Glencore Development Notes

Jonny Coombes

November 2020

Contents

Chapter 1

General Notes

1.1 Versioning

The general versioning scheme for Argon builds will comprise of three main components:

1. The *major* version number.
2. The *minor* version number.
3. Either:
   (a) A specific build identifier (monotonically increasing).
   (b) A specific patch/correlated fix identifier.

1.2 Source Control

The Git versioning scheme for Argon is straightforward:

1. The main development branch is *mainline*.
2. Each intermediate release will have a specific branch, named in accordance with the release. Currently planned intermediate releases are as follows:
   (a) 0.1.0 - build after the initial development sprint.
   (b) 0.2.0 - build after the second development sprint.
   (c) 0.3.0 - build after the third development sprint.
3. Individual feature implementations will be carried out on a dedicated branch, prefixed with the corresponding **Jira** ticket. For example, ticket number **JA-15** would have a branch named *feature*
   JA-15-Summary, where the summary is automatically generated as part of the development toolchain.
4. Within the local development environment, changes are mastered and then pushed to multiple remotes. (There may be multiple remotes based on the number of environments stood up).

1.3 Build Dependencies

The key libraries used throughout the build of the Argon project as given in table 1.1 below:

Library	Version	Description
.NET Core	5.0	Core .NET platform runtime
ASP.NET	5.0	ASP.NET Core library
EF Core	5.0	EF framework (plus RDBMS specifics)
Serilog	2.10.0	Logging library
Serilog.Sinks.Console	3.1.1	Console sink for Serilog
Polly	X.X	Policy library

Table 1.1: Key Argon Dependencies

1.4 Database Notes

General DB notes:

1. **Development SQL Server Version (Ryleh)**: Microsoft SQL Server 2019 (RTM) - 15.0.2000.5 (X64) (Sep 24 2019 13:48:23)
2. **Development Argon Login Name** : *argon*
3. **Development Argon Db Name**: *argon*
4. **Core Schema Name**: *core*

The nominated user used to connect through via the DB context, must be a member of the following SQL Server roles:

1. dbcreator

This allows for just a user login to be allocated on the target SQL instance, and then Argon can take care of creating the necessary database/schema objects. (*This adheres to the principal that configuration should be a close to zero as possible for new deployments*).

1.4.1 Core Tables

Schema	Table	Description
core	collection	The main collections table, one row per collection.
core	constraint	The collection constraints table. $1 \rightarrow *$ relationship between entries in the *collection* table and entries within this table.
core	document	The collection documents table. $1 \rightarrow +$ relationship between the *collection* table and entries within this table.

Table 1.2: Core Tables

1.4.2 EF Core Specifics

Some general notes on the EF Core implementation within Argon:

1. All useful model entities will utilise a *Timestamp* as a concurrency token.

1.5 AWS Deployments

Current AWS deployment details in table 1.3 below:

Component	Type	Name	Details
VPC	Virtual Private Cloud	Argon-Dev-VPC	Partitioned VPC
Subnet	VPC Subnet	Argon-Dev-SN-1	10.0.0.0/24 CIDR
Sec. Group	Security Group	Argon-Dev-SG-1	SSH, HTTPS, SQL (Restricted)
Internet Gateway	(E/I) Gateway	Argon-Dev-IG	Ingress and outgress
EIP	Elastic IP	No public DNS yet	34.249.105.124
Host	t3a.small	argon-dev-1	Ubuntu 18.04 LTS

Table 1.3: AWS Deployment Artifacts

1.6 Code Notes

1.6.1 Namespaces

The general layout of the Argon core code adheres to the following conventions:

1. The top level namespace for the core is **JCS.Argon**
2. Key code artifacts are organised so that:
   (a) Controllers are placed in the **JCS.Argon.Controllers** namespace.
   (b) Services are placed in the **JCS.Argon.Services** namespace.
   (c) Model elements have a top-level namespace of **JCS.Argon.Model**.

1.6.2 OpenAPI Endpoints

The OpenAPI specification that the API conforms to is published at the following location (relative to the deployment root):

```
/swagger/v1/swagger.json
```

The version path component is expected to remain constant during the initial release, and will only change with *significant* breaking change releases in the future. All initial API endpoints will be prefixed as follows:

```
http[s]://[host]:[port]/api/v1
```