Tenant Isolation Model

Last Modified on 20/01/2022 11:24 am AEDT

Overview

The ReadiNow platform is designed to support multiple tenants (clients) concurrently on the same platform. The design ensures that it is impossible for any information of one tenant to be accessible in any way by users of another tenant. This requirement has been a key factor in the design, architecture, and implementation of the ReadiNow platform since its inception.

Every tenant database record includes a tenant identifier. The design of the ReadiNow platform means that only a small number of code-paths can reach the database. Any calls to the database are made such that the query is always constrained to the tenant of the current authenticated user. In a similar manner, various server caches used by the ReadiNow platform isolate data by tenant, such that cached data is partitioned by tenant, and source code can only access cache data for the tenant of the current user. Binary data is associated with tenants via cryptographically secure hashes.

All requests to the ReadiNow platform to read or modify data include a security token for the current user. In this way, every request is necessarily associated with a user and tenant. In conjunction, this means that it is not possible for any authenticated user to interact with any database information for a tenant other than their own.

This design ensures that the following are achieved through a robust architecture design: 1. Isolation of tenant data between tenants, accessible only to users of that tenant; 2. all data for a tenant can be identified; 3. where required, all data for a tenant can be removed.

Detail

Database structure

The ReadiNow platform contains a relatively small number of database tables for holding customer data. These include a table of Entities (records), Relationships, and field data tables. Each of these contain a TenantID column so that each piece of data is associated with a tenant.

For example, the Relationship table contains columns:

TenantId	Rel type ID	From ID	To ID	Other column(s)

Database access

Many of the database queries performed by the ReadiNow platform are dynamically generated to match the particular data or reporting requirements of each customer and application.

There are two engines for generating dynamic SQL queries: a report query generation engine for generating tabular reports; a graph-based query generation engine for loading more general entity relationship data. The latter is used for loading much of the user data and application metadata data for the platform. In both cases, the query generation engines explicitly enforce a tenant filter on each table join such that it cannot be circumvented by the dynamic

queries. (Further, no portion of the dynamic query is ever derived directly from unfiltered string data, to prevent any possibility of SQL injection attacks).

The ReadiNow platform software also contains a relatively small number of other static queries that serve various parts of the product. These also explicitly filter the tenant. Tenant filtering is always done in addition to any other filtering clauses that may be present in the query.

In this way, the platform architecture ensures that database accesses are filtered to the tenant.

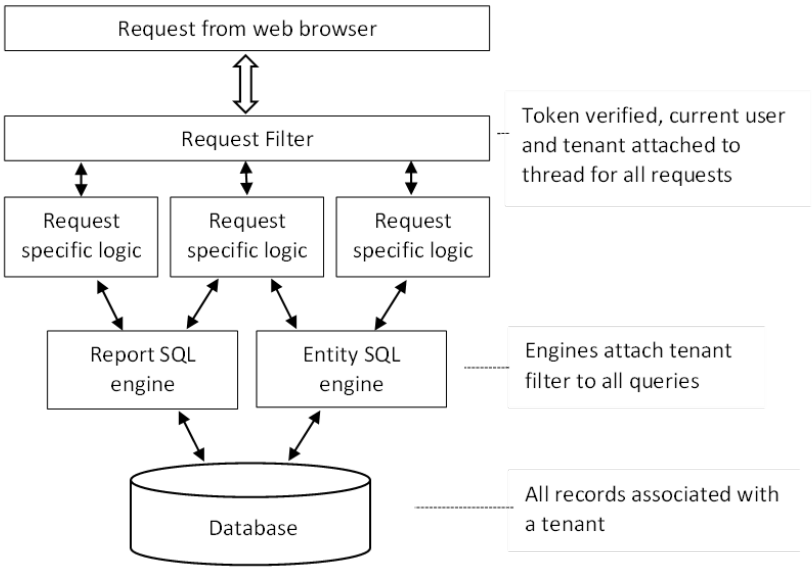
Identifying the Current Tenant

When a user logs into the ReadiNow platform, an authentication token is issued in the form of a per-session cookie. This token is then returned to the server as part of every web request to retrieve or modify data.

An ASP web filter is used to ensure that every request is intercepted at the server to ensure that the request contains a valid authentication token. This allows every request to be associated with a user and a tenant. This information is stored on the current executing thread, so that all code paths have reliable direct access to the current user ID and tenant ID. This tenant ID is what is applied in the database query filters described in the previous section. In this way, only authorised users can make requests to read or update data; and the architecture scopes all queries to only apply to data in their tenant.

Pipeline

As depicted, platform logic cannot reach the database without queries being constrained to the tenant for the current user.



Storage of binary data

Binary data such as application icons, documents, and generated report templates, are stored separately to the main database. Data deduplication is implemented by taking a cryptographically secure (SHA256) hash of the binary data and indexing data by the hash. This means that for all intents and purposes the hash securely uniquely identifies the

document. The hash is then stored in the database, along with the tenant ID and filtered in the same manner as other database data.

In this way, each binary is securely protected by tenant filtering via the hash. However, necessarily to achieve data de-duplication, one binary may be associated with multiple tenants.

It is therefore possible to identify all binaries associated with a tenant; and to remove all binaries uniquely owned by a tenant.

Cache Tenant Isolation

The ReadiNow platform server makes use of a number of caches – both in memory, and in separate processes/servers. A shared code-path is used by various systems for establishing and accessing caches. This code path includes a tenant-isolation layer that ensures that cached data is associated with a particular layer. This is typically achieved either by isolating data into sub caches with one sub-cache per tenant; or by augmenting the cache key to include the tenant. By design this is done by the common cache code used by all caches, rather than particular specific caches to maximise protection. Per tenant cache isolation also allows for caches to be refreshed on a per-tenant basis.

As with the query filtering, cache accesses use the tenant ID for the current thread. In this way a user requesting to read or modify data can only reach cache entries for their tenant.