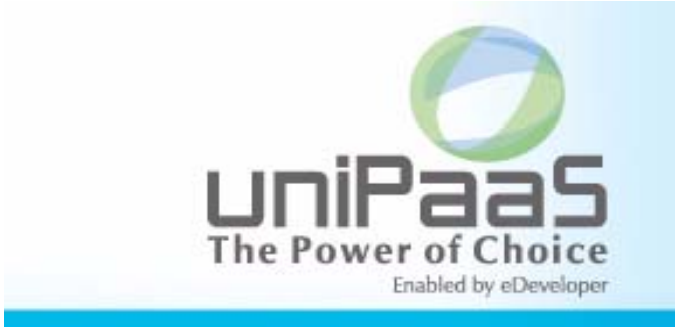

Application Installation Utility Guide

Do it Once. Do it Right
MAGIC SOFTWARE

uniPaaS
August 2010

Contents

3 Installation Files

A Defaults and Preferences

B Dynamic Variables

C Internal Commands

Preface 1

This chapter gives you a brief overview of the Application Installation Utility (AIU) and explains how you can use it to create a customized installation program for any uniPaaS application that you develop.

Previously, end-users of uniPaaS applications needed to first install a particular uniPaaS deployment environment before installing a uniPaaS application. End-users were often unable to install these applications without assistance. Thanks to the Application Installation Utility provided with uniPaaS, your application end-users will be able to install everything they need to run your application using a customized installation program supplied on your application CD.

Overview

The AIU lets you use the uniPaaS installation program as a template that you can modify and customize for you own applications.

Customizing can be done by defining the different sections of the **SetupConfiguration.ini** file supplied with the utility, the installation designer can control the following issues:

- Defining which uniPaaS product and uniPaaS components will be installed.
- Determining the uniPaaS installation defaults, such as the broker name, broker port number, and Web alias name.
- Selecting the user application modules. Those modules are actually a group of files that can be copied during the installation process.
- Determining the operations that can be implemented while copying the user application modules, such as executing commands, creating Web aliases, creating shortcuts, and selecting environment variables.
- Defining the input screen used to gather information from the end-user installer and pass it to the install program for processing.
- Selecting uniPaaS configuration files, such as Magic.ini, Mgrb.ini, and Mgreq.ini files.

How to Start Using the Application Installation Utility

To use the Application Installation Utility:

1. Organize your application into application modules.
2. Copy the application files into the **ApplicationModuleX\Data** directory.
3. Make sure to copy to the **ConfigurationFiles** directory and the required files for the operations. Examples of required files include ini templates, license agreements, installation bitmaps, batch files, and SQL scripts.

4. Edit the **SetupConfiguration.ini** file as described in Appendix E, Using Setup Skins.
5. Execute the **Install.exe** file to run the installation process.

Throughout this book the X symbol is used, to indicate a numeric value.

Installation Execution Sequence

This section provides information about the installation screens and the installation process.

Installation Screens

The screens you have in your installation program depends on how you defined the different sections of the **SetupConfiguration.ini** file, as explained in Appendix 2, INI Sections. The installation screens available are described below in the order that they appear.

Welcome Screen

This screen displays the product name that is defined in the **[Startup]** section under the **AppName=keyword** in your **Setup.ini** file. This built-in screen is displayed first and cannot be changed in the screen sequence.

Magic License Agreement Screen

This built-in screen displays the Magic license agreement. This screen cannot be modified with the AIU.

Application License Agreement Screen

The **License Agreement Screen** is only displayed if the name and location of the text file that contains your application's license agreement are entered under the **[Application_Information]** section for the

ApplicationLicenseAgreementFile keyword in the of the **SetupConfiguration.ini** file, as described on page 21.

Application Input Screen Loop

The installation program displays the user-defined dialog boxes specified in the **SetupConfiguration.ini** file under the **[Application Information]** section for the **InstallScreenOrder** keyword, as described on page 24.

Here are a few examples of how to use the Application Input Screen Loop:

- Select the application components
- Define the root directory
- Select the installation type — Typical or Custom

Application Modules Selection Screen

This screen is displayed only if **Yes** is specified for the **ShowApplicationModulesScreen** keyword, as described on page 44, or when the expression value evaluates to **True**.

This screen shows the end-user a list of the application modules defined in the **SetupConfiguration.ini** file, as explained on page 43. The end-user can select which application module or modules to install.

If you want the end-user to install only one of the application modules, it is recommended to use a radio button. If the end-user can install one or more components, it is recommended to use a check box.

Get Databases Information Screen

This screen requests database information according to the definition set to the **GetDatabaseInformation** keyword in the **[Application_Information]** section, as explained on page 40. The database information fields, depending on the database, are:

- Server name
- Database name
- Directory name

- DSN
- Database Location
- Database Username
- Database Password

You can design up to ten database questions. Each request for database information is based on the database name set for the Database Table keyword in the Magic.ini file and the database type.

For information on the databases supported by uniPaaS, see page 41.

Application Module Input Screen Loop

The installation program shows all the user-defined dialogs specified by the **ScreenOrder** keyword in the **[Application ModuleX]** section of the **SetupConfiguration.ini** file.

- If the **GetDirectoryFromUser** keyword, explained on page 43, is set to **Yes** in the **[Application_ModuleX]** section, the end-user is asked to enter an application module destination location.
- If the **ScreenOrder** keyword has a range of values, the end-user is asked to enter all information needed by the application module, as defined in the **[Screen#]** section.

uniPaaS Engine Installation Directory Screen

The end-user can select the directory where uniPaaS will be installed. You can define the default installation directory under the **[Magic_Information]** section of the **SetupConfiguration.ini** file, as described on page 20.

Note: This screen is shown only if **Yes** is entered for the **GetMagicDirectoryScreen** keyword.

Setup Review Screen

This screen displays a message just before the application installation process begins.

Setup Finish Screen

This screen appears at the end of the installation process. This screen informs the end-user that the installation process is finished.

If locked files were detected during the installation process, the end-user is prompted to reboot. If there are no locked files the user is prompted to review the readme file or to execute another action defined under the **[Finish_Install]** section of the **SetupConfiguration.ini** file.

Depending on what you define, the final screen can also be an **Upgrade** screen or an **Uninstall** screen.

Installation Process

The installation process is described below.

uniPaaS Installation

This process is determined by the selections made by the installation designer and the end-user.

User Data Installation

User data installation is processed in two phases:

- Before and Copy
- After and Copy

The AIU processes the specified application modules for installation as described below.

User Data Installation — Before Copy

This phase executes the commands as defined by the **BeforeInstallExecuteCommands** keyword.

User Data Installation — After Copy

The AIU does the following:

1. Performs text operations as defined for the **TextOperations** keyword.
2. Performs commands as defined for the **AfterInstallExecuteCommands** keyword.
3. Creates shortcuts as defined for the **CreateShortcuts** keyword.
4. Creates environment variables as defined for the **SetEnvironmentVariables** keyword.
5. Creates Web aliases as defined for the **CreateWebAliases** keyword.

INI Sections

2

The **SetupConfiguration.ini** file is the heart of the Application Installation Utility (AIU). Each section of this configuration file contains different settings. The keywords you define within each section determine how your application installation program looks and works.

In this chapter:

[Application_Information#]

This section lets you specify the following settings:

CompanyName

You can enter your company name. The information is essential for registry keys, titles, and identification for the installation program that appears in the **Upgrade** or **Uninstall** screens. The company name is displayed next to the program in the **Add and Remove Programs** screen.

Note: This keyword is **mandatory**.

ProductName

You can enter the product name. The information is essential for registry keys, titles, and identification for the installation program that appears in the **Upgrade** or **Uninstall** screens. The product name is displayed next to the program in the **Add and Remove Programs** screen.

Note: This keyword is **mandatory**.

Version

You can enter the product version number. The information is essential for registry keys, titles, and identification for the installation program that appears in the **Upgrade** or **Uninstall** screens. The version is displayed next to the program in the **Add and Remove Programs** screen.

The format for this keyword is: `X.XXX.XXX`

Note: This keyword is **mandatory**.

HelpLink

You can specify the URL for company support, which is displayed next to the program in the **Add and Remove Programs** screen; for example, www.support.magicsoftware.com.

BitmapFileInstallationIcon

You can specify a bitmap file name and location that can be used to display your application logo at the left side of each installation screen. If you do not enter a bitmap file name and location, your installation program displays the **Installshield** default logo.

Note: You can enter a dynamic variable for this property. For more information about dynamic variables see Appendix B, Dynamic Variables. Make sure to put the file in the **ConfigurationFiles** directory and use the <CONFIG DIR> dynamic variable to locate the file.

ApplicationLicenseAgreementFile

You can specify the name and location of the text file that contains your application's license agreement so that your installation program displays the agreement to the end-user on the **License Agreement** screen.

Note: You can enter a dynamic variable for this property. For more information about dynamic variables see Appendix B, Dynamic Variables. Make sure to place the file in the **ConfigurationFiles** directory and use the <CONFIG DIR> dynamic variable to locate the file.

WelcomeMessage

This keyword lets the install designer to customize the text displayed in the **Welcome Message** screen. In the event that the keyword does not appear in the **SetupConfiguration.ini** file, the setup program shows the following text:

```
The Magic Wizard will install [Product Name] as well as the Magic
Deployment engine on your computer. To continue, click Next.
```

Note: You can insert \n to advance to a new line.

WelcomeTitle

This keyword lets the install designer customize the text displayed in the Welcome Title. In the event that the keyword does not appear in the **SetupConfiguration.ini** file, the setup program shows the following text:

```
Welcome to the Magic Wizard for %P
```

Where **%P** represents the product name.

ReviewMessage

This keyword lets the install designer customize the text displayed in the Review dialog box. In the event that the keyword does not appear in the Setup Configuration.ini file, the setup program shows the default text.

Note: You can insert \n to advance to a new line.

ReviewTitle

This keyword lets the install designer customize the text displayed in the Review dialog box title. In the event that the keyword does not appear in the SetupConfiguration.ini file, the setup program shows the following text:

```
Setup: Ready to Install
```

GetMagicDirectoryScreen

You can enter **Yes** for the end-user to be prompted to specify the directory for uniPaaS Deployment.

If you enter **No**, the default directory will be based on the value entered in the **DefaultDirectory** keyword under the [Magic_Information] section.

Note: The default value is **No**.

ShowApplicationModules Screen

You can enter **Yes** for the Setup Program to display a screen with a list of application modules that the end-user can select for installation.

Note: The default value is **No**.

If you choose **No**, you will not be able to use the Add or Remove Components feature. As a result the end-user will not be able to make modifications to the installation at a later date.

You can also insert an expression to be evaluated during the installation process. It is best to use the expression when you have selected an application dialog for the setup type. When the user selects **Custom**, the expression is set to **True**. For any other user selection, the expression is set to **False**.

For more information, see the Expression Evaluators section on 84.

GetDatabaseInformation

In this setting, you can specify that databases to input during installation. Enter a number value (1,2...) as the identifier of the appropriate database in **[GetDatabaseInformation]** section, as explained on page 40.

CreateMagicShortcuts

You can enter **Yes** for the Setup Program to create uniPaaS Engine standard shortcuts.

Note: The default value is **No**.

InstallScreenOrder

You can enter the screen identifiers in the order of their appearance during the first installation. For example,

```
InstallScreenOrder = 1,3,7,4
```

Where the order of screen appearance is [Screen1], [Screen3], [Screen7], and [Screen4].

Note: There can be up to 20 values entered for this keyword.

UninstallScreenOrder

You can enter the screen identifiers in the order of their appearance during the uninstall procedure. For example,

```
UninstallScreenOrder = 1,3,7,4
```

The order of screen appearance is [Screen1], [Screen3], [Screen7], and [Screen4].

Note: There can be up to 20 values entered for this keyword.

UpgradeScreenOrder

You can enter the screen identifiers in the order of their appearance during the upgrade procedure.

For example,
UpgradeScreenOrder = 1,3,7,4

The order of screen appearance is [Screen1], [Screen3], [Screen7], and [Screen4].

Note: There can be up to 50 values entered for this keyword.

ModifyScreenOrder

You can enter the screen identifiers in the order they appear in Modify mode.
For example,
ModifyScreenOrder = 1, 3, 7, 4

The order of screen appearance is [Screen1], [Screen3], [Screen7], and [Screen4]

Notes: You can enter Modify mode after the product is installed by clicking the **Add or Remove Programs** icon from the Control Panel or by clicking the **Modify** button from the **Welcome** screen.

The behavior of the Modify mode is described below:

- You can enter up to thirty screen identifier entries for this keyword.
- All the information entered during Installation mode can be used in Modify mode. For example, if the user entered information for Screen1, the position of Screen1 can changed by using the **ModifyScreenOrder** keyword.
- It is essential to use the **ModifyScreenOrder** keyword, when you decide not to use the **ShowApplicationModulesScreen** keyword.
- Realize that by using the **ModifyScreenOrder** keyword, some components can be removed or added. The AIU displays the appropriate **Install or Remove** screen for those user components that are about to be changed.
- If there is no screen identifier entries entered for the **ModifyScreenOrder** keyword, and the **ShowApplicationModulesScreen** keyword is set to **No**, the following message appears when the user tries to enter Modify mode:
  `The Install designer has been blocked this option.The setup program will return to the last screen.`

TextOperation

You can use this keyword to make the setup program perform a text operation before or just after the last installation dialog, and before the installation starts to copy the files.

The numeric values for this keyword are separated by commas, for example, 1,2. The text operation information is retrieved from the [TextOperation1] and [TextOperation2] sections. There can be up to sixty values entered for this keyword.

Note: When this keyword is executed, no files are available accept for those under the **<CONFIG_DIR>** section. It's for this reason that the main use of the **TextOperation** keyword is to create the appropriate files before executing the entries entered for the **BeforeInstallExecuteCommands, BeforeUpgradeExecuteCommands**, and **BeforeUninstalolExecuteCommands** keywords.

The Text operation is executed after the screen appears before the files are copied, deleted, or updated, regardless of the installation mode.

ApplicationBuildDirSCR

You can specify a screen identifie,r which must point to a **GetDir** screen, see page 64. The AIU receives information from this screen after processing all the application screens and before the application module selection starts.

Note: You use this keyword to get the root directory of the installation. Once the user chooses the directory, the directory is accessible in other places where you have used the **<BUILD_DIR>** dynamic variable.

This is the preferred way to get to the root directory of the installation. When the **<BUILD_DIR>** variable is assigned to the root directory, the component directories and Magic directory can be searched for by using this keyword. For example, <BUILD_DIR>\Magic displays the Magic directory.

BeforeInstallExecuteCommands
BeforeUpgradeExecuteCommands
BeforeUninstallExecuteCommands

You can specify the numeric identifiers of the commands that are defined in the **[Commands]** section, as described on page 73. For the **BeforeInstallExecuteCommands** keyword, the specified commands are executed before the installation process. For The **BeforeUpgradeExecuteCommands** and **BeforeUninstallExecuteCommands** keywords, the specified commands are executed before the upgrade process. These commands include calling batch and executable files, loading and unloading services, and locking a process before copying files.

Note: These keywords can contain up to **60** command values.

DebugMode

You can specify **True** to display a message before each installation operation.

Note: The default value is **No**.

ReleaseVersion

You can use this keyword to make the AIU load much faster by skipping a number of tests and by not loading the SQL server names in advance. You should specify **Yes** only for the release mode of your software. The default value is **Yes**.

Sample Configuration

```
[Application_Information]
CompanyName = Magic
ProductName = eMerchant
Version = 2.0
BitmapFileInstallationIcon = <CONFIG DIR>\Messey.bmp
GetMagicDirectoryScreen = No
ShowUserComponentScreen = Yes
CreateMagicShortcuts = No
HelpLink = http://www.magicsoftware.com
```

[Magic_Information#]

This section includes general information about the uniPaaS product that is installed.

MagicProductToInstall

You can use this keyword to define the uniPaaS engine that you want to install with your application. The following options are:

- CSRT – Open Client Deployment
- ENT1 – uniPaaS Server
- PART1 – uniPaaS Partition Server
- COMP – You can specify a number of uniPaaS components to install without installing uniPaaS itself.

Notes: You should either use the keyword **MagicProductToInstall** keyword or **TotalOptionalMagicProductsToInstall** keyword in this section. You should not use both of these keywords.

If you use the **TotalOptionalMagicProductsToInstall** keyword, the AIU searches the **[Magic_Components]** section for the uniPaaS components to install.

If you use the **MagicProductToInstall** keyword, the AIU searches the **[Magic_Components]** section for the uniPaaS components to install.

TotalOptionalMagicProductsToInstall

Only one uniPaaS product can be selected for installation. However, By using this option, however, you can choose the correct product according to a condition defined in the **[MagicProduct#]** section.

For this keyword, you can specify a numeric value for the number of uniPaaS products that can be installed. The information about which uniPaaS products can be installed is in the **[MagicProduct#]** section, described on page 33.

Notes: You should either use the keyword **MagicProductToInstall** or **TotalOptionalMagicProductsToInstall** keywords. You should not use both of them.

If you use the **TotalOptionalMagicProductsToInstall** keyword, you must define your section sets. For example, if you specify:
`TotalOptionalMagicProductsToInstall=2`
then you must define these sections as follows:
`[MagicProduct1]`
`[Magic_Components1]`
`[MagicProduct2]`
`[Magic_Components2]`

MagicWebAliasName
MagicPublishedApplicationsAliasName
MagicRIAModulesAliasName
MagicRIACacheAliasName

If you install an Internet requester or the RIA modules, uniPaaS creates Web aliases for the uniPaaS folders.

If you do not specify different alias names, you take the chance that other installations of uniPaaS or of uniPaaS applications may overwrite the Web alias locations of your application.

You can enter a dynamic variable for these properties. For more information see Appendix B, Dynamic Variables.

DefaultDirectory

This is the default directory where the uniPaaS engine is installed on the end-user's host computer.

Notes: This keyword is **mandatory**.

You can enter a dynamic variable for this property. For more information see Appendix B, Dynamic Variables.

ChangeDefaultAssociation

You can enter **Yes** to set the default association of the Mcf and Mff file extensions with the currently installed uniPaaS engine.

Note: The default value is **No**.

MagicIconFolder

Specify the folder where the customized icons are created when you choose to overwrite the default icons.

MagicScriptLocation

You can specify an alternate location for the Magic script directory. You can use it to copy Magic requesters into a Web server.

Note: You can enter a dynamic variable for this property. For more information, see Appendix B, Dynamic Variables.

Sample Configuration

```
[Magic_Information]
TotalOptionalMagicProductsToInstall = 3
ChangeDefaultAssociation = No
MagicWebAliasName = /MyAppWebAlias
DefaultDirectory = <PROGRAMFILES>\test
```

[MagicProduct#]

This section is relevant when you use the **TotalOptionalMagicProductsToInstall** keyword in the **[Magic_Information#]** section.

You define a **[Magic_Product#]** section for each uniPaaS product to be installed together with your application. Each **[Magic_Product#]** section contains the keywords that are described in the section below.

MagicProductToInstall

You can use this keyword to define the uniPaaS engine to install with your application. The following are the options:

- CSRT - Open Client Deployment
- ENT1 - uniPaaS Server
- PART1 - uniPaaS Partition Server
- COMP - You can specify a number of uniPaaS components to install without installing uniPaaS

Note: This keyword is **mandatory**.

Condition

If the condition specified for this keyword evaluates to **True**, the installation program installs the product defined as the **MagicProductToInstall** keyword.

You should be aware of the following restrictions:

- The condition for each uniPaaS product must be unique, and only one **True** value is permitted for all conditions. If more than one condition meets the **True** condition, an error message is displayed.
- If none of the conditions evaluate to **True**, an error message is displayed.

See also the section about using Expression Evaluators on page 84.

Sample Configuration

If the condition value is equal to logical **True**, the check box value is **False** no matter what value is assigned to the **Value** keyword. The radio button cannot be selected. Also, when trying to retrieve information regarding a radio or check box button, the value returned is **False** or **Null**.

If all the control conditions are set to logical **True** in the radio or check box dialog box, the AIU does not display the specific dialog box.

The AIU does not let you move to another dialog box when you are still clicked on the **Next** button in a radio or check box dialog that is not enabled.

```
[MagicProduct1]
MagicProductToInstall = ENT1
Condition = <APPL_MODULE1_IS_INSTALLED> ==

[MagicProduct2]
MagicProductToInstall = COMP
Condition = <APPL_MODULE2_IS_INSTALLED> ==
No && <scr.11.value.YesNo> == Yes
```

[Magic_Components#]

This section includes a list of all the uniPaaS components. By default, the installation program knows which uniPaaS components are about to be installed according to the **MagicProductToInstall** keyword, described on page 33. In this section, you can change the behavior of a default uniPaaS installation by specifying which uniPaaS components will be installed and which will not be installed by defining each component as **Yes** or **No**, or a condition.

If you use the **TotalMagicProductsToInstall** keyword, you should make sure that each uniPaaS product has its own set of components. If you use the **MagicProductToInstall** keyword, you select the components in the

[Magic_Components] section. You should not include a component that is already part of the product definition.

Notes: If you choose a condition, a **True** value installs the components, while a **False** value does not.

Conditions are used for maintenance operations, such as adding or removing components.

If a user is upgrading components and the condition value is set to **False** with the previous value set to **True**, the uniPaaS component will be uninstalled.

You must create a **[Magic_Components]** section for each product defined.

The uniPaaS components that can be installed are categorized as follows:

- uniPaaS Environments
- Middleware
- Gateways
- Requesters
- License Components
- Language Support
- Help

The components available for each group are listed below:

uniPaaS Environments

The settings that are available:

- Browser-Based Deployment = Yes/No/Condition
- Magic Deployment = Yes/No/Condition
- AS400\AS400 = Yes/No/Condition
- AS400\AS400 Host = Yes/No/Condition
- XML Parser = Yes/No/Condition
- Browser Methodology = Yes/No/Condition

- Messaging = Yes/No/Condition

Protection

The options available are:

- Protection\NetHasp = Yes/No/Condition
- Protection\Monitor = Yes/No/Condition
- Protection\HDD = Yes/No/Condition

Middleware

The options available are:

- Middleware gateways\Broker = Yes/No/Condition
- Middleware gateways\J2EE = Yes/No/Condition
- Middleware gateways\SNMP = Yes/No/Condition

Gateways

The gateways available are:

- Gateways\Btrieve = Yes/No/Condition
- Gateways\ODBC = Yes/No/Condition
- Gateways\MS-SQL = Yes/No/Condition
- Gateways\Oracle = Yes/No/Condition
- Gateways\DB2 = Yes/No/Condition
- Gateways\P2K = Yes/No/Condition
- Gateways\MySQL = Yes/No/Condition

Requesters

The options available are:

- Requesters\ISAPI (Microsoft) = Yes/No/Condition
- Requesters\CGI (Other) = Yes/No/Condition

License Components

- License Server = Yes/No/Condition

Language Support

The languages available are:

- Language\English (International) = Yes/No/Condition
- Language\Hebrew = Yes/No/Condition
- Language\German = Yes/No/Condition
- Language\Hungarian = Yes/No/Condition
- Language\French = Yes/No/Condition
- Language\Dutch = Yes/No/Condition
- Language\Polish = Yes/No/Condition
- Language\Portuguese = Yes/No/Condition
- Language\Simplified Chinese = Yes/No/Condition

Other Components

These components are:

- Help = Yes/No/Condition
- Magic PDFs\Install = Yes/No/Condition

Sample Configuration

```
[Magic_Components]
MiddleWare\Magic Broker = <APPL_MODULE1_IS_INSTALLED> == No
Requesters\ISAPI (Microsoft) = <scr.3.value.Radio> == MAPI
Requesters\CGI(Other) = No
```

[BrokerInformation]

This section lets you specify the Magic Broker options available for installation, and is only relevant if the Magic Broker is installed. The **[BrokerInformation]** section has the following settings:

- InstallAsService
- NumberOfBrokersToInstall
- BrokerTCPIPAddress
- ServiceName
- BrokerPassword

InstallAsService

For Windows 2000, 2003, or XP, the Magic Broker can be installed as a service. It is best to install the Magic Broker as a service when installing the uniPaaS server.

Note: The default value is **Yes**.

NumberOfBrokersToInstall

If the Magic Broker is installed as a service, you can determine the number of services for the broker. You can create as many services as the license permits, up to a maximum of four.

ServiceName

A unique name assigned to the Magic Broker service.

BrokerTCPIPAddress

You can choose the default TCP/IP broker address of the first broker. All other broker services addresses are incremented by ten for each service. The Broker service address is used in the Magic.ini file, the Mgrb.ini file, and the Mgreq.ini file.

Note: The default value is **4101**.

The AIU is used by other applications. If two applications use the same TCP/IP broker address there will be problems with their broker definitions. To avoid this you should insert a unique number.

BrokerPassword

This keyword overwrites the default password for the broker in the uniPaaS configuration files.

Sample Configuration

```
[BrokerInformation]
InstallAsService = Yes
ServiceName = eMerchant Broker
BrokerTCPIPAddress = 4003
BrokerPassword =
```

[GetDatabaseInformation#]

The Application Installation Utility lets you define up to twenty separate database entries. Each database entry has a defined database type, such as MSSQL, DB2, Oracle, Informix, and so on, and each entry must also have its own **[GetDatabaseInformation#]** section. You can determine which database properties are displayed by specifying the **[GetDataBaseInformation#]** section number in the **GetDatabaseInformation** property under the **[Application_Information#]** section as explained on page 23.

The purpose of this screen is to update the Magic.ini file with the relevant information in the **[Databases#]** section, as well as setting the correct values for dynamic variables to be used later.

Name

Enter the database name as specified in the Magic.ini file.

Note: This keyword is **mandatory**.

DatabaseType

You can enter the database type assigned to the database, as specified in the database section of the Magic.ini file. You can select from the following gateways:

- Pervasive
- MSSQL
- DB2
- Oracle
- MySQL
- ODBC
- DB2/400

Note: This keyword is **mandatory**.

DatabaseName

You must enter the database name for the gateway to connect to an MSSQL or DB2 database. This keyword is relevant only for MSSQL and DB2 databases.

Note: You can enter a dynamic variable for this property. For more information see Appendix B, Dynamic Variables.

Condition

You can determine whether the database screen wis shown to the end-user. This keyword can also affects the update process of the Magic.ini file.

For more information, see the section about using Expression Evaluators on page 84.

DatabaseServerName

This keyword is only relevant for DB2/400, MSSQL, and DB2 databases.

Note: You can enter a dynamic variable for this property. For more information, see Appendix B, Dynamic Variables.

Alias

This keyword is relevant only for Oracle and Informix databases.

Note: You can enter a dynamic variable for this property. For more information, see Appendix B, Dynamic Variables.

DirectoryName

This keyword is relevant only for DB2/400 databases.

Note: You can enter a dynamic variable for this property. For more information see Appendix B, Dynamic Variables.

DSN

this keyword is relevant only for the ODBC database.

Note: You can enter a dynamic variable for this property. For more information, see Appendix B, Dynamic Variables.

Location

This keyword lets you specify the location of the database files. This is only relevant for the Pervasive database.

Note: You can enter a dynamic variable for this property. For more information, see Appendix B, Dynamic Variables.

UserName

You can enter the default application's user name. This keyword is relevant for all database gateways except for the Pervasive database.

Password

You can enter the default application password. This keyword is relevant for all database gateways except for the Pervasive database.

Sample Configuration

```
[GetDataBaseInformation1]
Name = eMerchantSQL
DatabaseType = MSSQL
DatabaseName = eMerchantSQL
DatabaseServerName = erezlp
UserName = sa
Password =
```

[ApplicationModule#]

An application module contains a set of files that you can use for creating installation operations, such as Web aliases and environment settings. This section lets you specify different kinds of installation procedures for different

requirements, such as screen field labels and screen order. You can define up to 25 different kinds of application modules.

You must specify a unique numeric identifier from 1 to 25 for each application module. In the application module, you design the actual application installation details, such as where the files are copied to, how they are copied, and which Web aliases and shortcuts are required for installation.

An application module is a set of files that are copied to a single location on the target computer. Application modules without files are invalid.

Notes: When upgrading an application, if there are application files that should not be updated during an upgrade procedure, you should place them in a different application module with different copy options, and set the **CopyFilesDuringUpgrade** keyword to **No**.

Blank directories that you place in the Application Module directory will not be copied to the target location. You can place an empty file in a directory to make sure the directory does copy over.

Name

Specify the name of the Application Module. The name identifier appears in the Application Module selection list, the registry, and in error messages.

Note: This keyword is **mandatory**.

ShowAtApplicationModuleScreen

Enter **Yes** to display the Application Module name in the Application Module Selection list. The screen is be displayed according to the definitions in the **ShowApplicationModulesScreen** keyword.

- If you to set this keyword to **Yes,** the application module is displayed on the screen.

- Only applications that have this keyword set to **Yes** are displayed in the **Setup Review** screen.

Note: The default value is **No**.

Condition

The Application Module selected for operation depends on the value of this property. You can insert Yes, No, or an expression. For more information, see the Expression Evaluators section on page 84.

Note the following keyword behavior:

1. The Condition keyword must be handled carefully. Be aware that the condition that you have assigned will be calculated again in the Upgrade mode, and the Add and Remove Components mode.
2. You should us the Condition keyword only if:
   a) It is a component that is dependent on another component. For example, if only Component 1 is selected, <APPL MODULE IS INSTALLED == Yes
   b) You are not using the **ShowApplicationModulesScreen** keyword and you want to provide user options in a Radio Button screen. For example, <scr.3.value.Radio> == SMTP
3. Be aware that the conditions that you have selected will be checked again in other modes. For example, if you use the condition as described above in 2a, and in Upgrade mode you insert Install = No or False for the condition of Component 1, Component 2 will be removed from the computer.

Note: A false value in a condition for a component that is being installed will cause the component to be removed from the computer.

Install

You can enter **Yes** to install the Application Module. Specify **No** if you do not want the Application Module to be installed.

Note: The default value is **No**.

Description

The Application Module description is displayed in the Application Module selection list.

GetDirectoryFromUser

Enter **Yes** to prompt the end-user to specify the destination directory for the files in the Application Module.

If you enter **No**, the Setup Program uses the destination location entered for the **Destination_Location** keyword.

Note: The default value is **No.**

Destination_Location

You can enter the default directory where the Application Module files are copied.

Note: This keyword is **mandatory.**

You can enter a dynamic variable for this property. For more information, see Appendix B, Dynamic Variables.

CopyDirective

You can specify the procedure of how the files are to be copied to the designated directory. You can select more than one operation by using the '|' character.

Note: The default value is **No**:

COMP_UPDATE_SAME|COMP_UPDATE_DATE|INCLUDE_SUBDIR

The options for the **CopyDirective** keyword are:

COMP_NORMAL - Copies the Application Module files to the designated directory, and updates any existing files with the same name regardless of the date, time, or version.

COMP_UPDATE_SAME - Updates the files even if the date, time, or version of the source file is identical to the target file.

COMP_UPDATE_DATE - Updates the target files by file date and time. The target file is updated only if the source file is of more recent time or date.

COMP_UPDATE_VERSION - Updates the target files by file version. The target file is updated only if the source file is a more recent version. If neither the source nor the target file have a version number, time and date are used for comparison. If only one of the files have a version number, InstallShield recognizes the file with the version number as the newer file.

SELFREGISTER - Implements the self-registration process immediately, when using the non-batch method of installing self-registering files. Always use SELFREGISTER with the SHAREDFILE option by connecting them with the logical OR operative.

SHAREDFILE - Combines both shared and locked file handling by making the XCopyFile treat all files as shared, and records locked DLL and EXE files for updating when Windows or the system restarts.

The SHAREDFILE option increments the registry reference counter by one when the file exists in the target directory and has a reference counter greater than 0. If the shared file does not exist in the target directory and has no reference counter, InstallShield creates the counter and sets it to 1. If the shared file already exists in the target directory but has no reference counter,

InstallShield creates the counter and initializes it to 2 as a precaution against accidental removal when uninstalling.

LOCKEDFILE - Records locked DLL and EXE files for update when the host computer is restarted. A locked file is one that is in use by the application or system at the time InstallShield is accessing or updating it. You cannot use the LOCKEDFILE option together with the SHAREDFILE option.

EXCLUDE_SUBDIR - This setting does not allow subdirectories contained in the source path to be copied during the installation process.

INCLUDE_SUBDIR - This setting allows subdirectories contained in the source path to be copied during the installation process.

CreateShortcuts

You can use this property to define the shortcuts that can be created while installing this Application module. Specify the numeric identifier of the shortcut defined under the **[Shortcut#]** section, as explained on page 71.

Note: You can enter up to 20 shortcut identifiers for each Application Module.

CreateWebAliases

You use this property to define the Web aliases that are created while installing this Application Module. Specify the numeric identifier of the Web alias defined under the **[WebAlias#]** section, explained on page 69..

Note: You can enter up to 20 Web alias identifiers for each Application Module.

CopyFilesDuringUpgrade

Enter **Yes** if you want the installation program to copy Application Module files during an application upgrade.

Note: The default value is **Yes**.

ScreenOrder

This keyword determines the order of the screens defined under the **[ApplicationScreen#]** section of the **SetupConfiguration.ini** file. Enter the screen identifiers in the order you want them to appear during installation.

For example,
`ScreenOrder = 1,3,7,4`

Where the order of screen appearance is `[Screen1]`, `[Screen3]`, `[Screen7]`, and `[Screen4]`.

BeforeInstallExecuteCommands

You can specify the numeric identifiers of the commands defined in the **[Commands]** section, as explained on page 73, in the order that they will be executed before the application module files are copied over. These commands can include calling Batch and EXE files, loading services or unloading services, and unlocking a process before updating.

Note: There can be up to 20 entries for this keyword.

AfterInstallExecuteCommands

You can specify the numeric identifier of the commands defined in the **[Commands]** section, as explained on page 73, in the order that they will be executed after the application module files are copied. These can include

commands such as inserting information to the registry and executing SQL commands.

Note: There can be up to 20 entries for this keyword.

BeforeUpgradeExecuteCommands

You can specify the numeric identifier of the commands defined in the **[Commands#]** section, as explained on page 73, in the order that they will be executed before upgrading the Application Module files. These commands can include the removal of locked files.

Note: There can be up to 20 entries for this keyword.

AfterUpgradeExecuteCommands

You can specify the numeric identifier of the commands defined in the **[Commands#]** section, as explained on page 73, in the order that they will be executed after upgrading the Application Module files. These commands can include the execution of SQL commands.

Note: There can be up to 20 entries for this keyword.

BeforeRemoveExecuteCommands

You can specify the numeric identifier of the commands defined in the **[Commands#]** section, as explained on page 73, so that these commands will be executed before the Application Module files are deleted during the uninstall procedure.

Note: There can be up to 20 entries for this keyword.

AfterRemoveExecuteCommands

You can specify the numeric identifier of the commands defined in the **[Commands#]** section, as explained on page 73, so that these commands will be executed after the Application Module files are deleted during the uninstall procedure.

Note: There can be up to 20 entries for this keyword.

SetEnvironmentVariables

You can specify the numeric identifier of the environment variables defined for each **[Environment#]** section, as explained on page 77.

TextOperation

You use this property to make the Setup program perform a text operation on an Application Module.

The format of this property is numeric values separated by commas.
For example: 1,2. In this example the text operation information is taken from the **[TextOperation1]** and **[TextOperation2]** sections.

Note: There can be up to 20 entries for this keyword.

You can use text operations to create or modify:

- uniPaaS Configuration files - MagicINIFile, MGRBFile, MGREQFile
- PlainTextFile
- PlainINIFile

You should not use Magic.ini, Mgrb.ini, or Mgreq.ini as file names when you use these files in a text operation. Instead, you should use the names of the files as they appear in the **Configuration** directory.

UninstallScreenOrder

You can enter the screen identifiers in the order they should appear during the uninstall process.

For example,
`UninstallScreenOrder = 1,3,7,4`

Where the order of screen appearance is `[Screen1]`, `[Screen3]`, `[Screen7]`, and `[Screen4]`.

Note: There can be up to 50 entries for this keyword.

Source_Location

You can specify the default directory where the Application Module files should be copied from.

The following behavior applies:

- This keyword is not mandatory.
- If you don't use the **Source_Location** keyword, the files are copied from **InstallDir\ApplicationModule\Data** directory.
- You can enter a dynamic variable for the **Source_Location** keyword value. For more information about dynamic variables, see Appendix B, Dynamic Variables.

UpgradeScreenOrder

Enter the screen identifiers in the order of their appearance during the upgrade of the application module.

For example,
`UpgradeScreenOrder = 1,3,7,4`

Where the order of screen appearance is [Screen1], [Screen3], [Screen7], and [Screen4].

Notes: There can be up to 50 entries for this keyword.

Sample Configurations

```
[ApplicationModule1]
ShowAtApplicationModuleScreen=Yes
Install=Yes
Name = eMerchant Application Server Components
Description = Install the eMerchant Application files and Database
GetDirectoryFromUser=Yes
Destination_Location = <MAGICDIR>\eMerchant\App
CreateShortcuts = 1,2
CreateWebAliases = CopyFilesDuringUpgrade = No
InstallScreenOrder = 1,2,3,4,5,6,7,8
BeforeInstallExecuteCommands = 11,12
AfterInstallExecuteCommands = 1
BeforeUpgradeExecuteCommands = 9,10
AfterUpgradeExecuteCommands = 5,6,7
BeforeRemoveExecuteCommands = 4
AfterRemoveExecuteCommands = 2,3
SetEnvironmentVariables = 1,2,3
```

[TextOperations#]

You can design up to 50 different Text operations and assign them to each of the defined Application Modules, in the Text Operations setting, described on page 51.

uniPaaS configuration files are created according to the selected uniPaaS product by default. When you use the **TextOperations** keyword, the installation overwrites these default files depending on your application's requirements.

The uniPaaS configuration files include:

- Magic.ini

- Mgrb.ini
- Mgreq.ini

This setting lets you access a user-defined configuration based on uniPaaS selections, dynamic variables mechanism fields, and user selections.

Note: You can enter a dynamic variable for this property. For more information, see Appendix B, Dynamic Variables.

The following is a representation of the different properties of the **TextOperation** keyword:

Condition =
Type= MagicINIFile / MGRBFile / MGREQFile / PlainTextFile / PlainINIFile / PlainTextDir
Operation = ParseFile / **MergeLines**
SRCParseFile=
PlainINIFile =
SRCMergeFile=
TargetFile=
PerformAtUpgradeMode = **Yes**/No

Two **TextOperation** keyword examples are displayed below.

[TextOperations1]

```
Type=MagicINIFile

SRCMergeFile=<CONFIG_DIR>\Magic.src

TagetFile=<MAGICDIR>\Magic.ini
```

[TextOperation3]

```
Type=PlainTextDir

Operation=ParseFile

SRCParseFile=<CONFIG_DIR>\ora*.sql

TargetFile=<BUILD_DIR.\DBscripts\oracle
```

Condition

The Text operation is performed depending on the value of this property.

Note: You can enter a dynamic variable for this property. For more information, see Appendix B, Dynamic Variables.

See also the section about using Expression Evaluators on page 84.

Type

The following options are:

- MagicINIFile - Creates the Magic.ini file
- MGRBFile
- MGREQFile
- PlainTextFile
- PlainINIFile
- PlainTextDir

Note: This keyword is **mandatory.**

Operation

The following options for the **Operation** keyword are described below.

ParseFile

This is only relevant for **PlainTextFile** or **PlainINIFile** types. The installer enters the parse file information in the **SRCParseFile** property.

The setup program takes the source file, **SRCParseFile**, and parses its contents. This means that you can include dynamic variables in your text file, and the setup program converts them into values. The result is copied to the **TargetFile**. For more information, see Appendix B, Dynamic Variables.

For example:

```
[MAGIC_LOGICAL_NAMES]
Logmain = <USERMODULE1_DIR>
SUPPORTMAIL = <scr.3.value.SelectionRadio>
```

MergeLines

This is relevant for all Text Operation types except **PlainTextFile**.

The setup program takes the source file and merges its contents to either the created uniPaaS configuration or plain.ini file. Afterwards, the result is copied to the destination file.

For example:

```
[MAGIC_ENV]Owner = Magic Software Enterprises Ltd
[MAGIC_SYSTEMS]StartApplication = <USERMODULE1_DIR>\
example.ecf
[MAGIC_LOGICAL_NAMES]Logmain = <USERMODULE1_DIR>
```

Note: The default value is **MergeLines**.

Limitations

The '+' character, is not supported. You should insert all the information in one line, even if this results in a line width of 400 or 500 characters.

SRCParseFile

For this property, you can enter the name and location of the file to be parsed. You can also use dynamic variables. For more information, see Appendix B, Dynamic Variables.

This property is used when **Operation** = ParseFile.

Note: This keyword is **mandatory**.

PlainINIFile

For this property, you can enter the name and location of the file that is the source for the merged file. You should only use this property for **PlainINIFile**.

Note: This keyword is **mandatory**.

You can use this keyword if you want to perform an operation in the Magic.ini file, and you want to use your own source and not the uniPaaS source.

SRCMergeFile

For this property, you can enter the name and location of the file containing the lines to be merged into the **PlainINIFile** or any other uniPaaS configuration file.

Note: You can enter a dynamic variable for this property. For more information, see Appendix B, Dynamic Variables.

This property is only relevant for INI files and therefore is not relevant for the **PlainTextFile** option.

TargetFile

For this property, you can enter the name and location of the created file after the Text operation. This option is only relevant for the following types: MagicINIFile, PlainTextFile, PlainINIFile

Note: You can enter a dynamic variable for this property. For more information, see Appendix B, Dynamic Variables.

PerformAtUpgradeMode

For this property, you can specify whether to repeat the operation in Upgrade mode or ignore the operation and leave the file as is.

Note: The default value is **Yes**.

Message

When you use this keyword, your message is displayed in a box while the Text operation is being executed.

You can use the **:\n** directive to represent a new line in the text.

Note: This keyword is useful for processes that do not display output information on the screen, such as a batch process and processes that take more than a few seconds.

[Screen#]

The **[Screen#]** section lets you define specific screens for entering information that your application's installation program requires. You can assign up to **50** different information screens. Each screen is assigned to a specific Application Module.

These screens prompt the end user to enter data necessary to the installation of the application and can be used to ask for the locations of the target directory to store application files, database information, email system information, installation options, and so on. You can specify the screens used by a specific Application Module by entering the screen identifier for the **InstallScreenOrder** setting under **[ApplicationModule#]**, as explained on page 24.

For example:
`ScreenOrder = 1,2`

where 1 is `[Screen1]` and 2 is `[Screen2]`

ScreenType

You can select screens from the following screen types:

- SelectionList - Displays a list box of options
- SelectionRadio - Displays a screen with up to seven radio buttons
- SelectionRadioPlus - Displays a screen with up to three radio buttons with a description of each option
- YesNo - Displays a message box with Yes and No buttons
- SelectionCheckBoxes - Displays a screen with check boxes
- SelectionCheckBoxesPlus - Displays a screen with up to four check boxes and their help files
- EditBox1 - Displays a screen with a single edit field for the end-user to enter data
- EditBox2 -Displays a screen with two edit fields for the end-user to enter data
- EditBox3 -Displays a screen with three edit fields for the end-user to enter data
- GetDir - Displays a screen that lets the end-user browse to a directory
- Password - Displays a screen with user name and password edit boxes
- PasswordPlus - Displays a screen with the user name, password, and host name edit boxes
- Folder - Displays a screen that lets the end-user define a folder for the shortcuts in the Start menu
- ExternalExecutable - Runs from an external program

Note: The ScreenType keyword is **mandatory**.

The following sections descruibe the values you must enter for each screen type.

SelectionList

- Title - The selection list title (**mandatory**).
- Description - The screen description (**mandatory**). You can use the **:\n** directive representing a new line in the text.
- SEL_Choices - The screen list box choices (**mandatory**).
- SEL_DefaultValue - The default list box option (**mandatory**).

YesNo

- Description - Screen description (**mandatory**). You can use the **:\n** directive representing a new line in the text.
- YesNo_DefaultValue - Enter **Yes** to use the default value.
  Note: The default value is **No**.

SelectionCheckBoxes and SelectionRadio

- Title - Screen title (**mandatory**).
- Description - Screen description (**mandatory**). You can use the **:\n** directive representing a new line in the text.
- TotalSelections - Total number of check box options or radio buttons (**mandatory**).
- Sel1Title to Sel7Title - Specify the check box or radio button title. You can specify up to seven check box options or radio buttons (**mandatory**).
- Sel1Value to Sel7Value - Enter **Yes**, for the check box to appear as marked. **No** means that the check box appears as not marked. Relevant only for check box selections. **No** is the default value.

  If the condition value is equal to logical True, the check box value will be False no matter what value is assigned in the **Value** keyword. The radio button is not be selected. Also, when trying to retrieve information regarding a radio or check box button, the value returned is False or Null.

If all the control conditions are set to logical True in the radio or check box dialog, the application installation does not display the specific dialog box.

The AIU does not let you move to another dialog box when clicking on the Next button in a radio or check box dialog that is not enabled.

- Sel1DisableConditon to Sel7DisableCondition = If the condition evaluates to True or the user enters Yes, the specified radio or check box is disabled during the dialog input process. You can specify the use, install, or get options that should be disabled because they are already installed on the user's computer.

SelectionCheckBoxesPlus and SelectionRadioPlus

- Title - Screen title (**mandatory**).
- Description - Screen description (**mandatory**). You can use the **:\n** directive representing a new line in the text.
- TotalSelections - Total number of check box options or radio buttons (**mandatory**). You can select up to four check boxes or radio buttons.
- Sel1Title to Sel4Title - Specify the check box or radio button title. You can specify up to four check boxes or radio buttons (**mandatory**).
- Sel3Value to Sel4Value - Enter **Yes** for the check box to appear as selected. Enter **No** for the check box to appear as not selected. Relevant only for check box selections. **No** is the default value.
- Sel1Description to Sel3Description - Specify the help description for the check box or radio button.

  You can define a condition that disables that display of check or radio boxes. The keyword is: **Sel1DisableCondition...Sel7DisableCondition = Condition**

  If the condition value is equal to logical True, the check box or radio button display is disabled. When retrieving information about the specific check box or radio button, make certain that the condition value is equal to logical True because then the return value always is False or Null.

  If all the control conditions are set to logical True in the radio or check box

dialog, the AUI does not display the specific dialog box.

The Application Installation Utility does not let you move to another dialog box when clicking on the Next button in a radio button dialog when none of the buttons are selected.

- Sel1DisableConditon to Sel7DisableCondition = If the condition evaluates to True or the user enters Yes, the specified radio or check box is disabled during the dialog input process. You can specify the use, install, or get options that should be disabled because they are already installed on the user's computer.

EditBox1

- Title - Screen title (**mandatory**).
- Description - A description of the screen (**mandatory**).
- Variable1_Title - Edit box label text (**mandatory**).
- DefaultValue1 - The default value if no value is entered (**mandatory**). You can also enter a dynamic variable. For more information about dynamic variables see Appendix B, Dynamic Variables.
- Val1Mandatory - If you enter **Yes** for the value, the end-user can only proceed if information is entered in this field. If you enter **No** for the value, the end-user is able to proceed even without entering information in this field. **Note:** The default value is **No**.

EditBox2

- Title - Screen title (**mandatory**).
- Description - Screen description (**mandatory**). You can use the **:\n** directive representing a new line in the text.
- Variable1_Title - Edit box label text for the first edit box (**mandatory**).
- DefaultValue1 - The default value for the first edit box.
  You can enter a dynamic variable. For more information, see Appendix B, Dynamic Variables.
- Variable2_Title - Edit box label text for the second edit box (**mandatory**).

- DefaultValue2 - The default value for the second edit box. You can enter a dynamic variable.
- ValXMandatory - If you enter **Yes** for this value, the end-user can only proceed if information is entered in this field. If you enter **No**, the end-user is able to proceed even without entering information.

Note: The default value is **No**. The **#** can either be for Field 1 or 2.

EditBox3

- Title - Screen title (**mandatory**).
- Description - The Screen description (**mandatory**). You can use the **:\n** directive representing a new line in the text.
- Variable1_Title - The edit box label text for the first edit box (**mandatory**).
- DefaultValue1 - The default value for the first edit box.
  You can enter a dynamic variable. For more information, see Appendix B, Dynamic Variables.
- Variable2_Title - The edit box label text for the second edit box (**mandatory**).
- DefaultValue2 - The default value for the second edit box. You can enter a dynamic variable.
- Variable3_Title - The edit box label text for the second edit box (**mandatory**).
- DefaultValue3 - The default value for the third edit box. You can enter a dynamic variable.
- Val**X**Mandatory - If you enter **Yes** for this value, the end-user can only proceed if information is entered this field. If you enter **No,** the end-user is able to proceed without entering information.

Note: The default value is **No**. The **X** can either be for Field 1, 2, or 3.

GetDir

- Title - The screen title (**mandatory**).
- Description - The screen description (**mandatory**). You can use the **:\n** directive representing a new line in the text.
- DefaultValue1 - The default value if no value is entered (**mandatory**). You can enter a dynamic variable. For more information, see Appendix B, Dynamic Variables.

Password

- Title - The screen title (**mandatory**).
- Description - The screen description (**mandatory**). You can use the **:\n** directive representing a new line in the text.
- Username - The default value.
- Val1Mandatory - If you enter **Yes** for the value, the end-user can only proceed if information is entered. If you enter **No,** the end-user is able to proceed without entering information. **Note:** The default value is **No**.
- Password - The default value.
- Val2Mandatory - If you enter **Yes** for the value, the end-user can only proceed if information is entered in the field. If you enter **No** for this value, the end-user can proceed even without entering information.
  Note: The default value is **No**.

Note: For security reasons the AIU does not save information entered by the user. Therefore, this information is not restored in Maintenance Mode, unlike the an uninstall or upgrade process.
If you require this information you must ask the end-user to reenter this information.

PasswordPlus

- Title - The screen title (**mandatory**)
- Description - The screen description (**mandatory**). You can use the **:\n** directive to represent a new line in the text.
- Username - The default value.
- Val1Mandatory - If you enter Yes, the end-user can proceed only if information is entered in the field. If you enter No, the end-user can proceed without entering information in the field. The default value is **No**.
- Password - The default value.
- Val2Mandatory - If you enter Yes, the end-user can proceed only if information is entered in the field. If you enter No, the end-user can proceed without entering information in the field. The default value is **No**.
- Server - Specify the default value for the server name. You can enter a dynamic variable. For more information about dynamic variables, see Appendix B, Dynamic Variables.
- Server_Description - Specify the title of the Server edit box. The default value is **Host**.
- SEL_Choices = <Oracle> or <MSSQL> - This keyword is used to display a selection button near the Server edit box, which displays the following selection lists:
    - For MSSQL, the list of available MSSQL servers
    - For Oracle, the list of connect strings

  You can choose only one option, Oracle or MSSQL. If you specify either Oracle or MSSQL but the respective client is not installed on your computer, when pressing the selection button, an error message is displayed letting you know that the MSSQL or Oracle Server list could not be retrieved.

Folder

- Title (**mandatory**) - The screen title
- Description (**mandatory**) - You can use the :\n directive representing a new line in the text.
- DefaultValue1 (**mandatory**) - The default value when no value is entered. You can enter a dynamic variable. For more information, see Appendix B, Dynamic Variables.

ExternalExecutable

- ExecutableLocationAndName (**mandatory**) - Specify the location and name of the external program. This property can also be a dynamic variable. For more information, see Appendix B, Dynamic Variables.
- CommandLineParameters (**mandatory**) - Specify parameters that can be called by other programs . You can also enter a dynamic variable.

ExecuteCheckCommands

Specify a numeric identifier for a command defined in the **[Commands#]** section, as explained on page 73. For example, you can specify ExecuteCheckCommands=1,2,5 where 1, 2, and 5 are commands listed in the **[Commands#]** section.

- This command is executed when the end user clicks the **Next** button.
- Use this option to execute a command that checks the data entered in the previous screen, such as a password.

CheckErrorCondition

If the condition specified for this keyword evaluates to **True**, the installation program displays a message box, and waits for instructions from the installer.

- You can use this keyword to check values entered by the end-user. For example: if the end-user installs an Oracle version, you should use this option to check in the registry whether an Oracle database/client is installed on the machine.
- If the condition evaluates to **True**, the message box offers the following options: **Ignore**, **Abort** or **Retry**.

See also the section about using Expression Evaluators on page 84.

CheckErrorMessage

Specify a message to be displayed when the condition specified in the **CheckErrorCondition** keyword evaluates to **True**.

Note: You can enter a dynamic variable for this property. For more information, see Appendix B, Dynamic Variables.

Sample Configuration

```
[Screen1]
ScreenType = SelectionRadio
Title = eMail setting
Description = Please select your eMail program
TotalSelections = 4
Sel1Title = SMTP
Sel2Title = MAPI
Sel3Title = NOTES
Sel4Title = NONE
Sel1Value = Yes

[Screen2]
ScreenType = EditBox2
Condition = <scr.1.value.Radio> == SMTP
Title = eMail account details
Description = Please enter your eMail details
Variable1_Title = Mail Server address
DefaultValue1 = 12
Variable2_Title = Sender
```

```
[Screen3]
Condition = <scr.1.value.Radio> == MAPI
ScreenType = EditBox1
Title =Mail account details
Description = Please enter the mail account name
Variable1_Title = Account

[Screen4]
Condition = <scr.1.value.Radio> == NOTES
ScreenType = GetDir
Title = Lotus Notes Location
Description = Please select the Lotus Notes installation directory
Variable1_Title = Folder
DefaultValue1 = D:\Lotus\Notes

[Screen5]
Condition = <scr.5.value.YesNo> == Yes
ScreenType = Password
Title = Database Server administrator
Description = Please enter the Database Server administrator User-
name and Password
Username=sa

[Screen6]
Condition = <scr.5.value.Check.1> == Yes
Condition = <scr.5.value.YesNo> == Yes
ScreenType = EditBox1
Title = Database Server location
Description = Please enter the Database address or alias.
Variable1_Title = Server
Val1Mandatory = Yes
DefaultValue1 = <uservar,type=env,key=PATH>

[Screen7]
Condition = <APPL_MODULE1_IS_INSTALLED> == No
ScreenType = EditBox2
Title = Connection to Application Server
Description = Please enter the details of the Magic Broker
Variable1_Title = IP Address
DefaultValue1 = <COMPUTER_NAME>
Variable2_Title = Port
DefaultValue2 = 3001
```

[WebAlias#]

The AIU lets you design up to fifty Web aliases that can be assigned to an Application Module by entering the numeric identifier defined for the **CreateWebAlias** keyword in the **[ApplicationModule#]** section, as explained on page 48. A Web alias is a logical name assigned to a directory path name located on the Web server. To create Web aliases, the IIS Web server must be installed on the computer that generates the installation program.

Condition

You can specify whether or not to create the Web alias using a condition value. This keyword is optional.

See also the section about using Expression Evaluators on page 84.

AliasName

You can specify a Web Alias name.This keyword is **mandatory**.

Note: Dynamic variables are available for this property. For more information, see Appendix B, Dynamic Variables.

Directory

You can specify the location of the directory where the Web alias is stored. This keyword is **mandatory**.

Note: Dynamic variables are available for this property. For more information, see Appendix B, Dynamic Variables.

Permission

This keyword lets you specify the following permission options for the Web alias:

- READ
- WRITE
- SCRIPT
- EXECUTE
- INDEX
- BROWSE

You can also combine more than one option using the | character.
For example: READ | WRITE | SCRIPT.

Note: The default value is **READ**.

CreateApplication

You can enter **Yes** to create an application for the Web alias. The application name is identical to the Web alias name.

Note: The default value is **No**.

Sample Configuration

```
[WebAlias1]
Condition = <scr9.value.YesNo> == Yes
AliasName = eMerchant1
Directory = <APPL_MODULE2_DIR>
Permission = READ | SCRIPT
```

[Shortcut#]

The AIU lets you define up to fifty different shortcuts for the installation that can be assigned to your Application Modules by entering the Shortcut identifier in the **CreateShortcuts** setting of the **[ApplicationModule#]** section, as described on page 48.

Name

Specify a shortcut name. This keyword is **mandatory**.

Note: You can enter a dynamic variable for this property. For more information, see Appendix B, Dynamic Variables.

ShortcutType

You can specify whether you want the Shortcut type to be an ICON or a URL. The default value is **ICON**.

Command

You can specify the ICON or URL command. This keyword is **mandatory**.

Note: You can enter a dynamic variable for this property. For more information, see Appendix B, Dynamic Variables.

TargetDirectory

For icon shortcuts, specify the directory path where the shortcut command is executed. This keyword is **mandatory**.

Note: You can enter a dynamic variable for this property. For more information, see Appendix B, Dynamic Variables.

IconsFile

You can specify the path and file name of the icon. If the file is a .dll or .exe file, the icon number will be set to **0**.

Note: You can enter a dynamic variable for this property. For more information, see Appendix B, Dynamic Variables.

RunMode

The options for this keyword are:

- 1 = Normal
- 3 = Max
- 7 = Min

IconNumber

This number is used as an identifier for an icon in an .exe or .dll file.

Condition

You can enter a condition for creating a shortcut according to the condition value. See also the section about using Expression Evaluators on page 84.

CreateLocation

You can specify the directory path where the shortcut is created.

It is preferable to use dynamic virtual fields. The following dynamic virtual fields available for this keyword are listed below:

```
<FOLDER_DESKTOP>
<FOLDER_PROGRAMS>
```

<FOLDER_STARTMENU>
<FOLDER_STARTUP>

This keyword is **mandatory**.

Sample Configuration

```
[Shortcut1]
Name = eMerchant Documentation
Command = <MAGIC_RUNTIME> /INI=<APPL_MODULE1_DIR>\Magic.ini
ShortcutType = ICON
TargetDirectory = <MAGICDIR>
CreateLocation = <FOLDER_DESKTOP>
```

[Command#]

The AIU lets you define up to fifty commands and assign them to your Application Modules by entering the command identifier number in the Command settings of the **[Application_Module#]** section, described on page 49. The Command settings let you specify the program, path, and parameters to execute the relevant command.

Condition

If the value of the condition is **True**, the command will be executed when the condition exists.

See also the section about using Expression Evaluators on page 84.

OperationType

Valid values are Internal and External. The default value is External.

If you select Internal, specify in the Program keyword the Internal command as described in Appendix C, Internal Commands. You can also use the Message keyword. All other keywords are useless in this mode.

Program

Specify a program name or internal command according to the OperationType value. If you select Internal, you can enter a dynamic variable for this property.

Path

Specify the program path. This keyword is **mandatory**.

Note: You can enter a dynamic variable for this property. For more information, see Appendix B, Dynamic Variables.

Parameters

You can define the Command Line parameters to be passed to the called program.

Note: You can enter a dynamic variable for this property. For more information about dynamic variables see Appendix B, Dynamic Variables.

Wait_Parameter

Enter **Yes** to make the setup program wait for the command to finish.

Note: The default value is **Yes**.

Message

The message is displayed in a box while the command is being executed. You can use the **:\n** directive representing a new line in the text.

Note: This is useful for processes that do not output information to the screen, such as a batch process.

SilentMode

Select **Yes** to hide the program's running window, or select **No** to display the program window. Select **Yes** when you are executing batch files or executables without a window. For all other files, select **No**.

When **Yes** is selected, the AIU opens and monitors the process. If **No** is selected, the AIU has Microsoft Windows execute the process. Make sure that you are using the correct value as determined by your program. The default value is **Yes**.

CheckErrorCondition

If the condition specified for this keyword evaluates to **True**, the installation program displays a message box, and waits for instructions from the installer.

- You use this option to verify the result of the last execution, for example, to check the log file of a SQL command.
- If the condition evaluates to **True**, the message box offers the following options: **Ignore**, **Abort** or **Retry**.

See also the section about using Expression Evaluators on page 84.

CheckErrorMessage

Specify a message to be displayed when the condition specified in the **CheckErrorCondition** keyword evaluates to **True**.

Note: You can enter a dynamic variable for this property. For more information, see Appendix B, Dynamic Variables.

CheckOKCondition

If the condition specified for this keyword evaluates to **True**, the installation program displays a message box, and waits for instructions from the installer.

- You can use this option to check the result of the last execution, for example, to check the log file of a SQL command.
- Use this keyword when you want to give the end-user feedback about the success of an operation.

See also the section about using Expression Evaluators on page 84.

CheckOKMessage

Specify a message to be displayed when the condition specified in the **CheckOKCondition** keyword evaluates to **True**.

Note: You can enter a dynamic variable for this property. For more information, see Appendix B, Dynamic Variables.

Sample Configurations

```
[Command1]
Condition = <scr.5.value.Check.1> == Yes
Program=eMerchant_db.bat
Path=<SRCDIR>\db_script
Parameters= <scr.7.value.EditBox1> <scr.6.value.Username>
Wait_Parameter=Yes
```

```
Message = Please wait while SETUP creates the eMerchant Database...

[Command2]
Condition = <scr.2.value.Radio> == SMTP
Program=blat.exe
Path=<APPL_MODULE1_DIR>\Util\Mail\Blat
Parameters= -install <scr.2.value.EditBox2> <scr.2.value.EditBox1>
Wait_Parameter=Yes

[Command3]
Condition = <scr.2.value.Radio> == NOTES
Program=notepad.exe
Path=<WINDIR>
Parameters= <scr.1.value.EditBox1>.txt

[Command4]
Condition = <scr.2.value.Radio> == NOTES
Program=notepad.exe
Path=<WINDIR>
Parameters=
```

[Environment#]

The Application Installation Utility lets you define up to fifty system environment variables for your Application Modules by entering the environment identifier defined in the SetEnvironmentVariables property of the **[ApplicationModule#]** section, as explained on page 51. Each environment variable must be defined in its own section and assigned a unique numeric identifier, such as, `[Environment1]` and `[Environment2]`.

Note: System environment variables are set in the system registry.

Condition

Specify the condition to determine if the environment variable is set.

See also the section about using Expression Evaluators on page 84.

Name

Specify the name for the environment variable. This keyword is **mandatory**.

Value

Specify the value for the environment variable. This keyword is **mandatory**.

Note: You can enter a dynamic variable for this property. For more information, see Appendix B, Dynamic Variables.

Sample Configurations

```
[Environment1]
Condition = <scr.1.value.Radio> == NOTES
Name = Host1
Value = <APPL_MODULE1_DIR>

[Environment2]
Condition = <scr.1.value.Radio> == NOTES
Name = Host2
Value = <APPL_MODULE1_DIR>\test
```

[Finish_Install#]

This section lets you design the **Finish Install** screen, displayed at the end of the first installation. When files are locked during the copying process, a default screen appears, prompting the end-user to restart the computer.

Depending on your installation configuration, you may also need to define **Finish Uninstall** and **Finish Upgrade** screens.

Message

Enter the message text to be displayed on the screen. You can use the **:\n** directive representing a new line in the text.

If no text is entered, the following default message is displayed:

```
The SETUP program has successfully installed the [Product-
Name] application on your computer.
```

ShowREADME

Enter **Yes** for the Setup program to prompt the end-user to view the readme file by displaying the readme check box. The default value is **No**.

FileName

Specify the file path of the text file used for the Readme. This property is only relevant when the setting in the ShowREADME keyword is **Yes**.

Note: You can enter a dynamic variable for this property. For more information, see Appendix B, Dynamic Variables.

PerformActionAtEnd

Enter **Yes** to display a check box with an action name defined in the **EndActionName** property. The default value is **No**.

EndActionName

Enter the action name text displayed in the **Finish Installation** screen.

EndActionCommandNum

You can specify a numeric identifier for a command specified in the **[Commands#]** section, as explained on page 73. The commands for this keyword is executed when the end-user checks the **PerformActionAtEnd** check box and clicks the **Finish** button.

Title

You can enter a string that is displayed as the title of a dialog box.The entered text string overwrites the setup default title text.

ForceReboot

If the value of the condition of this property is set to **True** or the expression value evaluates to **True**, the other properties of the **Finish Installation** screen are ignored, forcing a reboot.

READMETitle=xxxxx

Lets you change the readme title.

Sample Configuration

```
[Finish_Install]
Message = finish installation
ShowREADME = No
PerformActionAtEnd = Yes
EndActionName = View the eMerchant Documentation
EndActionCommandNum = <APPL_MODULE1_DIR>\Documentation\eMerchant.htm
```

[Finish_Upgrade#]

This section lets you design the **Finish Upgrade** screen, displayed at the end of an upgrade process.

Note: When files are locked during the copying process, a default screen appears, prompting the end-user to restart the computer.

Message

Enter the message text that is displayed on the screen. You can use the **:\n** directive representing a new line in the text.

If no text is entered, the following default message is displayed:

```
The SETUP program has successfully upgraded the [Product-
Name] application on your computer.
```

ShowREADME

Enter **Yes** for the Setup program to prompt the end-user to view a Readme file by displaying the Readme check box. The default value is **No**.

FileName

Specify the file path of the text file used for the Readme. This property is only relevant when the setting in the **ShowREADME** keyword is **Yes**.

Note: You can enter a dynamic variable for this property. For more information, see Appendix B, Dynamic Variables.

PerformActionAtEnd

Enter **Yes** to display a check box with an action name defined in the EndActionName property.

Note: The default value is **No**.

EndActionName

Enter the action name text that is displayed in the **Upgrade Installation** screen.

EndActionCommandNum

You can specify a numeric identifier for a command specified in the **[Commands]** section, as explained on page 73. The commands for this keyword is executed when the end-user checks the **PerformActionAtEnd** check box and clicks the **Finish** button.

ForceReboot

If the value of the condition of this property is set to **or** evaluates to **True**, the other properties of the **Finish Installation** screen are ignored, forcing a reboot.

Sample Configuration

```
[Finish_Upgrade]
Message = Finish upgrade
ShowREADME = No
PerformActionAtEnd = Yes
EndActionName = Test
EndActionCommandNum = NOTEPAD.EXE c:\Autoexec.bat
```

[Finish_Uninstall#]

This section lets you design the **Finish Uninstall** screen, displayed at the end of the uninstall process. When files are locked during the process of copying them to the target directory, a default screen appears, prompting the end-user to restart the computer.

Message

Enter the message text that is displayed on the screen. You can use the **:\n** directive representing a new line in the text.

If no text is entered, the following default message is displayed:

```
The SETUP program has successfully removed the [Product-
Name] application from your computer.
```

ShowREADME

Enter **Yes** for the Setup program to prompt the end-user to view the Readme file by displaying the **Readme** check box. The default value is **No**.

FileName

Specify the file path of the text file used for the Readme. This property is only relevant when the setting in the **ShowREADME** keyword is **Yes**.

Note: The file should not run from the CD but from the local machine.

PerfromActionAtEnd

Enter **Yes** to display a check box with an action name defined in the **EndActionName** property. The default value is **No**.

EndActionName

Enter the action name text that is displayed in the **Upgrade Installation** screen.

EndActionCommandNum

You can specify a numeric identifier for a command specified in the **[Commands]** section, as explained on page 73. The commands for this keyword is executed when the end-user checks the **PerformActionAtEnd** check box and clicks the **Finish** button.

Expression Evaluator

The Application Installation Utility lets you insert conditions in several places in the configuration file. These places are marked with the **Condition** keyword.

Note: You can use **Yes** or **No** for the **Condition** keyword.

The following logical tokens can also be used:

```
==, >=, <=, >, <, !=, %
```

For example:

```
<APPL_MODULE1_IS_INSTALLED> == Yes
```

Note: The **%** symbol is used for searching in strings. For example: `X%Y` will return **True** if **Y** is part of the **X** string. In the example if X=1234 and Y=34, a **True** value is returned.

You can use the <NULL> directive to specify empty values.

For example:

```
<scr.7.value.EditBox1> != <NULL>
```

You can also use the following Boolean tokens to create real expressions:

`&&` - which represents a logical **AND**.

`||` - which represents a logical **OR**.

For example:

```
<APPL_MODULE2_IS_INSTALLED> == Yes ||
<APPL_MODULE1_IS_INSTALLED> == No && <scr.11.value.YesNo> == Yes
```

Note: && has a higher priority than **||**, when an expression is being evaluated.

In this example:

```
(1 || 2 & 3)
```

the Expression Evaluator first makes a logical **and** between 2 and 3. The result is combined with 1 using the logical **or**.

User-Defined Variables

This dynamic variable lets you call a user-defined variable. Each variable requires a name and the value assigned to the variable.

The format is [Variable Name] = [Variable Value].

A few examples are:

[User_Variables]
AccountName=Lawrence
Temp=<TEMPDIR>

There is no limit to the number of user variables that can be defined.

[Conversions]

This section lets you specify a table with a value and convert values. The string value is converted into one of its defined convert value strings by using the Fetch Information Conversion dynamic variable, described on page 100.

Data Conversion tables are useful when you creating an installation program that has different language options.

The table definition is:
Value = Convert_Value1, Convert_Value2, Convert_Value3, Convert_Value4

For example,
Language = French, German, Spanish, Dutch

[Conditions]

This section is meant to store conditions.

You can:

- Enter all conditions into a single location

- Define complex expressions by using several conditions together or by using nested conditions, for example, Pop3=<scr.1.value.Radio>==MAPI
- Make the condition more readable by displaying the condition name instead of the condition value
- Minimize routine maintenance

The condition name and value are defined in the Fetching Condition Name and Fetching Condition Value in Appendix B, Dynamic Variables.

Installation Files 3

This chapter describes the files contained in the CD's Application Files Directory that are necessary for the installation process. The installation process includes two Setup executable files. One file handles a multi-installation option that determines whether to launch either a new installation or an upgrade process.

In this chapter:

To implement the Application installation Utility, you should execute the file **Install.exe**, located in the **ApplicationInstallationUtility** directory.

uniPaaS Installation

The uniPaaS installation files are:

- uniPaaS Installation Data files
    - Layout.bin
    - Data1.hdr
    - Data1.cab
    - Data2.cab
    - Install.exe
- Installshield Kernel Files
    - Engine32.cab
    - Ikernel.ex_
    - Setup.exe
    - Setup.ini
    - Setup.boot
    - Setup.skin
- Setup Compiled Script
    - Setup.inx

SetupConfiguration.ini File

This file contains all the directives for the Application Installation Utility.

ApplicationModule Directories

Each Application Module directory has the following structure:

ApplicationModuleX

The Application Module number, for example, `ApplicationModule2`.

Data

The files that are copied during the installation or upgrade procedure. The Data directory contains the hierarchy of files to be copied to the target machine, which are copied according to the settings in the **CopyDirective** variable of the **[ApplicationModule#]** section of the **SetupConfiguration**.ini file, described on page 43.

Configuration Files Directory

In the Configuration directory, you should store all the files that are used during the installation but which are not to be copied with the application modules.

For example:

- Source file for the application configuration templates such as the Magic.ini and Mgrb.ini
- Installation bitmaps
- Installation scripts

Defaults and Preferences A

When you supply uniPaaS with your application, you can provide any of the different product environments: uniPaaS Deployment Client or uniPaaS Server. Each uniPaaS environment comes with pre-selected components.

This appendix details the components supplied with each type of environment. You can find more information about uniPaaS components on page 34.

uniPaaS Deployment Client

The pre-selected uniPaaS components for the uniPaaS Deployment Client, CSRT, are:

- Deployment modules
- Gateways\DB2
- Bundled products\DB2 Express
- FLexLM License Server
- Web Services Framework
- Language\English (International)
- Help

uniPaaS Server

The pre-selected uniPaaS components for the uniPaaS Server, ENT1, are:

- Deployment modules
- Browser based deployment
- Middleware\Broker
- Middleware\J2EE Integration
- Gateways\DB2
- Bundled products\DB2 Express
- FlexLM License Server
- Web Services Framework
- Language\English (International)
- Help

Dynamic Variables

B

You can use dynamic variables in many of the entries in the SetupConfiguration.ini file. This appendix describes the available dynamic variables and their usage in the groups displayed below.

Shortcut Variables

These dynamic variables can used when specifying a shortcut location.

<FOLDER_DESKTOP>	This variable is used when you want to specify a shortcut location, in this case the **Desktop**.
<FOLDER_PROGRAMS>	This variable is used when you want to specify a shortcut location, in this case **Programs** under the **Start** menu.
<FOLDER_STARTMENU>	This variable is used when you want to specify a shortcut location, in this case the **Start** menu.
<FOLDER_STARTUP>	This variable is used when you want to specify a shortcut location, in this case the **Startup** menu of the **Programs** menu.
<FOLDER>	Folder name

System Directories

These dynamic variables can be used to specify the location of system directories.

<PROGRAMFILES>	The location of the operating system's program files.
<WINDIR>	The location of the Windows directory.
<WINSYSDIR>	The location of the System directory within the Windows directory.
<SCRIPTDIRECTORY>	The location of the script directory in the IIS Web server.
<WEBDOCUMENTPATHDIR>	The location of the Web document root in the IIS Web server.
<COMMONFILES>	The location of the Operating System's common files. This is usually found under **<PROGRAMFILES>\Common Files\xxx**
<TEMPDIR>	The temporary location of the operation system.
<DRIVE_LIST>	This variable provides a string of local and remote drives separated by commas (For example, C:,D:,F:), which can be very useful in the List Box screen.

Computer Information

These dynamic variables can be used to provide computer information.

<COMPUTER_NAME>	The name of the computer.
<COMMAND_PROCESSOR>	The name and location of the Operating System's Command Processor.
<WINSYSDISK>	The Windows system drive.

<PATH>	The value of the environment variable **PATH**.
<HOST _IP_ADDRESS>	Returns the IP address of the machine where the installation is running.
<DIR_EXIST,name=xxx>	Checks if the directory exists. You must use a dynamic variable for **xxx**.
<MSSQL_CLIENT_EXISTS>	Returns Yes when the MSSQL client exists. No is returned when the MSSQL client does not exist.

Installation Information

These dynamic variables can be used to provide installation information.

<INSTALLING_VERSION>	A string representing the current installing version.
<CURRENT_INSTALLED_ VERSION>	A string representing the currently installed version of the application.
<SRCDIR>	The location of the **Application Files** directory. See Chapter 3, Installation Files, which discusses Application Files.
<SRCDISK>	The **Application Files** directory.
<TARGETDISK>	The Magic directory's drive.

<APPL_MODULEx_DIR>	The location of the application module **X**, where **X** is a number between 1 and 25. This dynamic variable can be included in the **dir** keyword. This variable returns a Boolean value, Yes or No. This variable can be used in: 1. Normal Installation mode, during the initial installation. For this mode, the following message is displayed: **Is ApplicationModuleX about to be installed?** 2. Upgrade or Uninstall mode. For these modes, the following message is displayed: **Was ApplicationModuleX installed on the machine?**
<CMDLINE>	The command line used with the installation. For example: If you enter the command **Install.exe/test**, the variable **<CMDLINE>** has the value **/test**.

uniPaaS Information

These dynamic variables can be used to provide the name and location of the uniPaaS executable file.

<MAGICDIR>	The location where uniPaaS will be installed. This dynamic variable can be included in the **dir** keyword.
<MAGIC_RUNTIME>	The name and location of the uniPaaS runtime application, adding double quotations (“ ”) when needed. The keyword is useful for shortcuts.

Current Date

These dynamic variables can be used to provide current date information.

<CURRENT_YEAR>	Returns the current year as a four digit number.
<CURRENT_MONTH>	Returns the current month as a two digit number.
<CURRENT_DAY>	Returns the current day as a two digit number.
<CURRENT_DATE>	Returns a string for the current date.

uniPaaS Broker Information

These dynamic variables can be use to provide broker information.

<MAGIC_BROKER_CMDLINE>	This keyword points to the file and location of the current Mgrqcmdl.exe file, adding double quotations (“ ”) when needed. The keyword is useful for shortcuts. An example would be stopping the broker.
<MAGIC_BROKER_PWD>	A string representing the Magic Broker password.
<MAGIC_BROKER_PORT_NUM>	Returns the current broker port number.

Process Information

These dynamic variables can be used to provide information about service processes.

<PROCESS_EXISTS,name=xxx>	Checks if a process is active. The process name is the executable file name without the file extension.
<SERVICE_EXISTS,name=xxx>	Checks if the service exists in the OS service table. You have to replace **xxx** with the appropriate service name.
<SERVICE_START,name=xxx>	You can start a computer service by designating the service name.
<SERVICE_STOP,name=xxx>	You can stop a computer service by designating the service name.
<SERVICE_DELETE,name=xxx>	You can delete a computer service by designating the service name.
<PROCESS_KILL,name=xxx>	You can stop a process by designating the process name.

For example:

```
[Command1]
Condition = <PROCESS_EXISTS, name=kaka>
OperationType = Internal
Program = <PROCESS_KILL, name=kaka>
```

Existing Directory or File

These dynamic variables can be used to search for files and directories.

<FILE_EXIST,name=xxx>	You can search for the designated file name.
<DIR_EXIST,name=xxx>	You can search for the designated directory name.

Internal Dynamic Variable

<NONE> represents an empty string. It is useful for working with conditions and checking the contents of another dynamic variable.

Fetching Information from an INI File

You can use the <USERVAR> variable, as shown below, to retrieve information from an .ini file:

<uservar,type=**ini**,file=*xxx*,section=*yyy*,key=*zzz*

The values of the USERVAR variable are:

- File – The .ini file name and location
- Section – The section in the .ini file
- Key – The .ini file key name

Fetching Information from an Environment Value

You can use the <USERVAR> variable, as shown below, to retrieve environment information:

<USERVAR, type=**env**,key=*xxx*)

The values of the USERVAR variable for an environment value are:

- Key – The key for the requested information, such as PATH

Fetching Information from a Registry Value

You can use the <USERVAR> variable, as show below, to retrieve information from a registry:

<uservar, type=**reg**,basekey=*xxx*,key=*yyy*,value=*zzz*>

The values of the USERVAR variable for a registry value are:

- Basekey – The following options are available:
    - HKEY_LOCAL_MACHINE
    - HKEY_USERS
    - HKEY_CURRENT_USER
    - HKEY_CLASSES_ROOT
- Key – The name of the registry key, such as SOFTWARE\Microsoft\IE Setup\Setup
- Value – The name of the string value, such as PATH

Fetching a User Value

You can user the <USERVAL> variable, as shown below, to retrieve information from a user-defined variable in the user variables section:

<uservar,type=**user**,name=*xxx*>

The values of the USERVAL variable for a user-defined variable are:

- Name – The name of the user-defined variable

For more information, see User-Defined Variables in Chapter 2, INI Sections

Fetching a Condition Name

You can use the <CONDITION> variable, as shown below, to define the condition name that can be used in the [Condition] section in Chapter 2, INI Sections:

<condition,name=*abc*>

Note: This keyword can be used where Condition=option for each section but cannot be used in place of regular dynamic keywords.

Fetching a Condition Value

You can user the <USERVAL> variable, as shown below, to retrieve a condition value from specified condition in the Condition section:

<uservar=condval,cond_name=xxx, value1=yyy,value2=zzz>

For example,
DefaultValue1=c:\<uservar,type=condval,cond_name=IsIboltSuite, value1=iBolt Suite is selected, value2=iBolt Suite is not selected

You can use this dynamic variable in the text operation or as a default value for other variables, such as the radio button or an external event for Windows XP or Windows 95.

To use this dynamic variable, you must specify the condition name.

Fetching Information Conversion

the Data Conversion table as described on page 85 is accessed by the following dynamic variable format:

<uservar,type=cnv, value=str,Idx=n)

The parameters are:

- Value – The name of the string to be converted.
- Idx – The index of the string in the conversion table you want to convert to. This parameter is optional and if ommitted, the default value of 1 is used.

If the string to be converted or the string to convert to are not found, an empy value is returned.

The example below shows you how this dynamic variable works with a sample table definition:

[Conversions]
Rainbow=Red,Orange,Yellow,Green,Blue,Indigo,Violet

<uservar,type=cnv,value=Rainbow,idx=4> returns the value Green.

Scanning for a String within a File

You can use the <USERVAR> variable, as shown below, to scan for a specific string in a text file:

<uservar,type=grep,file=xxx,str=yyy>

The values of the USERVAR variable are:

- File – The file name and location
- Str – The string you are searching for

Note: You can find an example of this functionity provided in the SetupConfiguration.ini example.

Fetching System Information

You can user the <USERVAL> variable, as shown below, to retrieve system information:

<uservar,type=**info**,name=*xxx*, dir=*yyy*>

The values of the <USERVAR> variable for system information are:

- Name – You can choose from the following options:
    - **<TotalMemory>** – This dynamic variable returns the total amount of memory installed on the machine. Due to operating system limitations, the value returned can be slightly different from the actual amount of physical memory installed on the system. The difference is within 100 Kilobytes of the actual value. The returned measurement value is in Kilobytes.
    - **<TotalDiskSpace>** – This dynamic variable returns the total capacity in megabytes of the disk drive specified in the in **dir** keyword. For example:
      <uservar,type=info,name=TotalDiskSpace,dir=<MGICDIR>>

- **<FreeDiskSpace>** – This dynamic variable returns the amount of free space in megabytes on the specified directory. For example: <uservar,type=info,name=FreeDiskSpace,dir=<MGICDIR>>
- **<OS>** – This dynamic variable returns the operating system and returns the following values:
  - WinXP Server
  - Win2003 Server
  - Win2000 Server
  - WinNT4
  - Win95
  - Win98
  - WinME

  For the Server edition, Server is added to the name. For the Terminal Server edition, Terminal is added to the name.

Fetching Input Data from Screens

You can use the <SCR> variable, as shown below, to retrieve input data from screens:

<scr.###.value.type>

The values of the <SCR> variable are:

– The screen number

Type – The Type options are:

1. **EditBox1 - The first string**
2. EditBox2 – The second string
3. EditBox3 – The third string
4. Username

5. Password
6. GetDir – The directory
7. ListValue – A selected string from the list
8. Radio – The string for the selected value of the Radio
9. Check.1 – Boolean – Yes/No for checkbox 1
10. Check.2 – Boolean – Yes/No for checkbox 2
11. YesNo – Boolean – Yes/No for question

Fetching Database Information

You can use the <db> variable, as shown below, to display information entered on the Database Information screen:

<**db,SQLType=Oracle,name=***YYYYYY***,value=***XXXXXX*>

The values of the <db> variable are:

- SQLTyp – You can choose Oracle, MS-SQL, MySQL, DB2, Pervasive, ODBC, or DB2/400
- Name – The database name
- Value – The value extracted. The following options are available:

  1. **UserName**
  2. Password
  3. DatabaseName
  4. DatabaseServerName
  5. Alias
  6. Location
  7. DSN
  8. DirectoryName

Internal Commands

C

You can use the OperationType keyword to call an Internal Command. The keyword's internal mode lets you insert internal commands by using the following syntax, **Program=xxxx**. You can also use the Message keyword to display a message box before an internal command is processed.

OperationType Keyword

The valid values are Internal and External. The default value is External.

Internal Commands List

<EXIT>	Performs an Exit command while displaying a message to the end-user. The message is taken from the Message keyword. The default value is “The installation is about to be aborted.”
<PROCESS_KILL,name=xxx>	Kills all active processes that have names which are equivalent to **xxx**. Note that **xxx** is a process executable name without the file extension.
<SERVICE_DELETE,name=xxx>	Deletes a service from the OS service table. You must replace **xxx** with the appropriate service name.
<SERVICE_STOP,name=xxx>	Stops a running service. You must replace **xxx** with the appropriate service name.
<SERVICE_START,name=xxx>	Starts an inactive service. You must replace **xxx** with the appropriate service name.

Saving in the Registry D

For backup purposes the installation program saves all the information written in the registry, to two files.

Saving the Files

The installation saves the information into these files:

- <MAGICDIR>\Registry\Install.reg
- <MAGICDIR>\Registry\UnInstall.reg

Sometimes when an end-user wants to upgrade, uninstall, add or remove components, a message may display stating that the information cannot be found in the registry. In such a case the end-user can reload the information contained in these two backup files by opening them in Windows Explorer.

Packaging the Installation

If you only want to create one executable file for installation purposes, you should use either WinZip or PackageForTheWeb v4.x.

You should use the Autorun.inf file when you copy the installation file into a CD. The format of the text file is:

```
[autorun]

OPEN=Install.exe

ICON=Setup.exe,0
```

Using Setup Skins

E

A Skin file is a compressed file that include graphic and ini files that define the appearance of the setup program. The Application Installation utility provides a default Skin file called Setup.skin.

You can override the default Skin file by using the predefined Skin file from the Skin directory located in the ApplicationInstallationUtility directory. Make sure to copy the selected file and rename it to Setup.skin.

Predefined skin files are listed below:

- IsBlue
- IsSlate
- IsOlive
- IsBlueTC
- IsMidnight
- IsMonochrome

Maintenance Mode

F

Using the Application Installation Utility you can enable the end-user to perform maintenance operations on your application, such as repairing the application.

Accessing the Maintenance Mode

To enter Maintenance Mode:

1. Click the **Start** button.
2. Click **Settings** and then click **Control Panel**. In **Control Panel** click the **Add/Remove Programs** icon.
3. From the list select the installed application.
4. Click the **Add/Remove** button

Maintenance Mode Options

There are three options within Maintenance Mode:

- Modify
- Repair
- Remove

Modify

Choose **Modify** to add or remove application modules.

In this mode the Application Installation Utility displays the screens for application modules that you are about to remove or instal.

Note: **Modify** mode is only available if you choose to set the: **ShowApplicationModulesScreen** keyword to **Yes**.

Repair

Choose **Repair** to repeat the previous installation.

Remove

Choose **Remove** to remove the installed application from your computer.

Note: In **Remove** mode the user is asked to confirm the uninstall request. Once the end-user confirms the uninstall a series of screens is displayed according to the definitions you set in the **SetupConfiguration.ini** file.

Upgrade Mode

G

The Install.exe file is a wrapper program that executes the Setup.exe file with the correct options, Setup.exe - ig[GUID]. [GUID] is a unique number for each installation. The installer has to decide whether the Setup.exe should use an existing [GUID] number for the Upgrade mode or start a new and unique [GUID] number.

Notes:

- Be aware that all values entered by the user on the application installation screens can be used in the Upgrade, Add and Remove, or Uninstall modes.
- The Upgrade mode must also be considered when you enter screen text and conditions for the Install mode and Password or PasswordPlus dialogs. It is advisable to prompt the user repeatedly about executing scripts.
- The install designer also has to make sure to use the **PerformAtUpgradeMode** keyword in the text operation, so the text file is not created again in Upgrade mode.
- Use the **CopyDirective** or **CopyFilesDuringUpgrade** directives for user application modules with care.

Example Screens

H

In this appendix we have provided some example screens created using the Application Installation Utility. The screens are shown together with appropriate information about their details in the SetupConfiguration.ini file.

These screens are optional and are determined in the [Application_ModuleX] section of the SetupConfiguration.ini file. The purpose of these example screens would be to ask end-users about their installation preferences, database information, and other relevant installation data.

Email Settings Screen

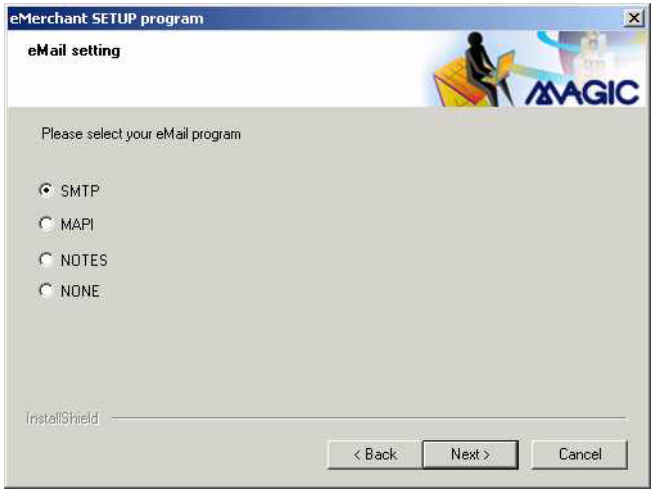

Figure H-1 Email Setting Screen

The settings for the **eMail setting** screen, shown above, are:

```
[Screen1]
ScreenType = SelectionRadio
Title = eMail Setting
Description = Please select your eMail program
TotalSelections = 4
Sel1Title = SMTP
Sel2Title = MAPI
Sel3Title = NOTES
Sel4Title = NONE
```

Mail Account Details Screen

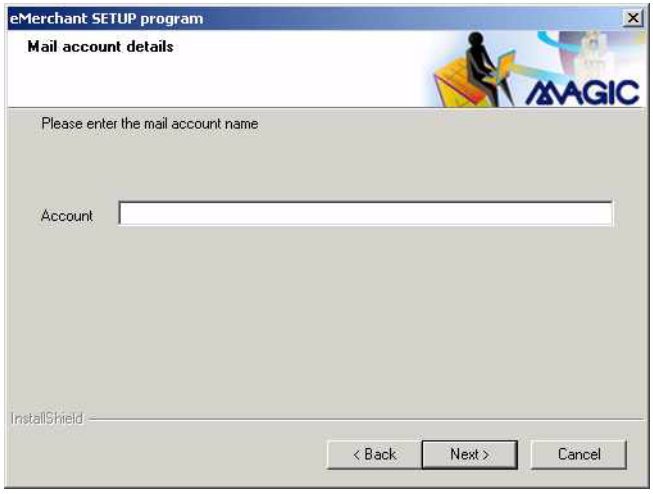

Figure H-2 Mail Account Details Screen

The settings for the **Mail account details** screen, shown above are:

```
[Screen1]
ScreenType = EditBox1
Title = Mail account details
Description = Please enter the mail account name
Variable1_Title = Account
DefaultValue1 = NONE
```

Database Server Screen

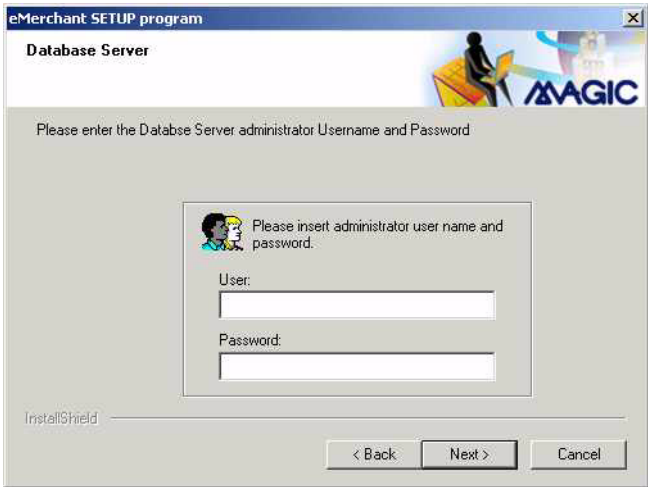

Figure H-3 Database Server

The settings for the Database Server screen, shown above are:

[Screen2]

```
ScreenType = EditBox2

Title = Database Server

Description = Please enter the Database Server administra-
tor Username and Password

Variable1_Title = User

Variable2_Title = Password
```

Application Installation Example

To run the example, please copy the **SetupConfiguration.ini** located in the Examples sub-directory to the installation root, and run the Install Application Utility.

This configuration file provides examples of the following features:

- Using a license agreement.
- Using a bitmap for the installation.
- Using keywords such as <PROGRAMFILES>, <TEMP_DIR>, and <CONFIG_DIR>.
- Complex conditions using | (the OR condition).
- Conditions based on input from screens.
- Fetching data from the registry.
- Changing the icon of the application.
- Using the **PasswordPlus** screen.
- Using the **MSSQL_CLIENT_EXISTS** keyword, which is only displayed if an MSSQL client is installed, and the selection choice of MSSQL, which displays all the registered MSSQL servers.
- Two shortcuts, one to the desktop and the other to Start Programs. Both shortcuts are displayed with an example icon, not the regular uniPaaS icon.

- A command that performs the following MSSQL statements:
    - Creates a file called **AIUexample.sql** in the temporary directory.
    - This file reads the **authors** table from the **pubs** database (Select * from pubs) and outputs the data into a file in the temporary directory called **AIUOutput.sql**.
    - If there is a problem, the command issues an error.
- An example of a readme file.

The provided example contains three modules and a Magic.ini source example. The directories are empty with a single file called **empty.txt**. This file is necessary because the directory cannot be empty.

Sample SetupConfiguration.ini File

[Application_Information]

This section provides information about the application, such as its name and version. Some of the information appears in the Add and Remove programs.

CompanyName = Magic Software Enterprises

ProductName = Application Installation Example

Version = 1.0.0.3

ApplicationLicenseAgreementFile = <CONFIG_DIR>\DemoLicense.txt

BitmapFileInstallationIcon = <CONFIG_DIR>\AIU_Example_BMP.bmp

CreateMagicShortcuts=No

GetMagicDirectoryScreen = No

HelpLink = http://www.magicsoftware.com

WelcomeMessage = The AIU example installation wizard will install the example or your machine.\nPlease note that this is meant as an example and not intended for proper use\n\nTo continue click Next.

WelcomeTitle = Welcome to the example AIU installation wizard

ReviewMessage = The Installation Wizard has enough information to begin the installation process. You can review the information and click Back to make any changes. When you are ready, click Install to begin the installation process.

ReviewTitle = Setup Information

[ApplicationModule1]

This is for the first module. The first module in this example contains general uniPaaS and application files.

ShowAtApplicationModuleScreen = Yes

Install = Yes

Name = uniPaaS AIU Example

Description = Install the uniPaaS AIU Example

GetDirectoryFromUser = Yes

Destination_Location = <PROGRAMFILES>\Magic\AIU

CreateShortcuts = 1,2

CopyFilesDuringUpgrade = No

ScreenOrder = 4,3,1,2,9

TextOperations = 1

AfterInstallExecuteCommands = 1

[ApplicationModule2]

This is the second module. This module contains sound files. This will only be installed if the user chooses the Yes option from the question about the Sound files.

ShowAtApplicationModuleScreen = No

Condition = <scr.1.value.Radio>==Yes

Install = Yes

Name = Sound Files

Description = Install the AIU example sound files

GetDirectoryFromUser = No

Destination_Location = <scr.2.value.GetDir>

[ApplicationModule3]

This is the third module.
This module is for configuring e-mails.

ShowAtApplicationModuleScreen = Yes

Install = Yes

Name = Use uniPaaS as an E-mail client

Description = Enable the utility to configure\nthe magic.ini for email.

GetDirectoryFromUser = No

Destination_Location = <MAGICDIR>

AfterInstallExecuteCommands =

ScreenOrder = 5,6,7,8

[Magic_Information]

This is the information about the product that is going to be installed. Here, uniPaaS Client Server Deployment will be installed.

MagicProductToInstall = CSRT

ChangeDefaultAssociation = No

MagicWebAliasName = /MagicAIUDemoReq

DefaultDirectory = <APPL_MODULE1_DIR>

Magic9Demo = Yes

Web Online = No

[Magic_Components]

Gateways\Btrieve = Yes

Middleware gateways\J2EE = No

Middleware\Magic Broker = Yes

Databases\Btrieve Workgroup = No

Language\English (International) = Yes

[BrokerInformation]

InstallAsService = No

ServiceName = Magic AIU Demo

BrokerTCPIPAddress = 4511

BrokerPassword = Magic AIU Demo

[TextOperations1]

This text operation defines the Magic.ini.
It takes the default .INI file and merges the values defined in Magic.src with it.

Type = MagicINIFile

SRCMergeFile = <CONFIG_DIR>\Magic.src

TargetFile =<MAGICDIR>\Magic.ini

[Screen1]

ScreenType = SelectionRadio

Title = Sound Capabilities

Description = Does your computer have sound capability ?

TotalSelections = 2

Sel1Title = Yes

Sel2Title = No

Sel1Value = Yes

[Screen2]

This screen will only be displayed if the user selects the Yes radio button in the previous screen.

Condition = <scr.1.value.Radio> == Yes

ScreenType = GetDir

Title = Sound files Location

Description = Please select the sound files installation directory

Variable1_Title = Folder

DefaultValue1 = <APPL_MODULE1_DIR>\SoundFiles

Val1Mandatory = Yes

[Screen3]

This screen is displayed only if the user defines the proxy in Screen 4.

Condition = <scr.4.value.Check.1> == Yes

ScreenType = EditBox2

Title = HTTP Proxy details

Description = These are details needed for the new HTTP Proxy option in magic.ini

Variable1_Title = HTTP Proxy

DefaultValue1 = 127.0.0.1

Variable2_Title = Port

DefaultValue2 = 8080

Val1Mandatory = Yes

Val2Mandatory = Yes

[Screen4]

ScreenType = SelectionCheckBoxes

Title = HTTP Proxy

Description = Do you have a Proxy server ?

TotalSelections = 1

Sel1Title = Do you have a Proxy server ?

Sel1Value = Yes

[Screen5]

This screen is displayed only if Module 3 is being installed.

Condition = <APPL_MODULE3_IS_INSTALLED>==Yes

ScreenType = SelectionRadio

Title = email settings

Description = Please select your email program. \nChoose "None" to setup your email later.

TotalSelections = 4

Sel1Title = POP3

Sel2Title = IMAP

Sel3Title = NOTES

Sel4Title = None

Sel1Value = Yes

[Screen6]

This screen will only be displayed if the user selected either POP3 or IMAP from the options in screen 5.

Condition = <scr.5.value.Radio> == POP3 || Condition = <scr.5.value.Radio> == IMAP

ScreenType = EditBox2

Title = email POP3 account details

Description = Please enter the email account name

Variable1_Title = Account

DefaultValue1 =

Variable2_Title = Incoming mail server address

Val1Mandatory = Yes

Val2Mandatory = Yes

[Screen7]

Condition = <scr.5.value.Radio> == NOTES

ScreenType = GetDir

Title = Lotus Notes Location

Description = Please select the Lotus Notes installation directory

Variable1_Title = Folder

DefaultValue1 = <uservar,type=reg,basekey=HKEY_LOCAL_MACHINE,key=SOFTWARE\Lotus\Notes,value=Path>

Val1Mandatory = Yes

[Screen8]

Condition = <APPL_MODULE3_IS_INSTALLED>==Yes &&
<scr.5.value.Radio> != NOTES

ScreenType = EditBox2

Title = email SMTP details

Description = Please enter your email details. \nThe "Server Address" is your SMTP server name or IP address.

Variable1_Title = Mail Server address

DefaultValue1 =

Variable2_Title = Sender

DefaultValue2 = example@example.com

Val1Mandatory = Yes

Val2Mandatory = Yes

[Screen9]

ScreenType = PasswordPlus

Condition = <MSSQL_CLIENT_EXISTS>==Yes

Title = MSSQL Database Details

Description = Enter the details of the MSSQL server.

Username = sa

Val1Mandatory = Yes

Server_Description = Server name:

SEL_Choices = <MSSQL>

[Shortcut1]

This shortcut will be placed on the desktop and will have the icon defined in the icon file.

Name = uniPaaS AIU Example

Command = <MAGIC_RUNTIME> /LicenseName=MGDEMO

ShortcutType = ICON

IconsFile = <CONFIG_DIR>\example.ico

TargetDirectory = <MAGICDIR>

CreateLocation = <FOLDER_DESKTOP>

[Shortcut2]

This shortcut will be placed on the Start\Programs folder.It is located in a subfolder called Magic that contains another subfolder. Its icon is defined in the icon file.

Name = uniPaaS AIU Example

Command = <MAGIC_RUNTIME> /LicenseName=MGDEMO

ShortcutType = ICON

IconsFile = <CONFIG_DIR>\example.ico

TargetDirectory = <MAGICDIR>

CreateLocation = <FOLDER_PROGRAMS>\Magic\uniPaaS AIU

[Command1]

This command also has an error check at the end. To check the condition, error codes are searched for in the result file.

Note: String 911 is an error message for an incorrect database. String 208 is an error message for an invalid table.

Program = example.bat

Path = <CONFIG_DIR>

Parameters = <scr.9.value.Server> <scr.9.value.UserName> <scr.9.value.Password> <TEMPDIR>

Wait_Parameter = Yes

Message = Performing an MS_SQL statement

CheckErrorCondition = <uservar,type=grep,file=<TEMPDIR>\AIUOutput.sql,str=Msg 911> ==Yes || <uservar,type=grep,file=<TEMPDIR>\AIUOutput.sql,str=Msg 208> ==Yes || <uservar,type=grep,file=<TEMPDIR>\AIUOutput.sql,str=Login failed> ==Yes

CheckErrorMessage = There was a problem with execution of the MS_SQL command.

[Finish_Install]

When the application is installed, this screen will be the final screen. You can offer to display a readme, or to perform an action such as loading the application or even to reboot the computer.

Message = The example has been successfully installed on your computer.

ShowREADME = Yes

FileName = <APPL_MODULE1_DIR>\readme.htm

PerformActionAtEnd = No

[Finish_Upgrade]

ShowREADME = No

[Finish_Uninstall]

ShowREADME = No

Tips and Tricks

J

Tip 1: Creating Files During the Installation Process

It is often necessary to create database files during the installation process. There are a few ways of doing this.

The first method is used when the DBMS is known in advance, for example MSSQL or Oracle. When it is necessary to create a MSSQL file, you can use the MSSQL utility osql.exe file, which enables the user to perform an SQL statement from the command line. An example is provided in Appendix I, Application Installation Example.

Another method is more dynamic and involves using uniPaaS. You can create an application that creates the files that are needed and, if necessary, the data, as a regular application. This can be included into the application installation utility as an application where you can define a specific Magic.ini file for it.

We can take advantage of the fact that the utility commands are executed after uniPaaS is already installed and create a command that looks like:

[Command1]

Program=mgrntw.exe

Path=<MAGICDIR>

Parameters = /INI=create.ini /StartApplication=2

Wait_Parameter = Yes

Message = Creating the application data

Tip 2: Creating or Deleting MSMQ Queues

You can create or delete MSMQ queues during the installation by using Visual Basic scripts.

To create an MSMQ queue:

1. Create a file called CreateQueue.vbs text file in the Configuration Files directory (Replace the name of the queue).
   The file should contain the following text:
   set iq = CreateObject ("MSMQ.MSMQQueueInfo")
   iq.PathName = ".\PRIVATE$\TestQueue"
   iq.Label = "Test Queue"
   iq.Create
   iq.Refresh
2. From the Application Installation utility, you can execute the script in the text file displayed below in the command option:
   <WINSYSDIR>cscript.exe <CONFIG_DIR>\CreateQueue.vbs

To delete an MSMQ queue:

1. Add the DeleteQueue.vbs text file in the Configuration Files directory (Replace the name of the queue). An example of how to delete a queue is shown below.
   set iq = CreateObject ("MSMQ.MSMQQueueInfo")
   iq.Label = "Test Queue"
   iq.PathName = ".\PRIVATE$\TestQueue"
   iq.Delete
2. From the Application Installation utility, you can execute the script in the text file in the format displayed below in the command option:
   <WINSYSDIR>cscript.exe <CONFIG_DIR>\DeleteQueue.vbs