1 Introduction

Introduction To the Metadata Aggregator

This section provides a general overview of the Metadata Aggregator product. It describes the use cases that initially prompted the development of the aggregator and provides a high-level overview of the concepts and architecture of the product. This section does not contain any configuration information but should be read before proceeding on to the installation and configuration documentation.

Initial Aggregator Use Cases

The following use cases are what prompted the development of the tool (note, while all the uses cases given below are SAML-focused the general aggregator product is not SAML-focused).

- A command line tool that can read in a bunch of SAML entity descriptors, schema validate them, check certain policy constraints, filter out entities based on given rules, assemble the entities into multiple different SAML metadata documents, and sign them all.
- A command line tool and web service that can perform the technical work of inter-federation. This essentially means doing everything in the previous use case but operating on whole metadata documents from various federations.
- A web service that can operate as a metadata "oracle" for IdPs and SPs and hide the complexity of fetching, validating, and munging (potentially many) metadata sources. This service could be deployed within an organization and the organization's IdPs and SPs could delegate most of the hard work of dealing with metadata to the oracle, making their configuration simpler, decreasing network load, and potentially increasing resiliency in cases where external metadata may go offline.

The aggregator itself is a fairly general tool and so it's quite likely that as adoption grows people will find other uses for it as well. It is certainly the goal of the developers to make it relatively easy to meet any use case with a general form of "read in a bunch of data, transform it in various ways, and write it out or search it".

Metadata Aggregator Architecture

The following sections give a high level overview of the architecture of the metadata aggregator as well as introduce terms that will be used throughout the rest of the document.

Core Concept: Items

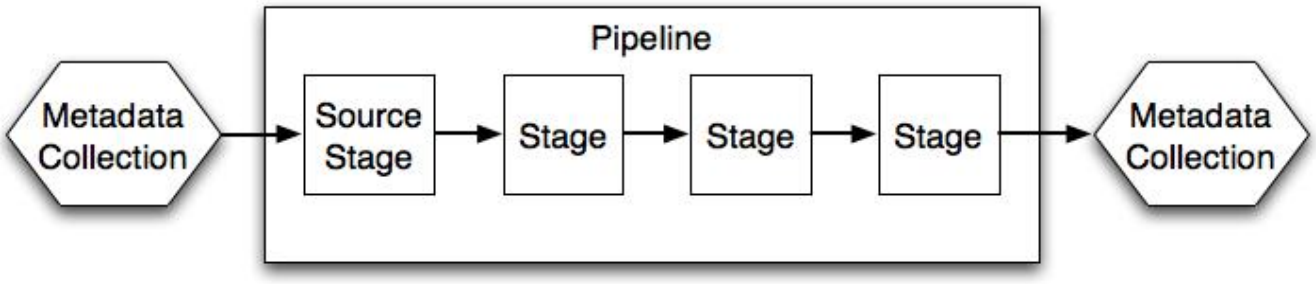
Within the aggregator, each individual unit of data is known as an *item*. As the name should suggest, this is a fairly generic construct. An item might represent anything: a person, a group, SAML entity. The data itself may also be encoded in any form: XML, JSON, ASN.1.

In addition to wrapping a unit of data, an item also carries a set of metadata about the item. This metadata is attached to the item as its processed by the aggregator and can really be any computed information that applies to the item. As will be seen later on, this includes things like identifiers by which the item may be looked up, error and warning messages if the item fails to pass various checks, and provenance information showing what parts of the aggregator worked on the item.

Core Concept: Pipeline

At the center of the metadata aggregator is the concept of a processing pipeline. A *pipeline* is a component which passes a collection of items through a number of *stages* which may transform, remove, add, or otherwise modify the collection.

In most cases, the first stage in a pipeline will be the *source stage*, or just source for short. Source stages read in data from somewhere (e.g., files on the filesystem, HTTP accessed file, configuration files), construct items, and populate the collection with them. While source stages normally occur at the beginning of the pipeline they are in fact just normal stages and so can actually occur anywhere in the pipeline flow.



As an example, imagine a *pipeline* whose *source stage* is an XML document pulled in over HTTP. The XML document is then passed through a set of *stages* that schema validate the document, check its digital signature, and then passes it through an XSLT transformation. In such a case the resultant collection would have a single entry. If a stage in the pipeline had broken the XML document up in to multiple documents then the resultant collection could have had more than one entry.

The configuration will discuss all the available stages and will give concrete examples (with configuration snippets).

Command Line

One method of using the metadata aggregator is as a command line tool. In this usage mode a primary pipeline is run and the result is (usually) written to a file. This is useful for integrating with a larger data processing environment, doing one-off processing, and testing the pipelines used in the web service.

Web Service

> ⚠ The web service component is not currently included in the Project Roadmap.

The web service interface provides a simple HTTP (REST, if you want to play buzzword bingo) interface that allows a consumer to retrieve one, or more, elements from an item collection based on an identifier or tag.

The web service is built of:

- a *generative pipeline* that is responsible for reading in all the data and creating the item collection that can be searched
- an *item store* that is responsible for holding on to, indexing, and querying the item collection created by the generative pipeline
- a *result processing pipeline* which is responsible for performing per-query work to prepare the query results to be sent back to the requester (e.g., merging multiple items in to a single document, signing the result, etc.)
- a *query controller* which receives query requests, gets queries the item store, runs the results through the result processing pipeline, and sends the response back (i.e., this components coordinates the others)