Swedish Alliance for Middleware

Dokument	swami-idmgmt-architecture-notes
Editor	Leif Johansson
Identifier	
Version	0.2
Last Modified	2006-03-29 19:50
Status	draft

The SwAMI identity management architecture

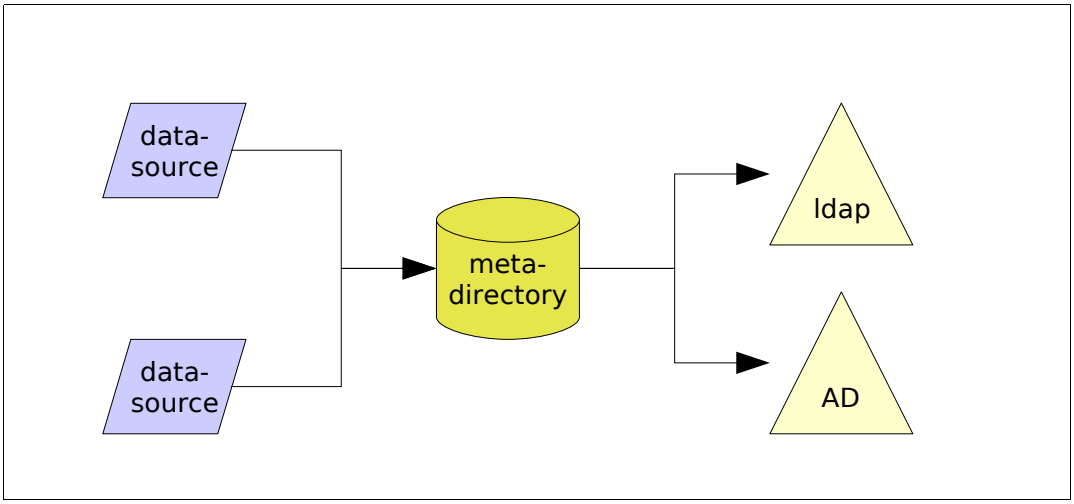
This document describes an architecture for an open identity management system for SwAMI. The goal is to identify the components of the system and the way these components interact with each other and with other services. This document could serve as basis for a set of specifications for the components and communication protocols.

This document is the outcome of a workshop on directory services held on 14/3 2006 at Stockholm university and represents a rough consensus view of identity management systems and requirements based on experiences from the members of SwAMI.

An identity management system is a system which aids in the management of electronic identities of entities and services. In this document we will extend this notion to include management of all types of directory objects (and attributes), eg roles, groups, courses, organizations, devices or services.

Identity managment then becomes the process by which attributes about such objects are maintained, including synchronization with external datasources. A system or service supporting identity-management in this broader sense is also called a meta-directory.

A common way to build a metadirectory is to use the registry model.



Some form of datastore is used to create a merged view of objects in all datasources. This common datastore is then used to export one or several application directories. This database sometimes goes by the name meta-directory but this is a misnomer since the entire system including import and export is the meta-directory. We will call the datastore a 'registry' for reasons will become clear later on.

The precise way in which the datastore is a directory (as in meta-*directory*) is by virtue of RFC3254: ”Definitions for talking about directories”, H. Alvestrand (INFORMATIONAL). In section 2.6 of that document a 'directory' is defined as a repository with search capability (defined in 2.2) and convergent consistency.

Convergent consistency roughly means that single updates may take different amounts of time to propagate through the (meta-)directory but will eventually reach all destinations. Put another way: there is no concept of transaction protecting a single update or set of updates which could guarantee that (for instance) all instances of an application directory always look alike.

The registry model above doesn't specify how updates get from the left-hand side of the diagram to the right-hand side. In many cases datasources are queried using dump-scripts which is fed into the registry. Similarly, output to application directories is also often done using dump-scripts.

The problem with this approach (and the reason why most SwAMI members have or is in the process of moving away from it) is that it doesn't scale very well. For instance a common datasource in the SwAMI community is LADOK where traversing 100's of thousands of entries is the rule rather than the exception.

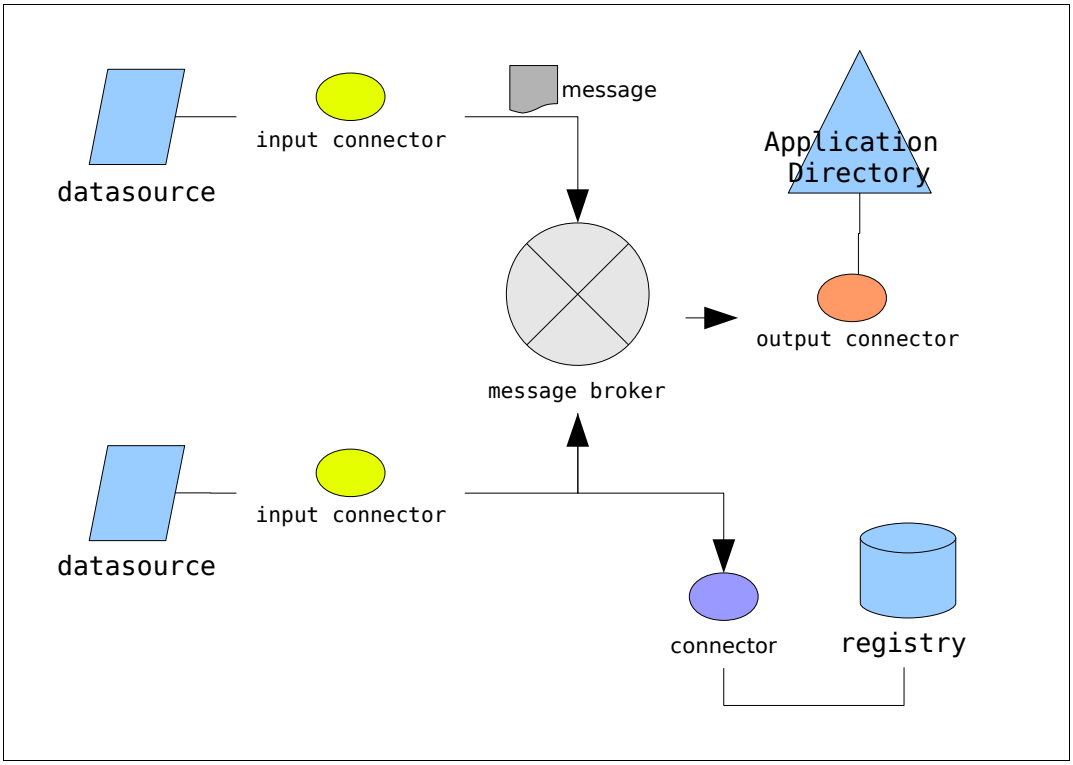
The solution to the scalability problem is of course event-driven data import and export. In traditional relational database technology this corresponds to using triggers to effect event-driven modifications to the database. In our case the use of triggers, although a useful tool for certain datasources, cannot form the basis for an open architecture.

A viable approach is to identify event-management as an architectural component and treat it as a first-order citizen in the meta-directory. The fundamental nature of event-management is that it is a message-oriented process. Message-passing is best modelled using the concept of a message-broker, i.e a service which is responsible for routing and adressing of messages between message producers and consumers.

The message-broker is beginning to gain popularity in the IT business as a design-component with the increasing momentum of SOA (service oriented architecture) and EDA (event driven architecture).

During the recent workshop on directory services it became clear that several members had already identified and in many cases implemented a message-broker either as a general service or as part of a meta-directory system.

This gives us the following updated version of the meta-directory model:



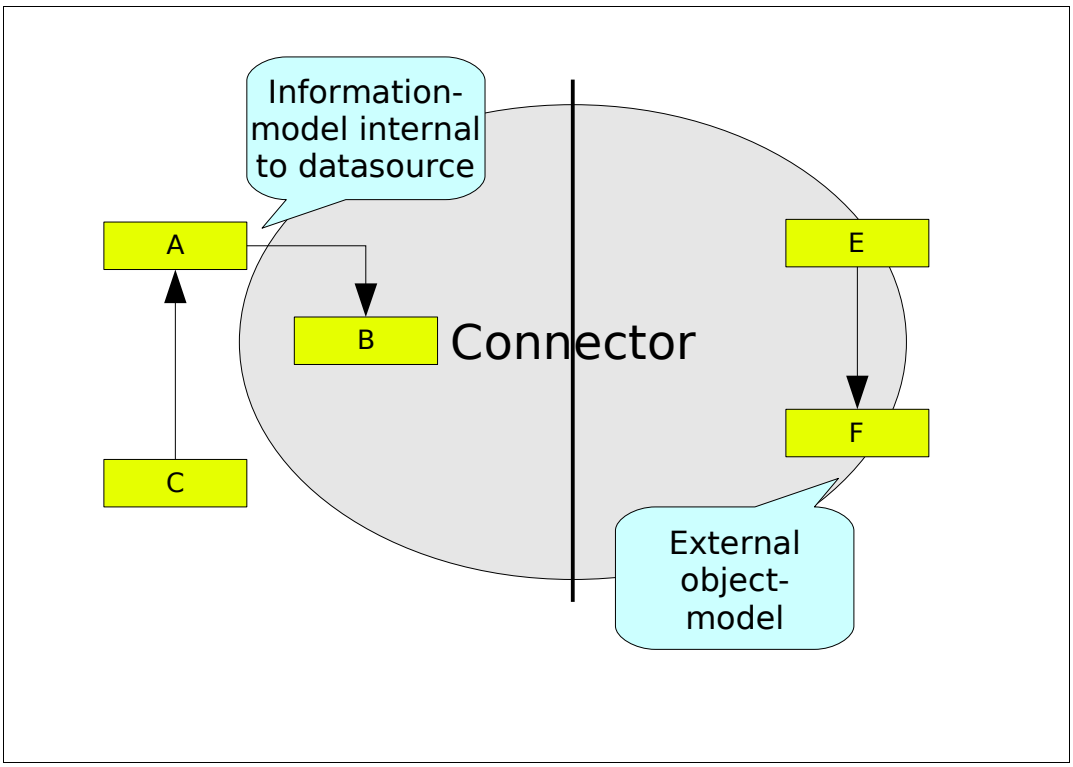
The move to an event-driven architechture (EDA) makes it necessary to create intermediary services (connectors) between the datasource and the message broker. The connector is responsible for the application-specific interface between the datasource (a HR system for instance) and the message-broker.

In this model connectors can be either message producers (aka sources) or message consumers (aka sinks). An extension to the model allows for connectors which acts as transformers of messages by both consuming and producing messages.

Connectors serve another important function: object model abstraction. The datasource may (or may not) represent datastores or systems with a clear object model.

In either case the local model may not be suitable as is for the purpose of building application directories. For instance the LADOK information model is not object oriented and contains much more information than is useful to the LDAP directory.

This is where the connector comes in. The messages represent serializations of objects (or modifications to objects) using the abstract object model presented by the connector. There are several possible choices for object serialization compatible with asynchronous messages-oriented protocols.



Several commercial meta-directory and identity-management systems have an internal architechture which resembles this model. The SwAMI identity management infrastructure goes one step beoyond this by opening up the internal architecture through the use of open protocols.

Connectors can both consume and produce messages, and could also contain synchronous RPC interfaces for administrative purpouses. A connector to a system which produces messages based on events in the backend system may need to consume messages in order to trigger events for the purpouse of recovery and state synchronization.

The messages carried in the system are data-oriented rather than process oriented. This means that messages will represent information about modifications to, or the current state of, an object in a datasource but will not (in general) represent a remote procedure call (RPC). Put another way the messages are descriptive rather than proscriptive.

Descriptive messages carry an implied semantics of ”synchronize to this state” which the sink connectors must honor. In order to fulfill the requirement of convergent consistency for the registry and the application directories the connectors should try to follow the principle of idempotence. Idempotence states that two consecutive identical operations produces the same result as one of them. This means that a source connector can retransmit a message wo adverse results, which makes for easier error handling for connectors.

It is expected that connectors will need to support some form of discovery and/or metadata mechanism which enable the development of management- and monitoring applications.

The benefit of this model and its acceptance by SwAMI members depends on a few assumptions:

- Connectors are sharable and easy to write
- The message-broker and registry are pluggable
- The communications protocols are based on open standards
- A self-contained reference-implementation is possible
- Components must only depend on published interfaces

These assumptions say that work created by one organization can be reused by others and that it is possible to deploy the architecture with minimal impact on existing infrastructure while it is possible to move towards full integration with existing infrastructure.

It is also important that existing SwAMI members that have systems which resembles this architecture can reuse as much as possible of the components produces by SwAMI for the architecture.

Pluggable message broker and registry implies that a single protocol or a small set of protocols is used. If multiple protocols are specified for the communication between connectors and the message-broker all connectors must either implement all protocols or the message-broker must implement all protocols. For instance if both REST and SOAP is specified then in order for connectors to be shareable all message-brokers must support both REST and SOAP as a way to send messages.

In this model the registry is no different than any other connector and data-source. From the point of view of the identity-management application however the registry is special since we must allow applications to modify objects in the registry. Here is why:

In the identity-management application the management of user identities is one of the most important processes (other processes beeing for instance the management of roles or groups). For SwAMI members most of the users will either be coming from the HR connector or from the LADOK connector, beeing either employees or students.

The problem is that these are not all users. Most SwAMI members have users which do not fall into either of these two categories, for instance guests or consultants. There are two solutions to this problem: either create a separate datastore (and connector) for external users or create objects directly in the registry. Both approaches have benefits.

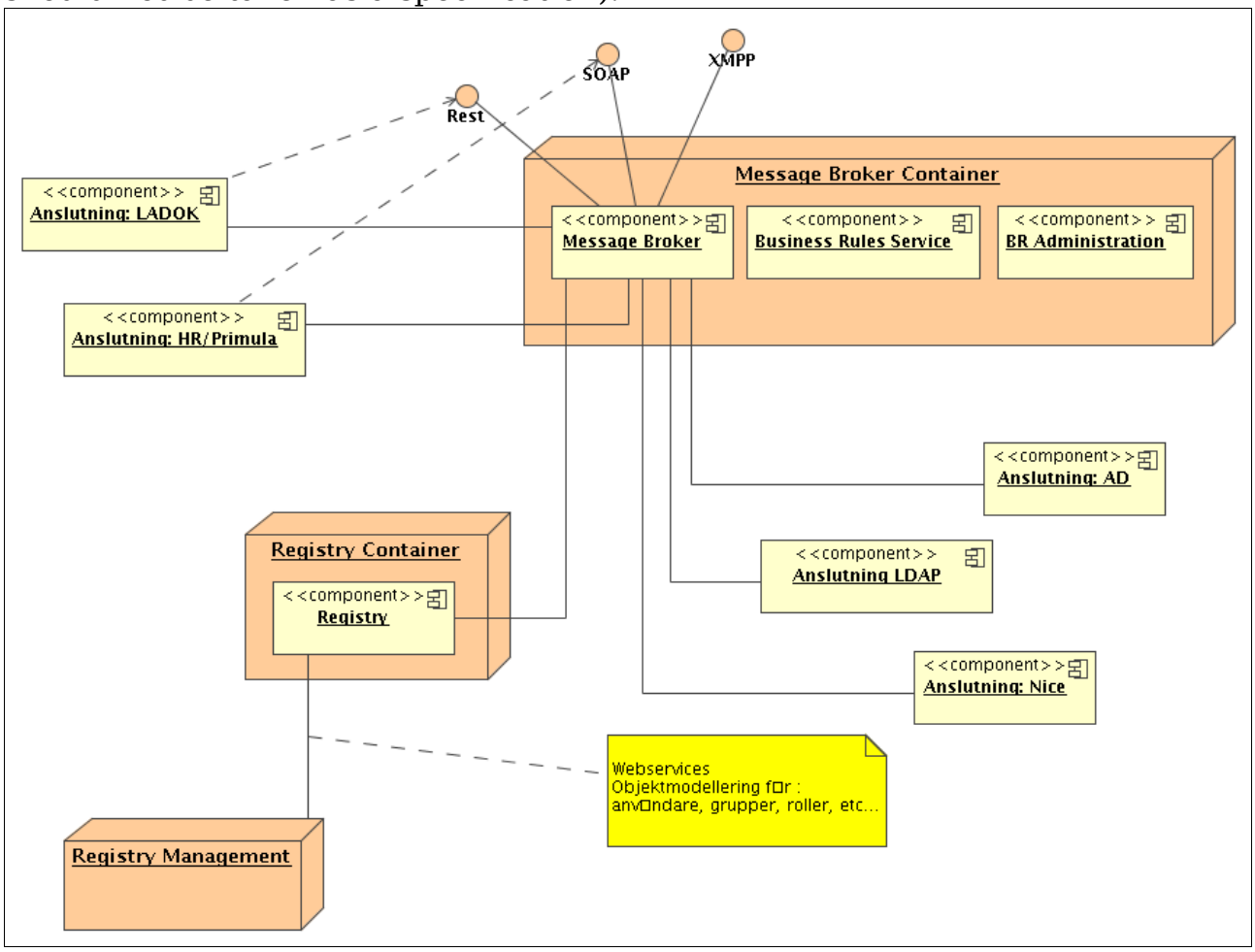
In this model we will assume the latter aproach, mainly since it reduces the number of components in the model. This means objects in the registry must be manageable (webb-interfaces, webb-services, etc). The model could easily be extended to cover the former approach.

Turning our attention to the message broker, several SwAMI members have identified the need for declarative programming in the operation of the message-broker. The message-broker mainly serves as a router. Incoming

messages are routed to one or several destinations based on either static configuration or dyamically based on the contents of the message.

Rule-based (declarative) programming is gaining popularity and has several techical benefits. Therefore this model includes support for a business-rule database used to store and manage the rules used to make routing decisions in the message-broker.

The full model looks like this using UML deployment diagram notation. In this diagram the message-broker is supporting multiple protocols for message-passing (SOAP, Rest and XMPP are only examples of such protocols and should not be taken as a specification).



The TODO-list for turning this architecture into a set of specifications includes at least the following steps:

- Specify the object model for the meta-directory application, including but not limited to:
  - Users
  - Groups
  - People
  - Roles
  - Organizations
  - Devices
  - Services

- ...

Specify the protocol(s) involved

- Specify the object serialization mechanism(s)
- Specify the connector/broker contract
- Specify the connector/registry contract
- Specify the security model
- List and prioritize a set of connectors for common sources and sinks.