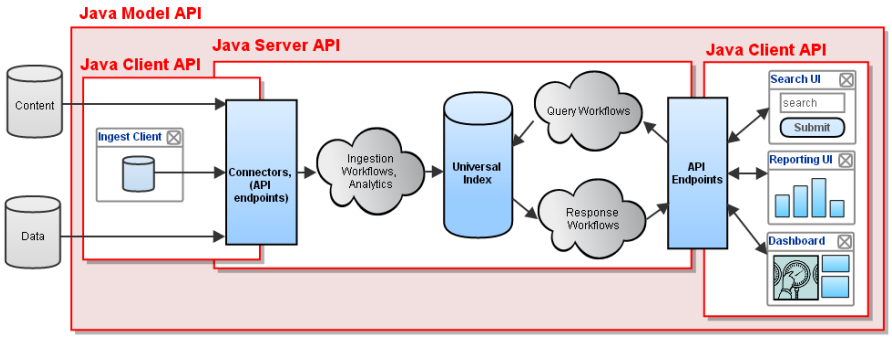
Using Java APIs

Overview

AIE's capabilities can be extended using the Java Server API, the Java Client API, and the Java Model API libraries.

- The Java Server API provides tools primarily for creating and modifying connectors, workflows, and components on both the ingestion and querying sides of the architecture.
- The Java Client API provides tools for feeding documents and submitting queries from external Java applications.
- The Java Model API provides many supporting classes that are used in both the server and client libraries.



AIE Designer is a turnkey Eclipse-based development environment for extending AIE through the Java APIs. Projects created through Designer (or through the createproject command-line utility) include a complete set of Eclipse project files.

You can also configure the project to include Java API sample code demonstrating many of the features of the Server and Client APIs. This makes it easier to write custom client applications, transformers and services. The examples also include abstract test classes, a sample unit test for a transformer, and a sample integration test. You can run everything directly from Designer.

This page is primarily about setting up a project for AIE development in Designer.

View incoming links. **>>**

Create a New Project , Creating Custom Response Transformers , Creating Custom Scanners , Developing in AIE - Concepts and Tools , Developing with Attivio , Ingest Application Example , Search Application Example , Streaming Query API

Software Requirements

To set up AIE for customization/extension using the Java APIs, simply include the AIE Designer in the installation. (Designer requires an add-on installer as it is not included included in the default installer)

Creating the Project

In the examples that follow, AIE is installed in the **C:\attivio** directory, which is referred to as **<install-dir>**. The project is called **plusjava** (meaning "factbook plus java"), and uses **C:\attivio-projects\plusjava** as the project directory. This is referred to as the **<project-dir>** in the examples.

To create the example project using AIE Designer, follow these steps:

Step 1

Be sure that AIE Agent is running on the computer where AIE is installed. It might be running as a service, or you can manually it.

Command Windows or Shell

```
<install-dir>\bin>aie-agent -p 16999
```

Step 2

Start AIE Designer. If you don't have a desktop icon for this purpose, you can start it manually.

Command Window or Shell

```
<install-dir>\designer\Designer
```

Card 3

If this is your first time using Designer it will ask you to designate a workspace (**C:\workspace)** then Click **Use Designer** to bypass the **Welcome to Designer** page.

Step 4

Open the **AIE Configuration** menu, and select **New AIE Project** then in the **New AIE Project** dialog box, give the project a name. We used "plusjava". Click **Next**.

Step 5

Add the necessary modules to the project. For this exercise we selected the **Demo** group and the **factbook** module then check the box labeled **Include sample code in the project**. Click **Finish**.

Testing the Software Installation

After generating a project, how do you know that everything has been set up correctly for Java API development?

The AIE project includes several example files, some of which can be run from Designer as JUnit tests. If the example file compiles and runs, then the environment has been set up correctly.

To verify that the environment is set up and functioning, run the demo integration test example.

SampleIntegrationTest.java

Before running this test, be sure that the ATTIVIO_HOME environment variable is set to the AIE installation directory.

Note that AIE does not need to be running for this test. The test will start and stop its own AIE server.

1. In the Designer **Package Explorer** view, access the **test/com.acme** directory.
2. Right-click on **SampleIntegrationTest.java**. From the pop-up menu, select **Run As > Junit Test**.

This is a simple test that runs a single IngestDocument through the ingestion process. It displays several screenfuls of status messages. The following lines indicate a successful test.

Console View

```
Got result OK from indexer.index-content-dispatcher2014-10-08 14:45:59,474 INFO  ContentFeeder - client
3800@192.168.6.102_3fc1e470-7044-469d-abf7-c21a7a004a5f waiting for message results, timeout=-1 workflows=[wf=
[ingest], rem=false, ri=false]
 for doc 1234
Got result OK from sink.sink for doc 1234
```

Creating Custom Code

See the Java Server API and Java Client API pages for detailed examples showing how to write and integrate customized Java components in your project.