

uniPaaS Technology Overview

October 2008

Magic Software is a trademark of Magic Software Enterprises Ltd. All other product and company names mentioned herein are for identification purposes only and are the property of, and may be trademarks of, their respective owners.

Magic Software Enterprises has made every effort to ensure that the information contained in this document is accurate; however, there are no representations or warranties regarding this information, including warranties of merchantability or fitness for a particular purpose. Magic Software Enterprises assumes no responsibility for errors or omissions that may occur in this document. The information in this document is subject to change without prior notice and does not represent a commitment by Magic Software Enterprises or its representatives.

TABLE OF CONTENTS

1 WHAT IS UNIPAAS? 4

1.1 KEY PRINCIPLES..... 4

1.2 SUPPORTING DEPLOYMENT ARCHITECTURES 7

1.3 SOA ENABLED 8

2 DEVELOPMENT PARADIGM – STUDIO..... 9

2.1 CUSTOMIZING THE PRE-COMPILED APPLICATION ENGINE..... 9

2.2 THE REPOSITORIES..... 9

2.3 THE TASK..... 10

2.4 EVENT DRIVEN..... 10

2.5 LOGICAL DATA VIEW 11

2.6 INTEGRATED FORM EDITOR 12

2.7 COMPOSITE APPLICATIONS – OPEN ENVIRONMENT 13

2.8 OPEN ENVIRONMENT..... 13

2.9 APPLICATION SOURCE MANAGEMENT 13

2.10 PROJECT DOCUMENTATION..... 14

2.11 APPLICATION DEBUGGING, MONITORING & LOGGING..... 14

2.12 REPORTS AND PRINTOUTS 15

3 RICH INTERNET APPLICATIONS 16

3.1 THE RIA ARCHITECTURE 16

3.2 A FIT CLIENT 17

3.3 A UNIFIED PARADIGM..... 17

3.4 AUTOMATIC LOGIC PARTITIONING 18

3.5 PERFORMANCE AWARE DEVELOPMENT 18

3.6 TRANSPARENT CONTEXT MANAGEMENT..... 18

3.7 INTEGRATED FORM EDITOR..... 18

3.8 LOCAL RESOURCES ACCESSIBILITY 18

3.9 TIGHT APPLICATION AND DATA SECURITY..... 19

4 SOFTWARE-AS-A-SERVICE (SAAS) 20

4.1 SAAS-ENABLED APPLICATION PLATFORM (SEAP)..... 20

4.2 MULTI-TENANT SUPPORT 20

4.3 MANAGEMENT & MONITORING 21

5	FULL CLIENT APPLICATIONS	22
5.1	SUPPORTS 1, 2, AND 3 TIER ARCHITECTURES.....	22
5.2	DIRECT DATA ACCESS.....	22
5.3	INTEGRATED END-USER REPORT GENERATOR.....	22
5.4	PARTITIONING.....	22
6	VERSATILE WEB APPLICATIONS	23
6.1	FLEXIBLE HTTP (MARKUP LANGUAGE) INTERFACE HANDLING.....	23
7	SERVICE PROVISIONING	24
7.1	SOA – WEB SERVICES.....	24
8	ABOUT MAGIC SOFTWARE ENTERPRISES	25

1 What Is uniPaaS?

uniPaaS is Magic Software's comprehensive application platform and the next generation of the company's award-winning eDeveloper series. It is also **the industry's first** RIA and SaaS-enabled Application Platform (SEAP) that uses a single development paradigm to automatically handle all Client and Server partitioning. Unique to the market, uniPaaS gives you the power to choose how you deploy your applications, whether Full Client or Web; on-premise or on-demand; software or Software-as-a-Service (SaaS); or global or local.

Another industry first is the hybrid deployment capability – all the various deployment modes (Desktop, Client/Server, HTML Web Applications and Web 2.0 Rich Internet Applications) are defined in the same application metadata and development project – meaning that application changes can be done once and are automatically propagated in any deployment mode. Here again, uniPaaS provides application owners significant cost savings and a significantly lower total-cost-of-ownership (TCO).

A third high impact is the seamless compatibility with existing eDeveloper V10 applications. This means that tens of thousands of vertical business applications can be easily and rapidly migrated to become available over the Web, while at the same time catering to the existing on-premise installed base. This responds to one of the main challenges of the SaaS wave, which is the availability of RIA applications. Magic Software expects that many of its ISVs will start upgrading their solutions to become Rich Internet Applications, becoming available for SaaS deployment with the second release of uniPaaS. This will make available to Platform-as-a-Service providers, who are using uniPaaS, one of the richest and most global application portfolios in the industry, while opening up the new SaaS market to Magic Software's ISV community.

1.1 Key Principles

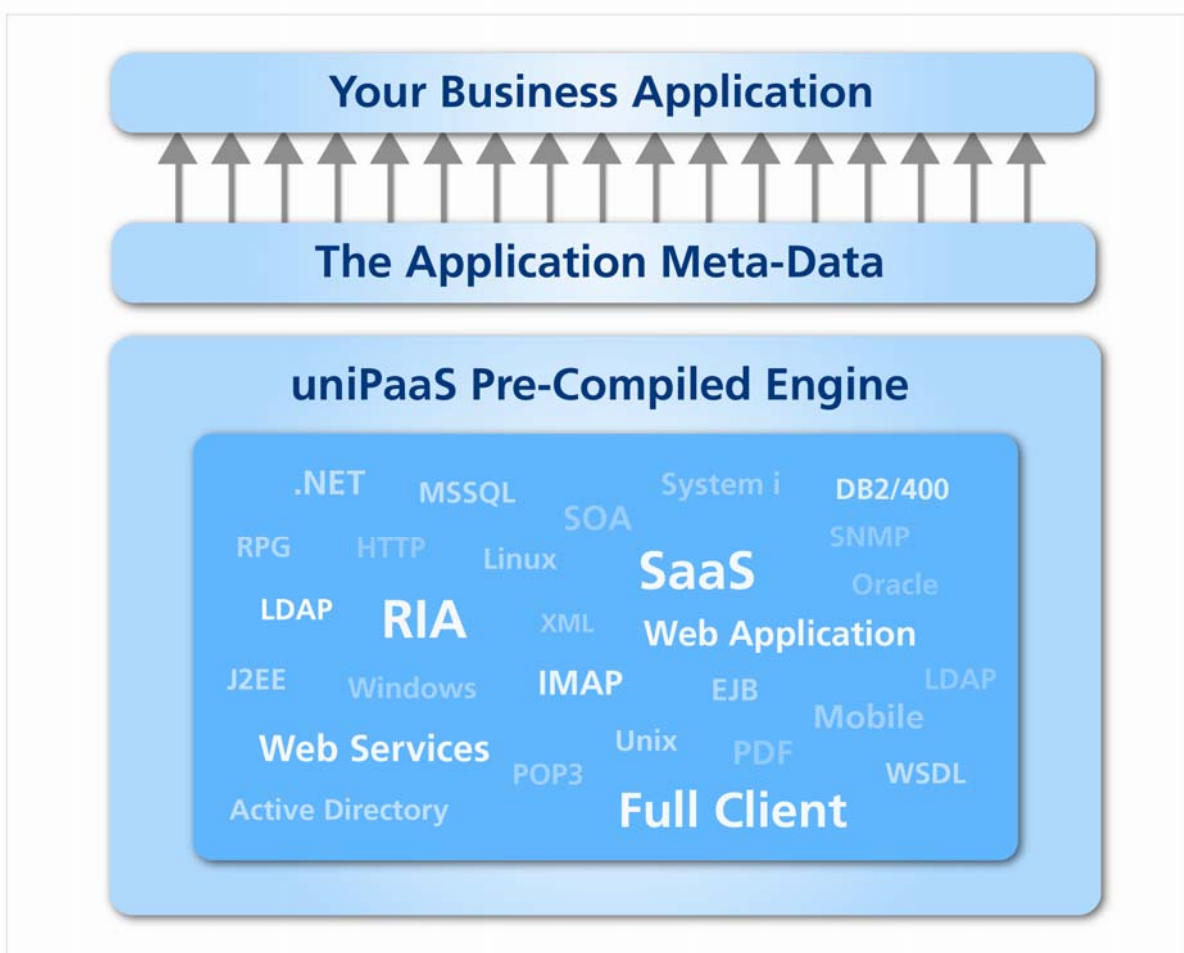
1.1.1 Metadata Driven Pre-Compiled Engine

Most programs in a business application share a common schematic flow. The key principle of the uniPaaS deployment engine is that the code for supporting the common flow of a business task has already been written, compiled and tested to facilitate any specific business program. This makes the development of any business task very quick and straightforward and boils down to a very concise way of fine-tuning the skeleton of the business program to serve the needs of the actual program.

Fine-tuning the schematic and pre-compiled program unit of an application is done by a simple declaration of what actual artifacts are to be handled by the program, what is the precise logic needed to be executed by the program and what is the user interface design required for the program.

This metadata definition of the programs, together with other elements of the application that are also described in a metadata form, such as models, data tables, and Help screens, are stored in XML-based files that constitute the application source “code”.

The pre-compiled deployment engine is designed to support any business application functionality to provide application virtualization.



uniPaaS' pre-compiled engine contains every technological component needed to build your unique application via meta-data driven specifications.

1.1.2 Virtual Machine

Being pre-compiled and metadata driven, the uniPaaS server engine is in practice the ultimate virtual machine, enabling the freedom of implementing the application on various server machines without making any modification to the application metadata. The virtual machine keeps a clear separation between the specific environment definitions and the application business logic, enabling easy customization for new deployments.

1.1.3 Backward Compatibility

In addition to being a cross-machine server engine, the uniPaaS virtual machine engine is at the core principle that facilitates the smooth migration from past versions of the platform to the newest version. The array of applications originating since over 20 years ago and being migrated over the years to the latest technologies are a living proof of the uniPaaS ability to preserve past IT investments.

1.1.4 Designed for Business Applications

As opposed to other development environments that cater to any type of software, uniPaaS is designed from ground up specifically for business applications. uniPaaS's primary focus is to satisfy the needs of complex business applications and to easily bridge between the bare business requirements of IT and the actual technology that is required to support the business requirements. A uniPaaS designer is almost entirely focused on the expected business benefit and outcome of the application and less troubled with the technicalities of how these outcomes can be achieved.

This principle is what makes uniPaaS designers such a special breed that on one hand have a clear understanding of the business aspects and requirements of an enterprise application and on the other hand have the ability to transform the business requirements into a live business application.

1.1.5 Rule-Based Engine – The Re-computation Acceleration

At the heart of uniPaaS' extreme productivity, lies the rule-based capability that enables simple requirements and data-driven definitions that govern most of the way the application is reflected to the end-user and the way data is collected and manipulated.

1.1.6 Multilingual Support

One of the challenges of producing cross enterprise applications that centralize the IT of disparate units is the ability to properly support multiple languages both in terms of the user interface of the application and the collected and presented live data. uniPaaS transforms any application into a multinational application by automatically translating any static text or portions of dynamic text based on a simple dictionary that is provided with the deployment engine.

1.2 Supporting Deployment Architectures

uniPaaS caters to various deployment modes using a highly flexible and scalable architecture.

The deployment architectures facilitate the entire range from Full Client application through Rich Internet Application to backend services provisioning.

1.2.1 Deployment Models

uniPaaS allows for the deployment of any application by the following models:

- **Rich Internet Application** – A human centric, fully interactive and fully connected internet-based application that can be activated from any location.
- **Full Client Application** – A human centric application providing fully connected applications that run fully on the client machine.
- **Partitioned Full Client Application** – Enhancing the service level of a Full Client application by partitioning selected tasks to run on a centralized server
- **Web Applications** – Web technology based applications ranging from plain dynamic HTML pages through Ajax based applications to Flash-based applications.
- **Service Provisioning** – A backend application servicing other applications, Web portals, mash-ups and so forth.



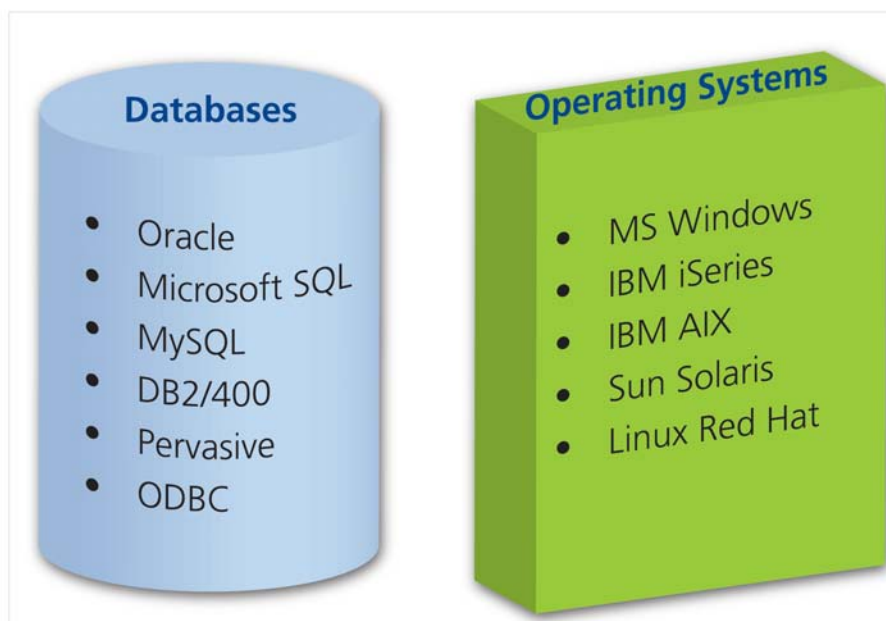
uniPaaS' centralized application meta-data enables a variety of deployment modes.

1.2.2 N-Tier

With the open architecture model uniPaaS offers an application that can be easily designed and deployed in any given N-Tier model to achieve greater scalability, flexibility and tight security.

1.2.3 Multi-Platform & Multi-Database Support

The freedom of choice in developing and deploying a uniPaaS application goes beyond the deployment mode of the application users. It also goes to the freedom of choosing any fundamental infrastructure to support the deployed application. This freedom of deployment allows the organization to easily adapt its uniPaaS applications to any varying infrastructure that the corporate decides to utilize.



The variety of databases and operating systems supported by uniPaaS.

1.3 SOA Enabled

In any form of deployment, uniPaaS enables the provisioning and consumption of services to either facilitate or adhere to any Service Oriented Architecture of the enterprise.

SOA-based applications can be achieved by various means that are inherently supported by uniPaaS, such as:

- Web Services
- Simple Web Access
- Messaging

uniPaaS offers a user-friendly wizard that enables smooth integration of service-providing facilities for the creation of composite applications.

2 Development Paradigm – Studio

Together with its unique productive and agile development paradigm, uniPaaS supports mainstream programming methodologies and offers a wide range of tools and utilities to complement the development cycle.

2.1 Customizing the Pre-Compiled Application Engine

The uniPaaS deployment engine is the driving force behind uniPaaS' ability to perform complex data manipulations that are largely transparent to the developer and the end-user. The engine provides a set of operations for the developer's use and a structure of execution steps called logic units, which together perform the tasks set for the end-user. The engine knows how to perform such actions as opening files, reading records, sorting, displaying data on the screen, and more. This is different from other application development tools, where most of these actions would be programmed by the developer. uniPaaS saves the developer considerable amounts of time by supplying these built-in operations.

2.2 The Repositories

The workspace of a uniPaaS application is constructed of a collection of repositories in which the building blocks of the application are defined, accessed and managed. The repository-based application development produces highly governed applications. Every uniPaaS application keeps the following repositories:

- Models – A collection of prototype definitions for the basic elements of the application, which includes data and visual elements.
- Data Sources – A collection of all the data sources of the application, including databases and XML documents.
- Programs – The core elements of the application. The executable entities that make the application run and function.
- Help Screens – A collection of help pages, tooltips, and comments screens used by the application.
- Rights – A collection of all the available user rights, which are referred to by the application for automatic resolution according to the assigned rights of the user running the application.
- Menus – A collection of menus and context menus that are used throughout the application.
- Composites Resources – A collection of all the components used by the application.

The development paradigm of uniPaaS is based on references made throughout the application to connect the various elements into a fully functional application.

2.2.1 Inheritance

Any modification introduced to each of the elements in each repository is reflected immediately in the referencing elements providing full inheritance and immediate reflection of cross application changes.

2.2.2 Application Governance

The object-based development offers enhanced application governance by which every element can be closely tracked and every reference made to an object can be followed.

2.2.3 Object Reuse

The pre-definition of every element in the application allows for constant reuse of the pre-defined elements, which leads to highly productive development cycles.

2.3 The Task

The most significant building block of a uniPaaS application is the **task**. A task is the basic object used to construct programs and implement procedures. The engine executes a task by looping through the data source defined for the task. In interactive tasks, the end-user browses through the data source by moving from record to record. This means that the only records processed are those browsed by the end-user. In non-interactive tasks, the uniPaaS engine loops through the data source, starting with the first record in the data view and moving to every record until the last record is reached. Once the data source and the task type are specified, the task can be customized by specifying the operations that must occur, while uniPaaS or the end-user loops through the data source records.

2.4 Event Driven

With uniPaaS you can better define your business logic using the event-driven architecture of the uniPaaS deployment engine. The event-driven development paradigm allows for creating full-featured applications with clearly defined logic.

An event-driven application can respond to any runtime occurrence generated by the end-user or by the developer internally as part of the business logic. Event-driven applications provide the following benefits:

- Clearer Code – Using uniPaaS' event-driven architecture, the business logic is neatly defined in each task for the event that the task should handle. This makes the task clearer, more understandable, and easier for other developers to maintain.
- Greater Flexibility – Event-driven architecture allows for better handling of runtime occurrences generated by the end-user or by third-party components.
- Reusability – The event-driven architecture allows for defining the business logic as global logic that is defined only once and reused throughout your application. Having such sharable logic segments saves time in writing and maintaining repeated logic segments.

2.5 Logical Data view

Complex data handling is easily achieved by a unified and coherent definition of a logical data view that serves the task. The logical data view can be comprised of various data source ranging from SQL through ISAM and memory stored tables to complex XML documents, making the data handling completely independent of the underlying source; this enables the developer to concentrate on the actual handling of the data source records and columns.

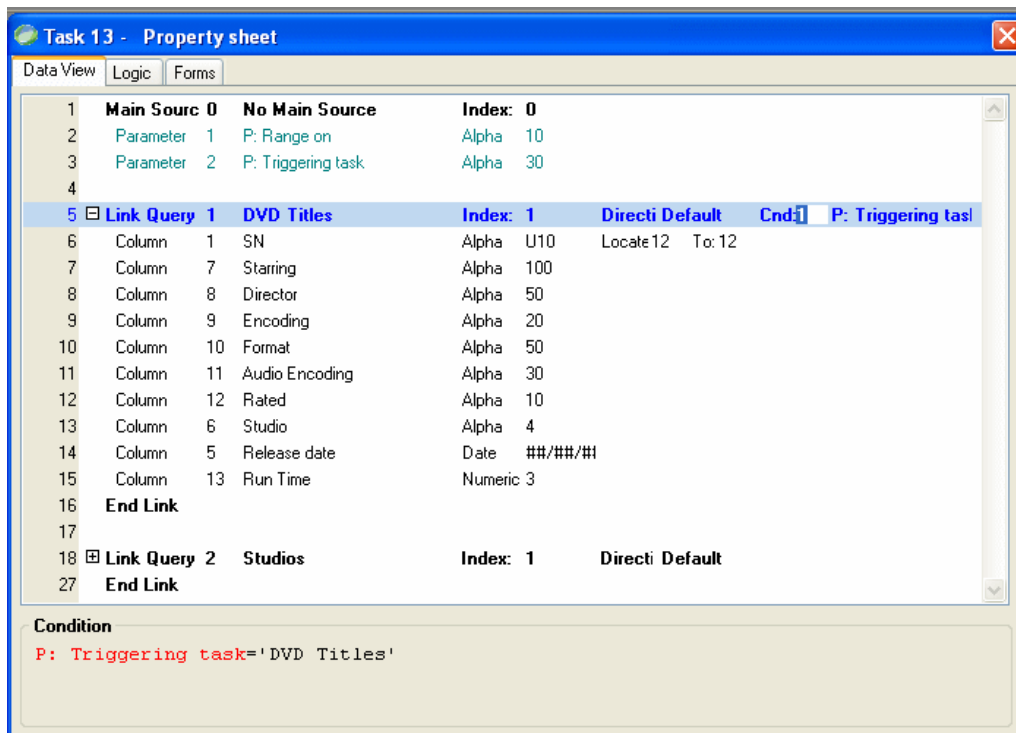


Figure 4: The uniPaaS data view editor.

2.5.1 SQL Databases

uniPaaS applications can access, utilize and manipulate data from various SQL databases. In addition to ODBC support as a generic gateway to various databases, uniPaaS also natively supports access to the following SQL databases:

- Oracle
- Microsoft SQL
- Pervasive SQL
- IBM DB2

2.5.2 XML Views

uniPaaS enables the developer to handle XML documents and to extract and manipulate their content in a completely automated manner by hiding the low-level complexity of the XML structure. Loading the schema of any XML document into the application, transforms the XML document into a simple XML view

that the developer can then easily access for the data view definition of every task as if it was a regular data source. All the physical handling of the XML content is done transparently by uniPaaS as response to common activities, such as fetching records, record insert, record update and record deletion.

2.5.3 DBMS Transparency

The native gateways of the various SQL databases automatically translate every visual definition of the logical data view into an SQL script tailored for the specification of the underlying database. This enables the developer to develop business applications using a high level data definition and, at any time, automatically switch from one database to the other and have the native gateway that uniPaaS provides adapt the application to the new database.

2.5.4 Direct SQL Script

In addition to the high level visual definition of specific data views of each task, the developer may choose to retrieve data by directly defining an explicit SQL script. uniPaaS assists the developer by providing a generator to easily construct the SQL script.

2.6 Integrated Form Editor

uniPaaS provides a built-in form editor by which the interface of both interactive and non-interactive tasks is created. Every control on the form can have its content and any other property fully data-bound by merely associating data elements to the controls. The event handling logic units of a task can be designed to respond and handle control-specific events to achieve a much more enhanced interactive user experience.

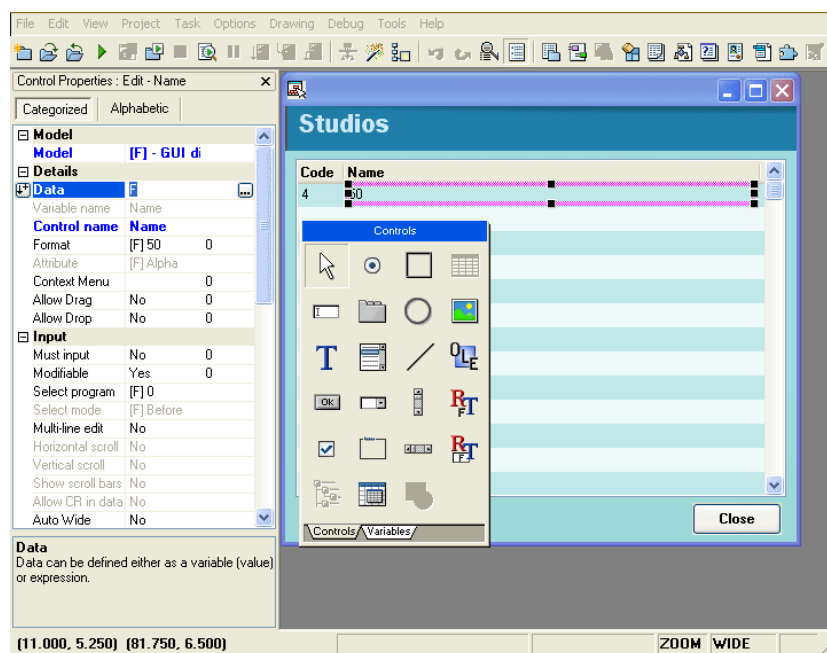


Figure 5: The uniPaaS form editor.

2.7 Composite Applications – Open Environment

2.7.1 Internal Components

Any internal object of a uniPaaS application can be easily encapsulated as an independent component, which can then be embedded within other uniPaaS applications. In this way, applications can easily be constructed by utilizing existing components that make up either a common skeleton of an application or as an extension to the functionality of the application.

2.7.2 Embedding External Components

Various types of external objects created by different tools can be easily interwoven into any uniPaaS application for user interface and functionality enhancements.

2.8 Open Environment

uniPaaS offers a highly open environment in which third-party resources can be easily combined with the application. Moreover, a uniPaaS application can also be accessed and triggered by external applications.

2.8.1 .NET Object Insertion

Any .NET object can be directly inserted into the interface and the flow of a uniPaaS application. In this way, developers can benefit from the abundant resources available within the .NET ecosystem.

2.8.2 Web Service Support

A uniPaaS application can be easily designed to consume Web Services through which centralized information can be retrieved and centralized activities can be triggered. uniPaaS also enables the developer to produce service provisioning applications by exposing the application tasks using a Web services interface (WSDL) and thereby provide Web services for any third-party consumer.

2.9 Application Source Management

uniPaaS applications can be set to be maintained on a centralized Version Control server to facilitate full source management and properly managed concurrent development. uniPaaS supports the standard SCCI (Source Code Control Interface), which enables the uniPaaS Studio to utilize any external version control software for providing the version control functionality. Though an external software is used for the source management, all source management related functionality is directly accessed from built-in commands, toolbars, and menu entries.

2.10 Project Documentation

Using the bundled Magic Optimizer utility, every uniPaaS application can have its entire definition and setup extracted in a well-formatted document describing the entire application. This documentation serves as a highly important reference of the application and serves as an excellent tool for code review and for application maintenance challenges.

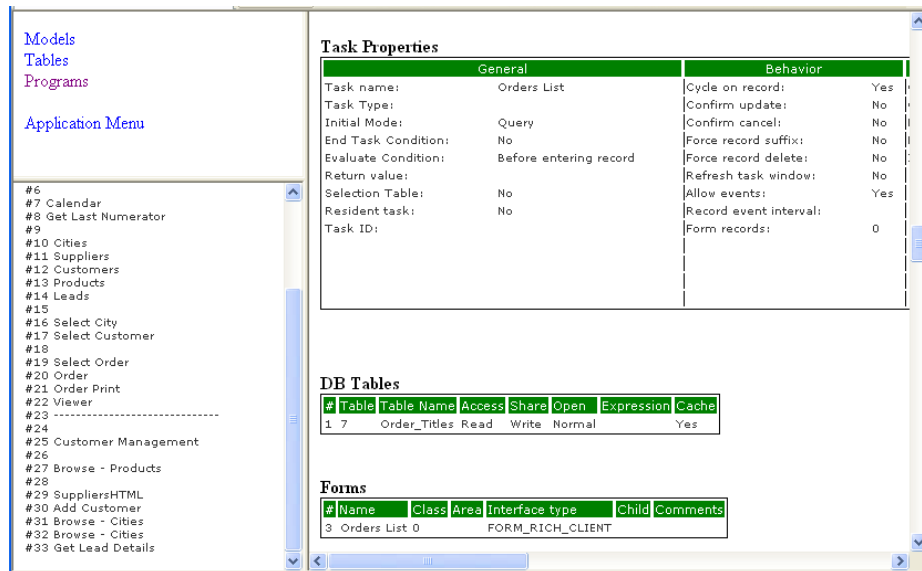


Figure 6: The Documentation of an example project.

2.11 Application Debugging, Monitoring & Logging

Keeping track of application quality, performance, and integrity is done using the built-in utilities of uniPaaS for monitoring, logging and debugging.

2.11.1 Application Debugging

The uniPaaS Studio is equipped with full debugging capabilities to ensure maximal detection of application bugs. The debugging capabilities include simple and conditional breakpoints, Variables Watch, direct variable updates, and a Call Stack view.

2.11.2 Deployment Monitoring

A deployed application, whether it is a server-based application or a client-based application, can be closely monitored for profiling its execution and for properly assessing its robustness and optimization.

2.11.3 Remote Debugging

In cases where unexpected behavior is reproduced only on the deployment site, uniPaaS also enables applications to be closely debugged when they are already deployed on remote sites.

Using a TCP/IP connection, a uniPaaS Studio can connect to the running application and allow the developer to fully debug the remote application.

2.12 Reports and Printouts

uniPaaS offers a built-in and easy-to-use mechanism by which developers can design any type of report and output, from plain text-based reports to highly sophisticated reports. The built-in mechanism of the non-interactive task allows for great flexibility in creating deep hierarchies of break levels with customized and varying break level commutations.

2.12.1 Integrated PDF Support

PDF documents can be automatically generated by the uniPaaS engine without the need for installing a printer driver. Every report can be dynamically configured to either send the output to a printer driver or to generate a PDF document.

3 Rich Internet Applications

By keeping the ease of use and productivity of desktop applications and combining it with a lean architecture and wide availability, Rich Internet Applications (RIA) have become the most preferred alternative for both internal and external business applications.

3.1 The RIA Architecture

The architecture that supports RIA consists of the following modules:

- **The uniPaaS Server**
The uniPaaS Server is the core process that serves the remote clients.
- **The Web Server**
A standard Web server used to handle remote HTTP-based requests.
- **The Magic Internet Requester**
An extension of the Web server that is used to connect the client modules to the uniPaaS server and vice versa.
- **The Magic Requests Broker**
The middleware governing the flow of communication between the Magic Internet Requester and the uniPaaS Server. The Magic Requests broker is in charge of the load balancing by which each requests is directed to the most suitable uniPaaS server process. The broker is also responsible to execute the fault tolerance policy and the capacity surge policy.
- **The uniPaaS RIA Client Module**
A generic client module that servers as the front-end of the application.



3.2 A Fit Client

Bridging between the two extremes of a high performing but costly Fat client and a disappointing experience but low cost Thin client, Magic Software introduces the Fit client – a high performing yet low cost client.

3.2.1 Generic .NET Module

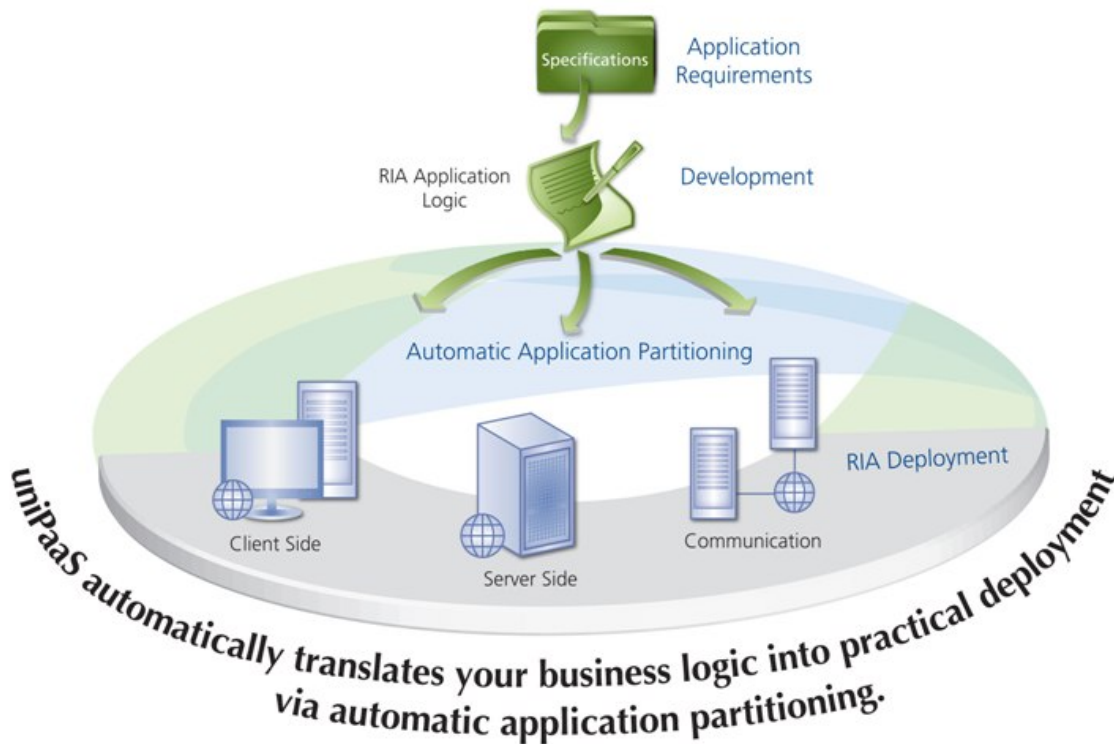
The front-end of a RIA is facilitated by a generic .NET client module. This generic module is used for all uniPaaS RIA applications.

3.2.2 Browser Free

One of the most significant advantages of the uniPaaS .NET-based client module is the independence of the client layer from any additional client-side layer, such as a browser. Being browser-free the client is less limited by functionality, thus offering greater flexibility and a much better user experience.

3.3 A Unified Paradigm

Unlike other RIA technologies, which call for a collection of toolsets in order to implement the client side, server side, and client/server communication, the uniPaaS RIA development paradigm relieves the developer from the deployment separation of the entities and allows the developer to remain focused on the business functionality of the application.



3.4 Automatic Logic Partitioning

Once a RIA is deployed, the uniPaaS platform automatically partitions the logic of the application according to the nature of each operation that the platform is about to execute. While running on the client, the client will automatically turn to the server whenever it encounters a server-side only operation. And while on the server side, the application will turn to the client whenever a client-side only operation is encountered or whenever the server-side activity is completed. The automatic partitioning is the fundamental feature of the platform, which enables the defining of logic without breaking the logic into client/server/communication modules.

3.5 Performance Aware Development

To prevent applications from producing too many client/server roundtrips, the Studio offers a variety of tools that let the developer fine-tune and optimize the application performance. Though the task logic is unified and both client and server-side logic are written within the same editor, the Studio denotes each line as being either client side, server side, or neutral to any side according to the nature of the operation or the use of the defined entity. Moreover, the syntax check facility of the Studio may recommend that the developer reconsider specific logic units for possible performance improvement bearing modifications.

3.6 Transparent Context Management

A uniPaaS RIA developer is fully relieved from any explicit definition and maintenance of the user sessions and user contexts. The uniPaaS platform automatically identifies the context of each request and implicitly maintains the context to follow the current session of the end-user. The developer can define and utilize context specific information and further enhance the personalization of the execution with a set of very easy to functions.

3.7 Integrated Form Editor

The user interface design of the RIA programs is achieved by directly designing the form in a WYSIWYG manner using the integrated form editor of the uniPaaS Studio.

3.8 Local Resources Accessibility

Being a completely independent .NET application, the client-side module can directly access the local client machine to invoke local applications, interact with local devices, retrieve files, access the OS environment, and so forth. This option adds an additional value to the result application by making it more adaptable to client-side needs and by enabling it to keep information persistent on the client machine.

3.9 Tight Application and Data Security

One of the greatest values of a uniPaaS RIA implementation is the enhanced security measures it offers. Compared to traditional client applications where sensitive resources such as data can be directly accessed by the user using other applications, which may lead to malicious use of the data, the uniPaaS RIA keeps all sensitive resources away from any direct access by the end-user. Compared to browser-based RIA solutions, the uniPaaS RIA proprietary client module prevents end-users from tapping into the internal communication between the client and the server.

3.9.1 HTTPS Support

In addition to the internal security mechanism of uniPaaS RIA and uniPaaS in general, it is also possible to deploy a uniPaaS RIA using a secured HTTP layer.

4 Software-as-a-Service (SaaS)

uniPaaS' support for lean and flexible enterprise applications goes beyond the mere introduction of RIA, but also addresses the requirements and management of implementing an internet application in a form of multi-tenancy, which is the key principle of providing enterprise applications in a SaaS model.

4.1 SaaS-Enabled Application Platform (SEAP)

uniPaaS allows for leveraging any internet-based application into a SaaS-enabled application by providing multi-tenancy support at the platform level, relieving the developer from resorting to complex tenant-aware application design and development.

4.2 Multi-Tenant Support

4.2.1 Tenant Encapsulation

uniPaaS makes sure that for a single implementation of a SaaS-type application, each tenant will be fully encapsulated and that every tenant specific characteristic is maintained and governed by uniPaaS Management and Monitoring facilities.

4.2.2 Data Space Sharing and Isolation

The platform level support for multi-tenancy enables the application vendor to design an application without taking special heed of multi-tenant design and can remain focused on the application's basic design. The uniPaaS platform will seamlessly turn the application into a multi-tenant application by transparently directing the application to the tenants' data space and by having each tenant served by an independent process. Nevertheless, the uniPaaS platform enables the application vendor to fine-tune the tenant encapsulated environment in a way that utilizes Data Space that is shared and common between multiple tenants.

4.2.3 Customization

The environment-driven and component-based architecture of the uniPaaS platform enables easy customization of every part of the application, thereby achieving a singular application deployment that serves multiple tenants where each tenant gets to experience a tailored application.

4.2.4 Local or Remote Hosting

The uniPaaS platform as a whole can be deployed on a cloud, using the facilities of a uniPaaS hosting service provider, independently by the application vendor, or even by the end-customer.

The flexibility of the platform to be deployed as a whole in any location offers the application vendor the flexibility to choose the most appropriate deployment models and commercial models.

4.2.5 Extreme Scalability

Application tenants are not bound to specific machines to support extreme usage surges. A redundant array of servers set for the disposal of the application are dynamically utilized for any tenant, according to the capacity required to serve the tenant at any given moment.

4.3 Management & Monitoring

4.3.1 Usage Metering & Activity Monitoring

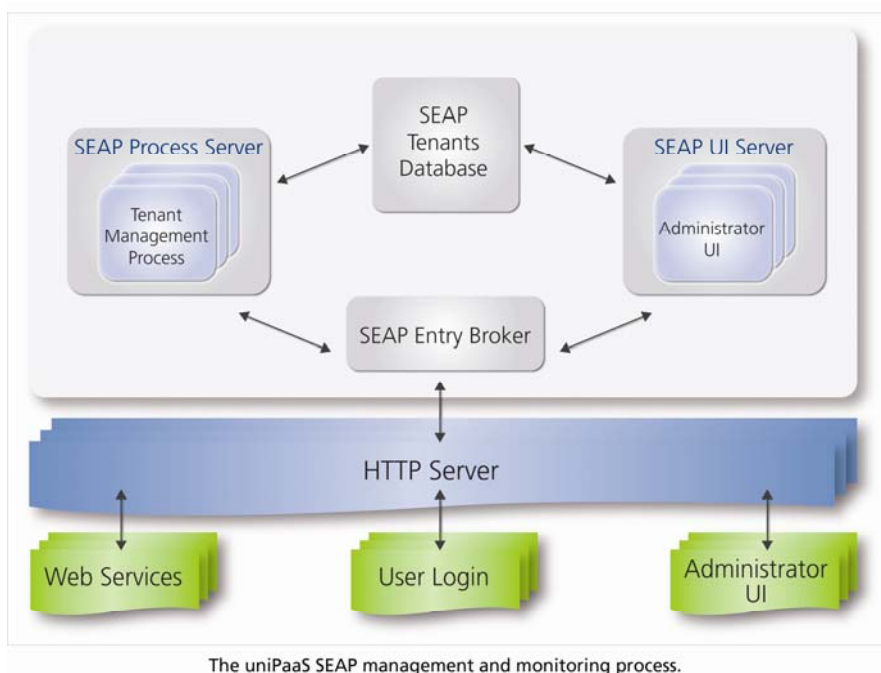
The application vendor can easily set for every application and for every tenant of an application, individual metrics parameters. These metrics parameters are tightly monitored by uniPaaS' SEAP Management and Monitoring facility.

4.3.2 Tenant Property Manipulation

At any given time, the configuration set for each tenant can be reconfigured to reflect any new characteristic of the tenant. All modifications are done without affecting the up-time of the application.

4.3.3 Interactive and Non-Interactive Interfaces

The Management and Monitoring server supports both interactive consoles and Web service-based APIs, supporting the full-range functionality required for proper multi-tenant management and monitoring.



5 Full Client Applications

uniPaaS supports the development and deployment of full client applications offering the ability to produce highly capable client modules servicing standalone applications and multi-users applications.

5.1 Supports 1, 2, and 3 Tier Architectures

A uniPaaS full client application can run as a standalone application utilizing standalone or even memory stored databases or run as a multi-user enterprise application that consumes and manipulates centralized and shared corporate data. The functionality of a full client application can be further fine-tuned by expanding its architecture to be served by an additional tier designed for logic [partitioning](#).

5.2 Direct Data Access

The full client uniPaaS application benefits by having the ability to directly connect to either local or centralized databases. Moreover, a full client application can directly access local XML-based data to further enhance the client capabilities by offering application data that is persistent on the client side.

5.3 Integrated End-User Report Generator

A full client application offers the end-user an integrated report generator by which the end-user can easily define and generate customized reports using highly intuitive wizards and editors.

5.4 Partitioning

uniPaaS full client applications can utilize the power of centralized processing by invoking centralized designated servers to perform complex or heavy tasks.

6 Versatile Web Applications

With uniPaaS application vendors can develop and offer any type of a Web-based application by utilizing the high productivity of the uniPaaS engine. The generic form of manipulating any type of Web response enables the developer to produce any style of Web-based application starting with plain HTML pages, through Ajax, to Flash. uniPaaS enables the production of any type of Web applications ranging from e-Commerce through Mashups and Social networking to Enterprise portals.

6.1 Flexible HTTP (Markup Language) Interface Handling

The uniPaaS-proven Merge technology and XML Views enable the application to respond to any HTTP-based request and to respond with dynamic content in any XML or other markup language.

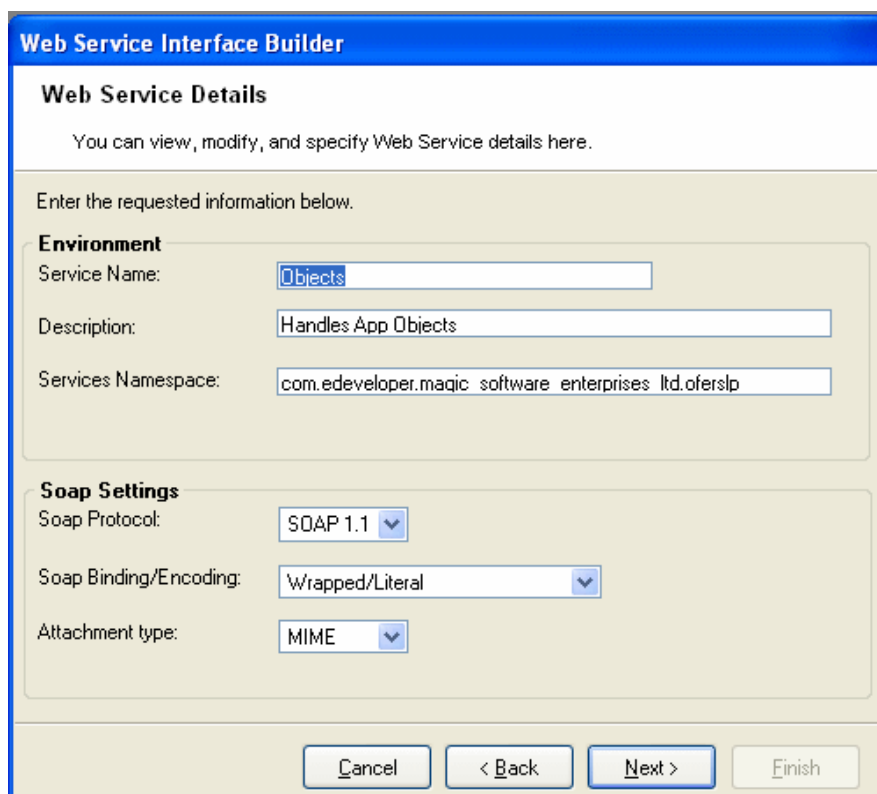
The Merge technology enables the developer to configure any response markup language template to be easily constructed at runtime by having real data merged into the template, thereby producing the actual live response. Using this technology, the developer can respond to any web client technology invoking HTTP based requests.

7 Service Provisioning

In addition to a variety of client-based applications, uniPaaS also supports non-interactive applications or any third-party applications by providing data manipulation and business processes execution in a form of services.

7.1 SOA – Web Services

Any non-interactive uniPaaS task can be easily transformed into a Web Service method by including it in a Web Service interface definition (WSDL). In this way, any uniPaaS task can be easily invoked from any third-party application in the form of a Web service.



Web Service Interface Builder

Web Service Details

You can view, modify, and specify Web Service details here.

Enter the requested information below.

Environment

Service Name:

Description:

Services Namespace:

Soap Settings

Soap Protocol:

Soap Binding/Encoding:

Attachment type:

Figure 8: An initial setup screen of the Web Services Interface generator.

8 About Magic Software Enterprises

Magic Software Enterprises Ltd. (NASDAQ: MGIC) is a leading provider of multiple-mode application platform solutions – including Full Client, Rich Internet Applications (RIA) or Software-as-a-Service (SaaS) modes - and business and process integration solutions. Magic Software has offices in 10 countries and a presence in over 50, as well as a global network of ISV's, system integrators, value-added distributors and resellers, and consulting and OEM partners. The company's award-winning code-free solutions give partners and customers the power to leverage existing IT resources, enhance business agility and focus on core business priorities. Magic Software's technological approach, product roadmap and corporate strategy are recognized by leading industry analysts. Magic Software has partnerships with global IT leaders including SAP AG, salesforce.com, IBM and Oracle. For more information about Magic Software Enterprises and its products and services, visit www.magicsoftware.com.

Magic Software is a subsidiary of Formula Systems in the Emblaze Group of companies.