Position Statement on Multi-tenancy

First draft, November 4-7, 2011, JGP/MAM

Minor revisions to KK comments, November 10, 2011, MAM

MT--1: Scope / Purpose

Multi-tenancy is a method within software architecture that allows a single instance of an application to serve multiple customers. Each customer, or "tenant," can configure the application as needed or desired. Application data for a given tenant has some degree of isolation from the data of other tenants depending on the chosen type of multi-tenancy. In addition, it may provide for better scalability for service providers and tenants alike.

The High-Level Requirements developed during the ArchivesSpace planning grant\(^1\) identified the following installation and packaging scenarios in its framing principles: packaged/deployable for a single user without a network connection, packaged/deployable for one or more repositories in a networked configuration, and packaged/deployable as a hosted service for multiple communities. These hosting scenarios identified also require the new application to address the following situations:

- **Situation A:** an external hosting provider for a number of independent or partner institutions (e.g. a commercial hosting provider, or a non- or semi-commercial hosting provider such as a library or archives collaborative);

- **Situation B:** a larger organization providing hosting for archival institutions that belong administratively to the larger organization (e.g. a university with many archival repositories or special collections units).

Despite the potential complexity of developing a multi-tenant application, the ArchivesSpace Technical Review Team discussed a number of possible solutions and committed to a specific strategy at their in-person meeting at the University of California San Diego in October 2011. This position statement reviews the discussions and decisions made by the Technical Review Team, and serves as a position statement to prompt additional discussions with both vendors and the user and hosting communities for the ArchivesSpace project.

---

Possible Solutions

The ArchivesSpace team considered three possible solutions for implementing multi-tenancy within the application at the level of the persistence layer for the application.

**Option 1: Separate database instances, separate database schemas**

This approach to multi-tenancy uses a separate database for each tenant. While the application would share application code across all tenants on a particular server, each tenant’s data is logically isolated from all others. Configuration and application-specific metadata associates each database with the appropriate tenant, and security at the level of the persistence layer blocks tenants from accessing or modifying application data owned by another tenant.

While this approach allows for the greatest security and the largest degree of application configurability and extensibility for tenants, this approach is the most costly in terms of maintenance costs for the hosting provider. This approach is most appropriate where individual customers are willing to pay a premium for the added security and configurability. This approach is the current approach used by California Digital Library for its Archivists’ Toolkit and Archon hosting service.

**Option 2: Shared database instances, separate database schemas**

This approach involves hosting multiple tenants within the same database instance, wherein each tenant has a separate schema created for each tenant. Each schema contains its own set of tables. Security at the level of the persistence layer is controlled by access permissions at the schema level. This approach provides a moderate degree of data isolation.

The primary disadvantage of this approach is that data for individual tenants is harder to restore in the event of corruption or failure. In this approach, recovery by restoring an individual database to the server is not possible because it would overwrite data for all tenants within that database. Accordingly, a tenant’s database may have to be restored to a temporary server and then imported to the production server, which is both complicated and time-consuming.

**Option 3: Shared database instances, shared database schemas**

This approach involves all tenants in a given deployment sharing the same set of tables. Within each table, an additional column for a tenant identifier associates each tenant with the data it owns at the row level. This approach is the most cost-effective in terms of ongoing operational costs, such as hardware and backup maintenance. However, the level of configurability for individual tenant instances is much lower. Additional development may be necessary to ensure the application is secure. Data recovery is
similarly complex to that of option 2 (shared instance, separate schema), but adds an additional complication in terms of lower performance because of repeated deletion and reinsertion of individual rows within the production database.

**MT--3: Chosen Multi-Tenancy Approach**

The ArchivesSpace Technical Review Team proposes the use and implementation of Option 3 (shared database instance, shared database schema). In terms of security, tenant-to-tenant data isolation is possible through incorporation of access controls and tenant view filters. This solution will also ensure the lowest overall costs to hosting providers. We believe that the level of data isolation provided by this option should be sufficient for most tenants within a given hosting environment. If the level of isolation provided by option 3 does not meet the needs of a given tenant, the tenant could subsequently contract with their hosting provider to deploy a separate ArchivesSpace instance or host their own instance if preferable.

The ArchivesSpace team recognizes that there is a difficult balance between the requirements of the two situations identified in MT--1. In the case of the Situation A, an external hosting provider would want to ensure a balance between data security and flexibility of configuration across tenants. For example, a repository that contracts the hosting of their application with an outside vendor may want to ensure that other repositories will not be able to view data such as accession records. In the case of Situation B, a given institution may need to generate reports across multiple repositories.

Within the context of the ArchivesSpace project, multi-tenancy should be seen as addressing Situation A, as identified in MT--1. In other words, tenants within the application should be viewed as what would be otherwise freestanding instances of the application, and information should not be shared across tenants. If an institution should need to share data between clients within the application, implementers should use the approach where the application hosts more than one repository within one tenant.

**MT--3: Implementation**

*Data Model*

The application will support multi-tenancy starting at the level of the data model. Tenants will be modeled as a subclass of the Agent class. All instances of the application must have at least one tenant declared. A tenant record may have one or many

- Repository records
• User records

Each datum within the ArchivesSpace application, including some system administration data, will belong to one and only one Tenant.

Security

All access to non-public data and privileged operations will be mediated by the ArchivesSpace authorization system.

The ArchivesSpace authorization system will prevent one Tenant from accessing or modifying another Tenant’s data. The ArchivesSpace authorization system will provide role-based access controls within each Repository to enforce access restrictions within a single Repository (see Staff User specification).

Search filters will be used to prevent one Tenant from executing discovery operations on another Tenant’s data. Subsequently, tenant information must be available to the search and indexing layer of the application as well.