://www.attivio.com/configuration/features/base http://www.attivio.com
/configuration/features/baseFeatures.xsd http://www.attivio.com/configuration/features/core http://www.attivio.
com/configuration/features/coreFeatures.xsd">
  <f:deployWebapp authentication-provider-ref="default-users-authentication-provider" context-path="/rest"
directory="webapps/rest" enabled="true" featureNameSource="contextPath" nodeset="*" requires-authentication="
true">
    <f:menuEntry label="Debug Search" name="debug-search" path="search" uri="/rest/search"/>
  </f:deployWebapp>
</ff:features>
```

Save and close the file.

**Step 5**

Edit the file **`<project-dir>\conf\features\security\ContentSecurity.contentSecurity.xml`** and set **`allowUnsecuredDocuments`** to **`false`**:

## 3. Ingest Principals

In order for content security to work, there are three required parts: 1) the content (or documents), 2) the ACLs and 3) the users and groups. In this example, we have only users, no groups. Working with groups is not very different from working with users. You can learn more about groups in the Content Security guide. All of these parts must be in the Attivio index. So let's first ingest the users, also known as **principals** in the Attivio Security Model. Because our users are stored in an XML file, we'll use an **XMLScanner**.

**Step 1**

Go to http://localhost:17000/admin/connectors to open the Attivio Admin UI.

**Step 2**

Click the **+New** button to create a new connector.

Search for and select **XML Files**. Then press **OK**

**Step 3**

The New Connector dialog will open up. Enter values for the following fields:

- **Name**: users
- **Document Root**: /principals/*
- **ID Path**: @id
- **Start Directory**: resources/users.xml
- **Ingest Workflow**: ingestXmlPrincipals

**Step 4**

Click **Save & Test**

The Save & Test feature allows you preview documents that would be fed in by the connector. Click on either document id in the left panel and that document's details will be displayed to the right.

Click **OK**

Click **Save**

**Step 5**

Run the connector by selecting it and clicking the **▶Start** button in the top toolbar.

Click Start in the confirmation dialog window.

Once the connector finishes, you will see the number of documents it has ingested.

**Step 6**

To confirm that our users have been ingested, go to http://localhost:17000/rest/search to open the Debug Search interface.

Change the **Workflows** value to **searchPrincipals** and click **Search**. You should see a response similar to the following:

```
{
  totalHits: 2,
  totalTime: 6,
  facets: [

  ],
  documents: [
    {
      fields: {
        .score: [
          0
```

```
<?xml version="1.0" encoding="UTF-8"?>
<ff:features xmlns:ff="http://www.attivio.com/configuration/config" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance" xmlns:security="http://www.attivio.com/configuration/features/security" xsi:schemaLocation="
http://www.attivio.com/configuration/config http://www.attivio.com/configuration/config.xsd http://www.attivio.
com/configuration/features/security http://www.attivio.com/configuration/features/securityFeatures.xsd">
  <security:contentSecurity allowUnsecuredDocuments="false" class="com.attivio.security.model.
FullSecurityModelImpl" enabled="true" name="contentSecurity" secureResponseWorkflow="defaultResponse"
securedIndex="index"/>
</ff:features>
```

Save and close the file.

**Step 6**

Open the CLI and execute the **deploy** command.

```
<install-dir>\bin\aie-cli.exe -p C:\attivio\projects\Factbook

aie> deploy
```

Once the system restarts, you should be able to log into Search UI using either of the name/password combinations in the **users.xml** file as well as the default administrative **aieadmin** user.

You may notice Search UI reports that there are 0 documents in the index even if you had previously ingested documents. This is expected. The last configuration change we made above told Attivio not to return any documents which do not have any ACLs on them. Any previously-ingested content would not satisfy this requirement. Fear not, we'll soon ingest more content, this time with ACLs.

```
        ],
        .id: [
          "prin:test:user1"
        ],
        .zone: [
          "security/principals"
        ],
        table: [
          "AIE.principals"
        ],
        parentid: [
          "user1@test"
        ],
        ancestorids: [
          "user1@test"
        ],
        security.realmid: [
          "test"
        ],
        security.principalid: [
          "user1"
        ],
        security.recordsetid: [
          "test:user1"
        ],
        security.principaltype: [
          "user"
        ],
        security.principalname: [
          "user1"
        ],
        security.principal.guid: [
          "test:user1"
        ],
        security.principal.subject.guid: [
          "test:user1"
        ],
        security.principal.object.guid: [
          "test:user1"
        ]
      }
    },
    {
      fields: {
        .score: [
          0
        ],
        .id: [
          "prin:test:user2"
        ],
        .zone: [
          "security/principals"
        ],
        table: [
          "AIE.principals"
        ],
        parentid: [
          "user2@test"
        ],
        ancestorids: [
          "user2@test"
        ],
        security.realmid: [
          "test"
        ],
        security.principalid: [
          "user2"
        ],
        security.recordsetid: [
          "test:user2"
        ],
        security.principaltype: [
          "user"
        ],
        security.principalname: [
          "user2"
        ],
        security.principal.guid: [
          "test:user2"
        ],
        security.principal.subject.guid: [
          "test:user2"
        ],
        security.principal.object.guid: [
          "test:user2"
        ]
      }
    }
  ]
}
```

## 4. Create secured documents

There are many Attivio connectors which are capable of feeding ACLs along with documents. Some examples include Sharepoint, Exchange, Windows file scanners, JIRA and Confluence. For the sake of this tutorial we'll push some custom content in using Attivio's REST IngestAPI. We'll do this using a tool named Postman. Postman is a handy utility which allows users to make HTTP POST requests and provides a convenient way to explore RESTful web services.

**Step 1**

Download and install Postman

**Step 2**

First, connect to the Ingest API:

1. Create a new **GET** request with the URL of http://localhost:17000/rest/ingestApi/connect
2. On the **Authorization** tab, select **Basic Auth** from the **Type** menu
3. Set Username to **aieadmin** and password to **attivio**
4. Click **Update Request**
5. Click **Send**
6. You should receive a 200 HTTP status code along with a session id such as **`91c878fc-4898-4e00-b860-e45ea5a9236c`**
7. Make note of this session id, it will be required for the next steps

**Step 3**

1. Create a new POST request with the URL of **`http://localhost:17000/rest/ingestApi/feedDocuments/<session id from the previous step>`**
2. On the **Authorization** tab, select **Basic Auth** from the **Type** menu
3. Set Username to **aieadmin** and password to **attivio**
4. Click **Update Request**
5. On the **Body** tab, ensure that **raw** is selected along with **JSON (application/json)** from the drop-down
6. Paste the following into the body:

```
[ {
  "fields" : {
    "title" : [ "This is content for User 1" ],
    "date" : [ "2018-01-01" ],
    "table" : [ "rest" ]
  },
  "id" : "restricted-1",
  "correlationId" : null,
  "mode" : "ADD",
  "permissions" : [ {
    "principal" : {
      "name" : "user1",
      "realm" : "test",
      "type" : "user"
    },
    "readable" : true
  } ]
}, {
  "fields" : {
    "title" : [ "This is content for User 2" ],
    "date" : [ "2018-02-01" ],
    "table" : [ "rest" ]
  },
  "id" : "2",
  "correlationId" : null,
  "mode" : "ADD",
  "permissions" : [ {
    "principal" : {
      "name" : "user2",
      "realm" : "test",
      "type" : "user"
    },
    "readable" : true
  } ]
} ]
```

**Step 4**

1. Create a new **GET** request with the URL of **`http://localhost:17000/rest/ingestApi/commit/<session id from the previous step>`**
2. On the **Authorization** tab, select **Basic Auth** from the **Type** menu
3. Set Username to **aieadmin** and password to **attivio**
4. Click **Update Request**
5. Click **Send**
6. You should receive a 200 HTTP status code

## 4. Create secured documents (without Postman)

If you are working in an environment where you are unable to install Postman, you can use the following method to create the secured documents.

**Step 1**

Download the attached external module: createdocs-0.1.0-SNAPSHOT.zip to a temporary location such as C:\temp.

**Step 2**

Install the external module by executing the following command:

```
<install-dir>bin\aie-exec.exe modulemanager -i file:///c:/temp/createdocs-0.1.0-SNAPSHOT.zip
```

**Step 3**

Open the CLI and execute the **`deploy`** command.

```
<install-dir>\bin\aie-cli.exe -p C:\attivio\projects\Factbook

aie> deploy
```

Wait for the system to restart.

**Step 4**

Execute the following command:

```
 <install-dir>bin\aie-exec.exe createsecuredcontent
```

## 5. Test the results

**Step 1**

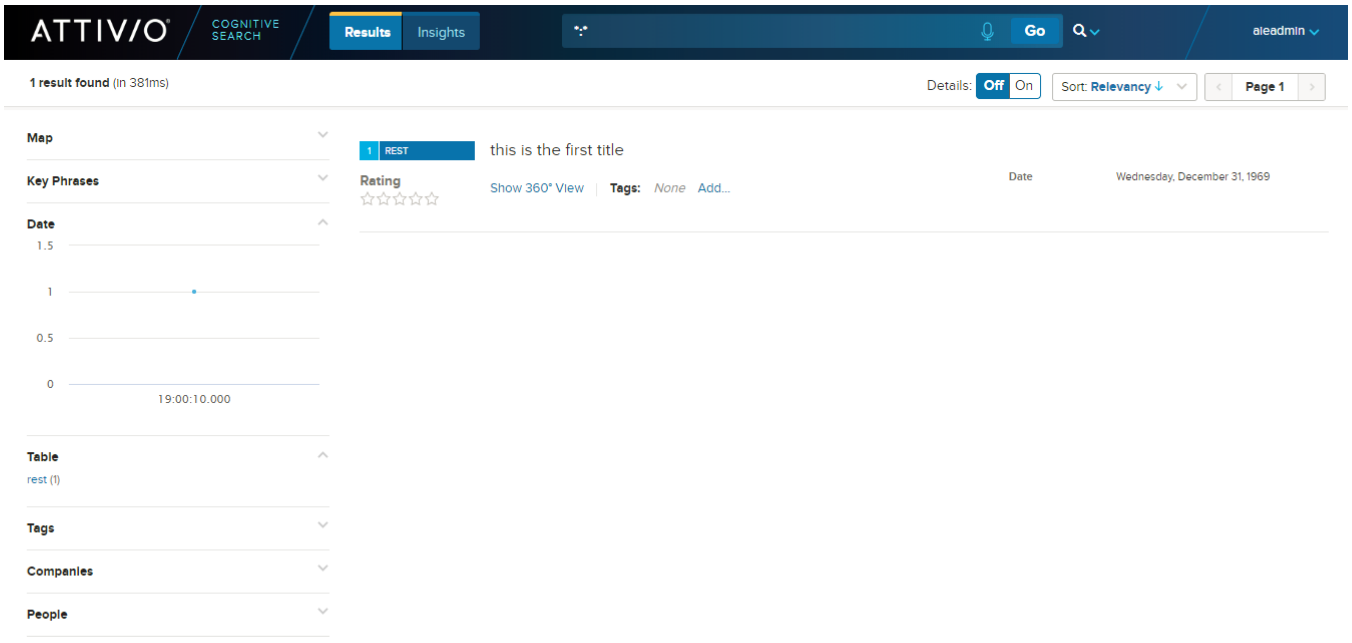
Open a new incognito window to http://localhost:17000/searchui/ and log in with **`user1/password1`**

**Step 2**

Click the **Go** button to submit the default **`*:*`** query to view all documents in the index.

**Step 3**

You should see the document we created with read permissions for User 1.



1 result found (in 381ms)

1 REST this is the first title

Rating

Show 360° View | Tags: None Add...

Date Wednesday, December 31, 1969

Table: rest (1)

**Step 4**

Repeat steps 1 - 3 using the credentials for **`user2`**.

To completely log out of Search UI, use the the Log Out link on a page of the Admin UI such as http://localhost:17000/rest/search or close all browser windows.

## Summary

In this tutorial, we've seen how Attivio can ingest user records, and access control lists (ACLs) along with the usual documents and data. As a result, the index engine can filter not only by query criteria but also by access criteria.  We've only touched on the basics of the Attivio Security Model. We could also create groups of users and show how users who are a member of a group can access content permissioned for the group rather than individual users. We could also allow users to access the appropriate content from disparate source, which use different security domains by creating aliases to associate multiple principals that represent the same user.

## What Next?

Now that we've covered the basics of content security, you may be interested in the following:

- Learn more about the Content Security.
- Learn about Active Directory and LDAP Configuration.
- Read about security all aspects of Attivio, including web applications and endpoints in the Security Guide.
- Learn more about the REST Ingest API.