uniPaaS V1.9i for UNIX Platforms

Release Notes

We are proud to introduce **uniPaaS V1.9i for UNIX Platforms**. Read the information in this document to find out more about this uniPaaS version.

Introducing Magic Software's uniPaaS V1.9i for UNIX Platforms

We are delighted to announce the launch of a new release of Magic Software's uniPaaS SaaS-Enabled Application Platform.

uniPaaS allows organizations to quickly and cost effectively enjoy all the benefits of Rich Internet Applications (RIA) and Software-as-a-Service (SaaS) applications, whether on-premise, or on-demand.

Based upon a unique, unitary development paradigm, uniPaaS gives the power to quickly develop, enhance, and deploy business applications under multiple deployment models and at a fraction of the cost and time compared to conventional .NET or Java environments.

uniPaaS Web Page

Make sure to visit our uniPaaS Web page, where you can view and download various documents.

uniPaaS Licensing

In addition to the new unitary development and deployment paradigm for RIA and SaaS, uniPaaS also supports any previous Magic Software editions and forms of development and deployment. However, in order to maintain your former development and deployment capabilities, you need to obtain new uniPaaS licenses that reflect your current eDeveloper V10 licenses.

To obtain uniPaaS licenses, please contact your local Magic Software representative.

Migrating from eDeveloper V10.1 to uniPaaS

Migrating an eDeveloper V10.1 application to uniPaaS is quite straightforward and no explicit migration procedure is required.

You can directly access and open eDeveloper V10 applications from the uniPaaS Studio and Server engines.

Migrating from eDeveloper V9.4 to uniPaaS

Migrating an eDeveloper V9.4 application to uniPaaS is fast and easy.

uniPaaS provides a collection of wizards to easily migrate your application, INI settings, and interface builder data.

In the Migration subfolder of the product, located in the Start menu of your desktop, you will find a shortcut to each of the available conversion wizards.

We recommend reading the Migration chapter in the *uniPaaS Help* before migrating your eDeveloper V9.4 applications.

Compatibility

For more information about the various platforms on which uniPaaS has been checked for operation by Magic Software Enterprises, refer to the **Compatibility Guide.pdf** file provided with this installation.

Known Issues

When deploying a mobile application, the **appname.CAB** file created by the Rich Client Deployment Builder, needs to be renamed to **appname.cab** (lower case).

Installing uniPaaS

Pre-Installation

- During the installation, several user-environment files are overwritten. Therefore it is best to back up the following files before starting the installation process:
  .cshrc, .profile, .bash_profile (applies to Linux only)
- If you already have a previous uniPaaS server version installed, it is best to install the product using a different user name.

Installation Steps

1. Create a new user. (The installation should be performed using a non-root user.)
2. Log in as the new user.
3. Uncompress the installation file (**uniPaaS_*xxxx*.<platform>.tar.Z**) using the local uncompress utility or a compatible utility, such as gunzip).
4. Run the command from **$HOME directory: tar xvf <installation file>**.
   The installation file name is **uniPaaS_*xxxx*.<platform>.tar**.
5. Run the **./unipaasinstall** command and enter the requested information.
   Note: The RedHat 7.x cgi-bin directory is located in /var/www/cgi-bin.
6. After the installation has been successfully completed, run the **$HOME/sbin/mgroot.sh** file as a root user. This script copies uniPaaS files that should be accessed by your Web Server.
7. To set up an Apache Web Server, append the **$HOME/web_utils/magic.conf** file to the Apache configuration file (**httpd.conf**) and restart the Apache Web Server.
8. Log out from the new user and log in again to enable the new environment settings.
9. If you need to uninstall the product, delete the user home directory created for the installation. For a complete removal, delete the files copied by the **$MAGIC_HOME/sbin/mgroot.sh** script and remove the changes that were applied to the web server.
10. **For Linux installations:** When installing on RedHat Advanced Server 3.0 (or more recent versions) the following commands should be executed:
    a. Log in as the new user.
    b. Execute: cd $MAGIC_HOME/lib
    c. Execute: ln -s /usr/lib/libssl.so.0 libssl.so.2
    d. Execute: ln -s /usr/lib/libcrypto.so.0 libcrypto.so.2

Post Installation

It is necessary to define for uniPaaS that a specific gateway must be loaded by pointing to a variable that contains a DB number. The DB number points to a specific executable that is the relevant gateway.
In UNIX operating systems, an environment variable points to the executable, which should be used for a specific gateway.
For example:
MAGIC_ DB_ 14_ DRIVER=$ MAGIC_ HOME/ bin/ mgoracle10
where the number 14 refers to the DB number.

Installation Components

- uniPaaS Server (bin/unirte)
- uniPaaS Broker (broker/unirqbroker)
- uniPaaS command line requester (broker/mgrqcmdl)
- uniPaaS gateway 10g and 11g (bin/mgoracle10, bin/mgoracle11) – versions are platform specific
- uniPaaS gateway for DB2 UDB Version 8.1 (bin/mgdb2) – for AIX and Linux platforms only
- uniPaaS gateway for ODBC (bin/mgodbc) – for Linux platform only
- uniPaaS memory gateway (bin/mgmemory)
- uniPaaS CGI requester (cgibin/mgrqcgi019)
- uniPaaS requester for Apache Web Server (cgibin/mod_V2_mgrequest019.so , cgibin/mod_V2.2_mgrequest019.so)
- uniPaaS UDF/UDP examples (userproc directory)
- uniPaaS Web utility files used for Browser Client support (web_utils directory)
- uniPaaS license server (FlexLM 7 in license directory)
- uniPaaS Hangul support (language/mglocal.kor)
- uniPaaS SNMP sub-agent (snmp/mgsnmp.so)
- uniPaaS Messaging component (messaging/messaging.mff)
- Systinet server for Java (web_services directory)

This product includes software developed by the Apache Software Foundation (http://www.apache.org/).

Note:
- If the installation fails, it is best to delete all files in **$MAGIC_HOME** and start a new installation from the beginning.
- A remarked entry (meaning that it's preceded by semicolon) named **MGLOCAL**, which points to the Korean support library **mglocal.kor**, exists in the **mgenv** file. This file is required for proper functioning of browser-based applications when using Korean/Hangul characters

Starting the uniPaaS Server and Broker

To run the uniPaaS Server:

1. Start the license server by running the **$HOME/license/mglmstart** script.
2. Use the **unitre.sh** script or invoke the **unirte** executable file. By default uniPaaS uses the INI file specified in the **MGENV** environment variable.
3. For the non-default INI file, use: **unirte –ini=<ini file> &**
4. For an additional INI file, use: **unirte –ini=<ini file> @<additional ini> &**

Several scripts exist in the **sbin** directory to simplify the uniPaaS server administration:

startb	Start the uniPaaS Broker
stopb	Stop the uniPaaS Broker (and Server engines)
stopm	Stop all uniPaaS Server engines connected to the uniPaaS Broker
checkm	Check which uniPaaS Server engines are connected to the uniPaaS Broker

Note: The **stopb** and **stopm** scripts require supplying the broker password, as shown in the following example:

stopb –password=<broker supervisor password>

Note:	The uniPaaS log file is created for each server you start. Its name is determined by the ExternalLogFileName entry in the MAGIC.INI file.

uniPaaS Requesters

The MGREQENV environment variable points to the MGREQ.INI file used by the uniPaaS Server, the uniPaaS Broker, and the uniPaaS command line requester. (The installation sets MGREQENV = $HOME/etc/ MGREQ.INI).

To send a request to a uniPaaS Server on UNIX from an Internet Browser, there are two types of requesters:

1. The uniPaaS CGI requester (mgrqcgi019):

   http://<server_name>/cgi-bin/mgrqcgi019?appname=example1&prgname=prog1

2. The uniPaaS requester for Apache (mod_V2.2_mgrequest019.so):

   http://<server_name>/mgrequest019?appname=example1&prgname=prog1

Additional Settings

The following settings in the MGREQ.INI file affect the requester execution.

- RetryMainTime
- KeepAlive

File Names

In Windows platforms, files can be referred to by either a URL or by file name, relatively or by full path/URL.

In non-Windows platforms, such as UNIX platforms, files can be referred to by a full URL only. Any reference to a file name with a slash (/) is considered to be a path name, either full or relative.

Examples:

/etc/home1/a.jpg (full path)

http://myserver/myalias/a.jpg (full URL)

myalias/a.jpg – is considered to be a relative path name, not a relative URL.

Colors

To use colors properly on UNIX platforms, you must define all the colors that are used as non-system colors. The easiest way to do this is to access the color file in the uniPaaS Studio and define the colors accordingly.

External Code Pages

When installing uniPaaS on UNIX platforms, the **ExternalCodePage** ini setting is set to 1252 (Windows 1252 is the Western European code page). This setting must be modified for any-non Western European languages, such as Hebrew or Thai, since it affects Unicode to ANSI conversions.

Apache Requester Installation and Configuration

Apache Module Requester Setup

uniPaaS 1.9i includes requesters for the Apache Web Server version 2.0 and version 2.2. The requester module **mod_V2.2_mgrequest019.so** should be placed in the **modules** directory of the Apache installation (default: /usr/local/httpd/modules) with execute permissions.

The installation includes the following requesters:

mod_v2_mgrequest019.so to be used with Apache 2.0

mod_V2.2_mgrequest019.so to be used with Apache 2.2

1. Add the following lines to the Apache configuration file, **httpd.conf**.

```
LoadModule mgrequest019_module   modules/mod_V2.2_mgrequest019.so

<Location /mgrequest019>
   SetHandler mgrequest019-handler
</Location>

SetEnv MGREQ_INI_PATH  <directory>
```

2. **For AIX:** Add the **$MAGIC_HOME/lib** to the **$LIBPATH** environment variable.
   For Linux and Solaris: Add the **$MAGIC_HOME/lib** to the **$LD_LIBRARY_PATH** environment variable.
3. Restart the Apache Web server.

The Apache requester is configured using the MGREQ.INI file. The directory location of the MGREQ.INI file is specified by the **MGREQ_INI_PATH** setting in the Apache configuration file, **httpd.conf**.

Example

SetEnv MGREQ_INI_PATH /usr/local/httpd/conf

The Apache requester uses the /usr/local/httpd/conf/ MGREQ.INI file.

To use this requester, call uniPaaS using a URL, such as:
http://server/mgrequest019?appname=...

You should also modify the MAGIC.INI file to read: **InternetDispatcherPath= /mgrequest019**

Using an Apache Web Server with a Non-Default Port

To use Apache with a non-default port (port number other than 80), change the setting shown below in the MAGIC.INI file:

InternetDispatcherPath= http://server:port/cgi-bin/mgrqcgi019

instead of

/cgi-bin/mgrqcgi019

Platform-specific Information

AIX

The uniPaaS 1.9i Server for AIX should be used with AIX 5.3 or with other more recent operating systems that are backward compatible.

The Oracle gateway should be used with the Oracle 10g client.

The uniPaaS DB2 gateway for AIX should be used with the DB2 Version 8.1 client.

The WebSphere MQ 5.3 client/server is required for working with the MQ messaging capabilities.

JRE 1.6 is required for working with Java integration capabilities.

Apache 2.0.45 (or a more recent version) is required in order to use the Apache 2 requester.

Solaris

The uniPaaS 1.9i Server for Solaris should be used with Solaris 8 or with other more recent operating systems that are backward compatible.

The Oracle gateway should be used with the Oracle 10g or Oracle 11g client.

The WebSphere MQ 5.3 client/server is required for working with the MQ messaging capabilities.

JRE 1.6 is required for working with Java integration capabilities.

Apache 2.0.45 (or a more recent version) is required in order to use the Apache 2 requester.

Linux

For Intel processors only, Linux requires Kernel 2.4.20-8smp (and up) and GLIBC 2.3.2-11.9.

The Oracle gateway should be used with the Oracle 10g client.

The Websphere MQ 5.3 client/server is required for working with the MQ messaging capabilities.

JRE 1.6 is required for working with Java integration capabilities.

Apache 2.0.43 (or a more recent version) is required in order to use the Apache 2 requester.

Gateway-specific Information

To enable the use of a particular gateway, remove the # sign from the corresponding entry in the **$MAGIC_HOME/etc/mgenv** file.

When using the Oracle gateway, make sure that **ORACLE_HOME** and **ORACLE_SID** are set in the **$MAGIC_HOME/etc/mgenv** file, and that the environment variable **LD_LIBRARY_PATH** (or **LIBPATH** for AIX) includes the **$ORACLE_HOME/lib** directory.

When using the DB2 gateway, make sure that **DB2INSTANCE** is set in the **$MAGIC_HOME/etc/mgenv** file.

ODBC Gateway on the Linux Platform

General Information

Gateway name: **mgodb**c

Required software: This gateway works with the **UnixODBC** ODBC manager.

It was tested with the following database gateways:

MySQL MyODBC driver (libmyodbc-<ver>.so) – to access MyODBC software and for more information on this particular driver refer to http://www.mysql.com

Installation and Setup Instructions

1. Uncomment the entry **MAGIC_DB_20_DRIVER** in the **mgenv** file. Uncomment means to remove the semicolon preceding the entry.
2. Install the **UnixODBC** ODBC manager, this product can be downloaded from: http://www.unixodbc.org. Follow the online instructions to generate the ODBC manager.

   Locate the following two files (shared libraries): **libodbc.so.1.0.0** and **libodbcinst.so.1.0.0**

   Copy the files to the directory **$MAGIC_HOME/lib**.
3. In the same directory create symbolic links for the two libraries:

   ln –s libodbc.so.1.0.0 libodbc.so.1

   ln –s libodbcinst.so.1.0.0 libodbcinst.so.1
4. Install the ODBC driver. Refer to the specific driver documentation for installation instructions.
5. Make sure that the libraries have Execute permission. Use the **chmod +x** command to set execute permission.
6. Create a hidden file named .**odbc.ini** in the user's home directory. For example: /usr/magicadm/.odbc.ini. This file is used to configure ODBC DSNs. Refer to the ODBC manager documentation for more explanations regarding the setup of this file.

To help you setup quickly, we have included the following **.odbc.ini** file as an example:

```
[mysql]

Driver   = /usr/magicadm/mysql/libmyodbc-2.50.23.so

Trace    = No

Tracefile= mysql.log

Database = samp_db
```

Each section defines a DSN (Data Source Name). In the above example, there is one defined DSN named mysql. The driver entry in each section should be set to the full path of the ODBC driver. For a list of valid entries and their meanings, refer to the ODBC driver documentation.

Alternatively a general /etc/odbc.ini file can be used.

Setting the Magic Configuration File (MAGIC.INI)

1. Set a uniPaaS database using the Database repository.
2. Copy the database definition in the **MAGIC_DATABASES** section from the MAGIC.INI file on Windows to the MAGIC.INI file on Linux. It is highly recommended to backup the MAGIC.INI file before editing.

Limitations and Recommendations

JMS

Connectivity to messaging servers via JMS is not supported using the provided Messaging component.

Before you can use JMS with the Sun Reference application, the environment variables listed below are needed to run J2EE applications on UNIX platforms:

Variable Name	Values
$JAVA_HOME	Directory where the Java 2 SDK, Standard Edition, is installed
$J2EE_HOME	Directory in which the J2EE SDK 1.3.1 is installed, usually $HOME/j2sdkee1.3.1
$CLASSPATH	Include the following: .;$J2EE_HOME/lib/j2ee.jar; $J2EE_HOME/lib/locale
$PATH	Include $J2EE_HOME/bin

Backups

We highly recommended backing up uniPaaS configuration files, such as MAGIC.INI, MGRB.INI, MGREQ.INI, and license.dat, before modifying them.

Compression

There is no compression when the server is a UNIX platform.

Java Integration

The Java CLASSPATH separator character on UNIX platforms is a colon (:) as opposed to the Windows platform separator character, which is a semicolon (;).

For example: **CLASSPATH = /java/MyClasses:/java/OtherClasses**

For more information, please refer to the Java documentation (Java 2 SDK Tools and Utilities at http://java.sun.com/j2se/1.4.2/docs/tooldocs/tools.html**).**

AIX

The **JAVA_HOME** entry should be set in the MAGIC_JAVA section of the MAGIC.INI file.

For example: **If JAVA_HOME = /usr/java16**

uniPaaS appends **/jre/bin/classic/libjvm.a** in order to find the **libjvm.a** library.

If you encounter problems locating this file you can use the environment variable: **MG_JAVALIB**, which should be set to the absolute path of the library file.

For example: **MG_JAVALIB = /usr/java16/jre/bin/classic/libjvm.a**

The **AIX LIBPATH** variable should include **/usr/java16/jre/bin:/usr/java16/jre/bin/classic**

Solaris

If Java is installed on your server, you should edit the following scripts: **.cshrc** and .**profile**.

The **LD_LIBRARY_PATH** environment variable should include **$JAVA_HOME/jre/lib/sparc/client**

Linux

If Java is installed on your server, you should edit the following scripts: .**cshrc** and **.profile**.

The **LD_LIBRARY_PATH** environment variable should include **$JAVA_HOME/jre/lib/i386/client** and **$JAVA_HOME/jre/lib/i386**

WebSphere MQ

If you are using an MQ client software, you should set the following logical name in the MAGIC.INI file: **WMQ_ModuleName = C**

If you are using an MQ server software, meaning that the MQ Queue manager runs on the same machine as the uniPaaS Server, you should set the following logical name in the MAGIC.INI file: **WMQ_ModuleName = S**

External Procedures

User-defined procedures should be compiled according to this platform specific list:

Platform	Compiler Version & Vendor	c++ Compiler	c Compiler
Linux	gcc version 3.2.2	g++	gcc
Solaris	Sun Studio 11 Software	CC	cc
AIX	IBM XL C/C++ for AIX, V11.1	xlC_r	cc_r

FQDN (Fully Qualified Domain Name)

The broker and enterprise server should bind using a specific network adapter by specifying a FQDN (instead of IP address). The requester layer should translate the FQDN to IP and bind using IP on a specific adapter.

FQDN stands for fully qualified domain name – for example "linuxdev.Magic"

The MGREQ.INI file contains the following entry: BindFirstIPAddress=Y/[N]

Y – During binding to a port, the server will resolve the host name and will bind to the resolved IP address.

N – The server will bind to any IP address (*.port – for backwards compatibility)

To enable a uniPaaS engine and broker to work with a specific network adapter (if there are multiple adapters on a machine):

1. Edit the MGREQ.INI file and enable **BindFirstIPAddress** (= Y) and set **MessagingServer** to **FQDN/port**.
2. Edit the MGRB.INI file and set **MessagingServer** to **FQDN/port**.
3. In the MAGIC.INI file, set **TCP/IP = 2, 30, 1500-2000 /LocalHost=FQDN**.
4. In the MAGIC.INI file, set the **Default Broker** to **FQDN/port**.

The table below shows the binding for the server module:

Port Number	V9.4sp6c-1 and above
Port-No	BindFirstIPAddress=N */Port-No BindFirstIPAddress=Y IP-address/Port-No
Ip Address/Port-No	IP-Address/Port-No

Systinet Installation

Installing Systinet Server for Java (SSJ) can be done via the same user used to install uniPaaS or as a different user. All files pertaining to this installation are in the **web_services** directory.

Prerequisites

1. Install Java using the same version as used in the uniPaaS Studio.
2. Define the **JAVA_HOME** variable as the folder where Java is installed (in the user environment where SSJ will be installed).

Installation

- In the SSJInstallConfig file, change the **installation.destination** entry to the target directory of the installation.

Post Installation

1. Define the **WASP_HOME** variable as the installation directory.
2. Copy the **unissj.jar** and **unirequester.jar** files from the uniPaaS **support** folder to the **$WASP_HOME/lib** folder.
3. Go to the **bin** folder in the installation directory and edit the **server.sh** file as follows:
    a. Add the **$WASP_HOME/lib/unissj.jar** and **$WASP_HOME/lib/unirequester.jar** files to the **classpath**.
    b. After the **classpath**, add: **-Djava.library.path={MAGIC_DIR}/lib**
    c. Add **{MAGIC_DIR}/lib** to the **LD_LIBRARY_PATH**
    d. Add the **MGREQENV** environment variable to point to the same value as in the uniPaaS installation.
4. Run **serverstart.sh** to start the SSJ server.
5. If the uniPaaS environment and Systinet environment are different, do the following in the uniPaaS environment:
    a. Define the **JAVA_HOME** variable as the folder where Java is installed (in the user environment where SSJ will be installed).
    b. Define the **WASP_HOME** variable as the Systinet installation directory.
    c. Copy the **conf** folder from **WASP_HOME** (**$WASP_HOME/conf**) to the **ssj** directory.

Deploying a Rich Client Application

To be able to deploy a Rich Client application on UNIX platforms:

1. The following files and folders are created once you use the Rich Client Deployment Builder:

   `appname\appname.application`

   `appname\appname.publish.html`

   `appname\uniRC_x_y_z_www\` (x,y,z represent the uniPaaS version and www is a unique number representing the specific version)

   `appname\Images\`

   a. Place them in the **uni19RIAApplications/*appname*** alias on the Web server.

   b. Users can access the application from the following URL: http://*appserver*/uni19RIAApplications/*appname*/*appname*.publish.html

2. Add the following into the **httpd.conf** Apache configuration file in this order:

```
AddType application/x-ms-application .application
AddType application/x-ms-application .manifest
AddType application/octet-stream deploy
AddType application/x-msdownload .dll
AddHandler default-handler .jpg .gif .js .txt .bat .msi
```

3. Manually change the **HTTPCompressionLevel** in the application's **publish.html** file to **None**, since there is no compression when the server is a non-Windows platform.
   For example:

```
<body onload="initialize()">
    <xml id="rcExecProps">
         <properties>
            <property key="protocol" val="http"/>
            <property key="server" val="aix51:2261"/>
            <property key="requester" val="/mgrequest019"/>
            <property key="appname" val="frame"/>
            <property key="prgname" val="START"/>
            <property key="arguments" val=""/>
            <property key="envvars" val=""/>
            <property key="UseWindowsXPThemes" val="Y"/>
            <property key="HTTPCompressionLevel" val="None"/>
            <property key="DisplayStatisticInformation" val="N"/>
            <property key="InternalLogLevel" val=""/>
            <property key="InternalLogFile" val=""/>
            <property key="InternalLogSync" val="Session"/>
            <property key="LogClientSequenceForActivityMonitor" val="N"/>
         </properties>
      </xml>
    <table align="center">
```

Fixed Issues

Direct SQL logging (QCR #232292)

The size of a buffer is hard-coded to 1000 characters and therefore, there was no method to return the size of the buffer for Unicode conversions. A new environment variable called MG_MAX_STRING_SIZE, when set to a size larger than 1000, will determine for uniPaaS how many bytes to allocate for logging purposes only.

HTTP – Upload (QCR # 253120)

HTTP upload did not work with Apache requester modules.

Profiler logs (QCR #276492)

In AIX and Linux servers, if the WorkingDir logical name was not passed as a parameter, then the Profiler wrote the file in two locations and the wrong profile name was sometimes used.

Broker – Endless loop (QCR # 308559)

Killing the engine with -9 would end up in an unreachable engine and the broker going into an endless loop.

Functions – FileInfo wrong display (QCR # 496774)

Date and Time of a file were not displayed while using the FileInfo() function if the file name contained a logical name.

Broker – Context count (QCR #725907)

The broker monitor displayed the number of contexts including a terminated context even though it was terminated by the RqCtxTrm function.

Command line requester – Return values (QCR # 733701)

Executing a program that uses Numeric type parameters without an update, showed a Null value when passed using mgrqcmdl.

Broker – Crash (QCR # 734766)

Sending a telnet to the host and port on which the broker is listening, then typing a 4 digit number (1234 <enter>) caused the broker to crash.

.NET – Execution (QCR # 766630)

The Invoke .NET operation did not work.

Profiler logs (QCR #934284)

The Profiler logs showed the wrong total task timings in uniPaaS.

Ini files – Logical names (QCR #935871)

After loading two separate applications with different ini files containing different logical names and calling the first application a second time, the logical names of the second applications were loaded.

Rich client (QCR #999931)

Rich client did not update the record correctly when the update was done by the server after a specific sequence of actions.