Multi-Tenant Architecture

Ext.prc.002.02 | 01.06.2024

UPM 24.x

ZINFI Confidential & Proprietary
Shared Under NDA

Contents

Multi-Tenant Data Architecture

ZINFI's Unified Partner Management Platform has two core solutions:

- Partner relationship management (PRM)
- Partner marketing management (PMM), which is also know in the industry as through-channel marketing automation (TCMA) or through-partner marketing automation (TPMA)

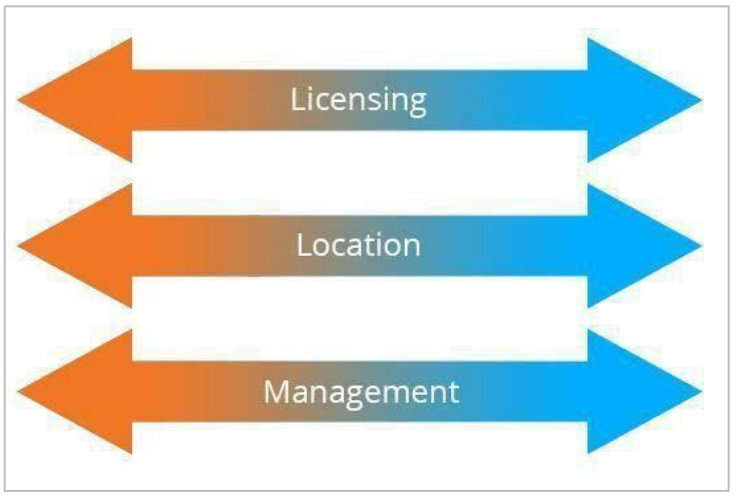
This document addresses the core UPM platform architecture and thus applies to both PRM and PMM.

In ZINFI's multi-tenant architecture, a single instance of software runs on a server and serves multiple tenants. A tenant is a group of users who share common access with specific privileges to the software instance. With a multi-tenant architecture, ZINFI's software application is able to provide every tenant a dedicated share of the instance—including its data, configuration, user management, individual tenant functionality and non-functional properties. Multi-tenancy contrasts with multi-instance architectures, where separate software instances operate on behalf of different tenants.

A case could be made that data is the most important asset of any business—data about products, customers, employees, suppliers and more. And data, of course, is at the heart of software-as-a-service (SaaS). ZINFI's UPM SaaS applications provide customers with centralized, network-based access to data with less overhead than is possible when using a locally installed application.

ZINFI provides two options for SaaS: private cloud (dedicated infrastructure for the client) and public cloud (shared infrastructure). The differences are explained below.

In the "pure" form of SaaS, ZINFI hosts an application centrally and delivers access to multiple customers over the Internet. In practice, however, the defining characteristics that distinguish an on-premise application from a SaaS application are not absolute, but are graduated along three different dimensions: how software is licensed, where it is located and how it is managed. Each of these variables can be visualized as a continuum, with traditional on-premise software on one end and pure SaaS at the other. In between are additional options that combine aspects of both.



- **Licensing:** On-premise applications are typically licensed in perpetuity, with a single up-front cost for each user or site, or—in the case of custom-built applications—owned outright. SaaS applications often are licensed with a usage-based transaction model, in which the customer is only billed for the number of service transactions used.

- **Location:** SaaS applications are installed at ZINFI's data center location, while on-premise applications are installed within a client's own IT environment. In between is the appliance model, in which the vendor supplies a hardware/software component as a "black box" that is installed at a client location instead of the vendor's.

- **Management:** Traditionally, the IT department is responsible for providing IT service to users, which means being familiar with network, server and application platforms; providing support and troubleshooting; and resolving IT security, reliability, performance and availability problems. This is a big job, and some IT departments subcontract some of these management responsibilities to third-party service providers that specialize in IT management. At the other end of the spectrum, SaaS applications are completely managed by ZINFI.

Approaches to Managing Multi-Tenant Data

ZINFI has an architecture that allows it to deploy multi-tenant functionality in three separate modes. The following is a brief description of the architectural flexibility that ZINFI can provide.

Separate Databases (E1)

Computing resources and application code are generally shared between all the tenants on a server, but each tenant has its own set of data that remains logically isolated from data that belongs to all other tenants. Metadata associates each database with the correct tenant, and database security prevents any tenant from accidentally or maliciously accessing other tenants' data.

Giving each tenant its own database makes it easy to extend the application's data model (discussed later) to meet tenants' individual needs, and restoring a tenant's data from backups in the event of a failure is a relatively simple procedure. Unfortunately, this approach tends to lead to higher costs related to maintaining equipment and backing up tenant data. Hardware costs are also higher than they are under alternative approaches, as the number of tenants that can be housed on a given database server is limited by the number of databases that the server can support.

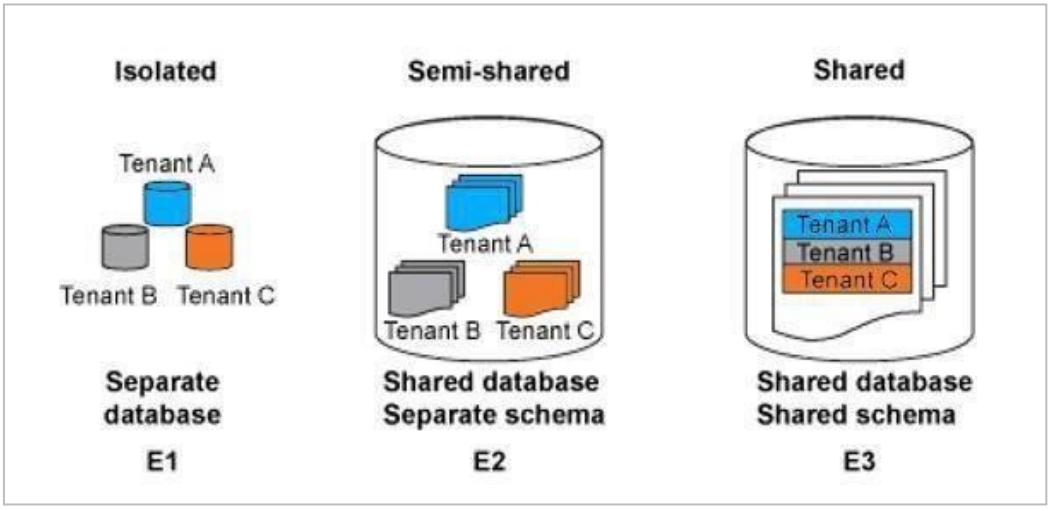
Shared Database, Separate Schemas (E2)

Another approach involves housing multiple tenants in the same database, with each tenant having its own set of tables that are grouped into a schema created specifically for the tenant. When a customer first subscribes to the service, the provisioning subsystem creates a discrete set of tables for the tenant and associates it with the tenant's own schema. Like the isolated approach, the separate-schema approach is relatively easy to implement, and tenants can extend the data model as easily as with the separate-database approach.

Shared Database, Shared Schema (E3)

A third approach involves using the same database and the same set of tables to host multiple tenants' data. A given table can include records from multiple tenants stored in any order; a Tenant ID column associates every record with the appropriate tenant.

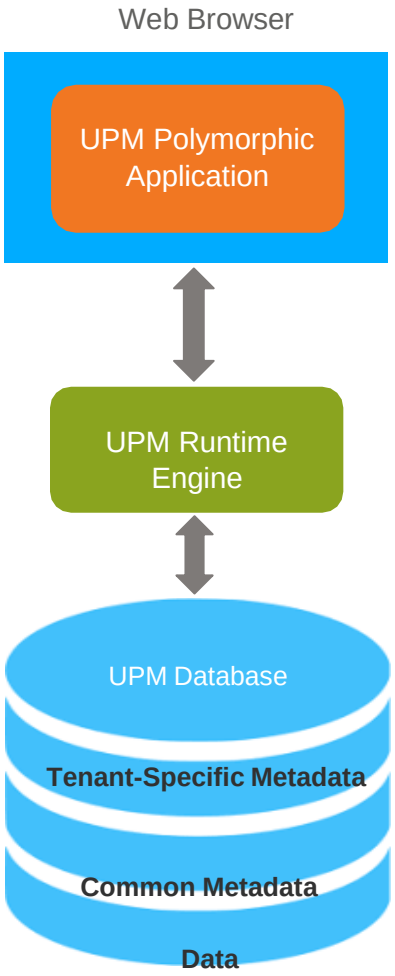
Among the three approaches explained here, the shared schema approach has the lowest hardware and backup costs, because it allows you to serve the largest number of tenants per database server. However, because multiple tenants share the same database tables, this approach may incur additional development effort in the area of security, to ensure that tenants can never access other tenants' data, even in the event of unexpected bugs or attacks.



With this as a backdrop, in a majority of ZINFI's deployments we use the "Separate Databases (E1)" approach to provide services to multiple tenants and secure each database independently. However, the architecture allows implementation in E2 and E3 modes as well.

Meta-Data-Driven Reference Architecture

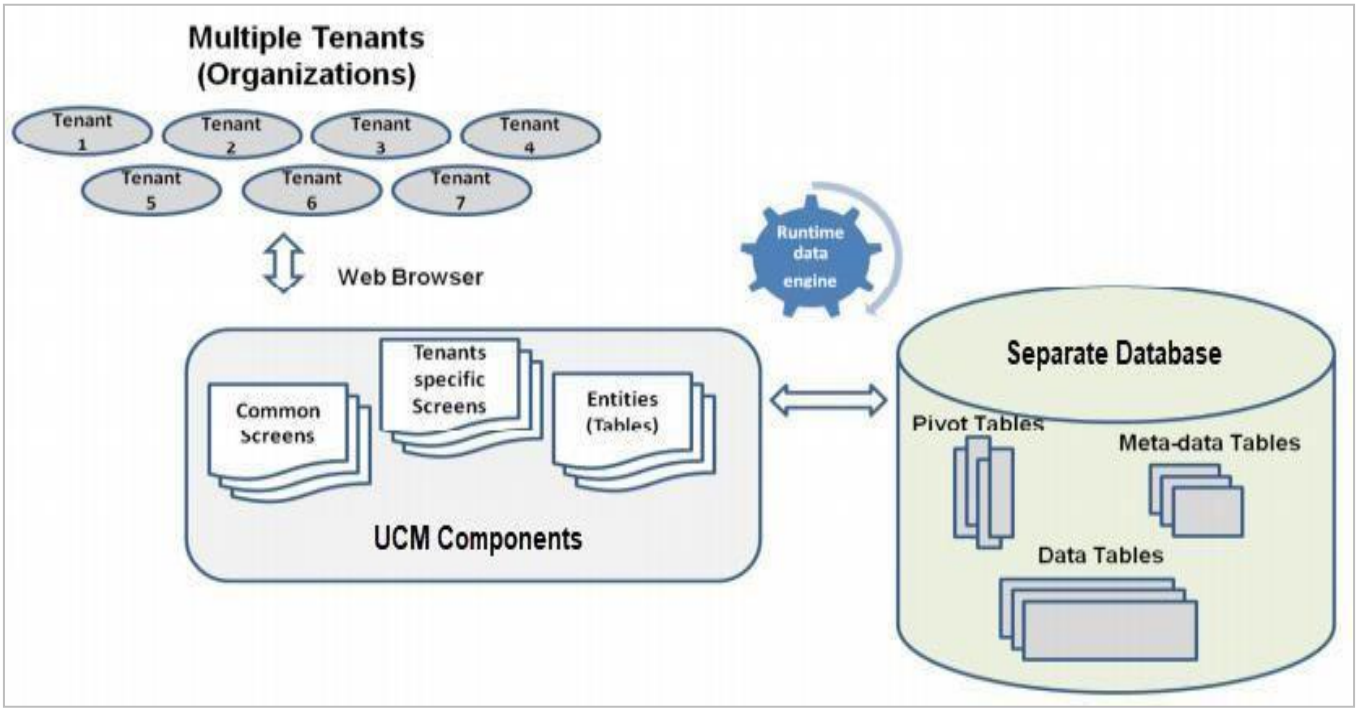
- ZINFI's multi-tenant application—UPM—is dynamic, or polymorphic, in nature so it can fulfill the specific expectations of various tenants and their users.



- Application components are generated at runtime from metadata (i.e., data about the application itself).

- The metadata-driven architecture is well-defined with a separate component for:
  - Runtime application data
  - Metadata that describes the base functionality of an application
  - Metadata that corresponds to each instance of tenant-specific data and customizations

- UPM makes it possible to independently update the core system, modify the core application and customize tenant-specific components with virtually no risk affecting others.



Approach to Data Security

We have implemented the following items to develop a secure multi-tenant application for ZINFI UPM platform:

- **Trusted Database Connections** – With the trusted subsystem access method, the application always connects to the database using its own application process identity, independent of the identity of the user; the server then grants the application access to the database objects that the application can read or manipulate.

- **Secure Database Tables** – This pattern is appropriate for use with the separate-database and separate-schema approaches. In the separate-database approach, you can isolate data by simply restricting access on a database-wide level to the tenant associated with that database, although you can also use this pattern on the table level to create another layer of security

- **Tenant Data Encryption** – With this method, the application uses impersonation to access the database using the tenant's security context, which grants the application process access to the tenant's private key. The application (still impersonating the tenant) can then use the tenant's private key to decrypt the tenant's symmetric key and use it to read and write data.

Benefits of UPM's Multi-Tenant Architecture

- **Reduces investment costs in the long run**

  A prime benefit of using a multi-tenant architecture is that it reduces the size of investments in the long run. Compared with a single-tenant architecture, a multi-tenant SaaS applicationcosts significantly less. That's because multi-tenancy enables the sharing of applications and resources.

- **On-boarding a new partner is very convenient**

  Customer onboarding has become a prime focus of managing partner networks because poor onboarding experiences can directly affect growth prospects. UPM's multi-tenant application ensures the lowest cost of solution delivery because no new software resources, code changes,or database setups are required for each incremental tenant.

  This application automatically performs tasks like setting up and configuring default data for users. Hence, the cost of onboarding a new partner begins to approach zero at full scale, resulting in an ever-increasing revenue margin.

- **Maintaining applications is efficient because all partners use the same application**

  Modules in UPM are highly configurable. This enables each tenant to use the application most conveniently, without changing the underlying code. The data structure does not need to change, as the code is shared and remains common.

- **Allows the application to hold billions of tenants**

  In ZINFI UPM, unlike a single tenant solution, the tenants use a common infrastructure. There is no need for raising the number of data centers for the individual tenants. Hence, scaling has much less daunting implications for vendors.

- **Maximum usage of resources**

  The ability to optimize resource utilization is much higher in UPM compared to a single-tenant application. ZINIFI UPM uses the same infrastructure and resources, so utilization is automatically optimized, and maintenance becomes progressively more efficient. Also, if a tenant does not utilize a specific resource at any given point in time, another tenant can use it. This makes the utility of all resources commensurate with each other.

