

tools4ever



User Management Resource Administrator

User Management Resource Administrator

UMRA.COM

Copyright © 2005, Tools4Ever B.V. All rights reserved. No part of the contents of this user guide may be reproduced or transmitted in any form or by any means without the written permission of Tools4Ever.

DISCLAIMER - Tools4ever will not be held responsible for the outcome or consequences resulting from your actions or usage of the informational material contained in this user guide. Responsibility for the use of any and all information contained in this user guide is strictly and solely the responsibility of that of the user.

All trademarks used are properties of their respective owners.

Contents

UMRA COM	3
Introduction	3
Component Object Model (COM).....	3
COM objects and interfaces	4
COM registration	4
Type library	4
UMRA COM - Main functions.....	5
UMRACOM.DLL	5
ASP and IIS	6
UMRA COM objects and software	7
Scenarios for using UMRA COM	8
Executing an UMRA project on the UMRA Service	8
Accessing the data of a form table	9
Accessing the data of a table variable	9
UMRA COM object reference.....	10
UMRA COM object: Umra.....	10
UMRA COM object: UmraFormProject	17
UMRA COM object: UmraFormTable.....	18
UMRA COM object: UmraDataTable	19
UMRA COM in VB scripts	21
Configuring the UMRA project.....	21
Create user action	22
Input and output variables	22
MS Access database.....	23
Jet engine	23
Microsoft Office Access.....	24
IIS configuration Windows 2003	25
Visual Basic script	26
Visual Basic Script	26
Script section: Setting up the database connection.....	29
Script section: Connecting to the UMRA Service	30
Script section: Executing projects on UMRA Service	31
Testing and executing the script.....	33
UMRA COM in IIS	37
Introduction to using UMRA COM in IIS	37
Security and authentication.....	39
Creating the web site.....	40
Configuring the UMRA project.....	44
Setting up the IIS web site	45
Web site page: ShowResults.asp.....	46
Explanation of the ASP code	47
Testing the web site	50
UMRA Service log file.....	52
Web site page: CreateAccount.asp.....	55

CHAPTER 1

UMRA COM

In This Chapter

Introduction.....	3
Component Object Model (COM)	3
UMRA COM - Main functions	5

Introduction

This document describes how Microsoft's Component Object Model (COM) is used in combination with User Management Resource Administrator (UMRA). Using COM with UMRA is referred to as UMRA COM.

Component Object Model (COM)

The Component Object Model (COM) is a technology that is used by applications to interact with other applications. The COM technology is designed by Microsoft. The most important Microsoft applications that use COM are:

Internet Information Services

Office applications

Visual Basic Scripting (VB) and Visual Basic

COM objects and interfaces

In principle, a COM object is a piece of software that implements one or more functions. Different COM objects support different functions. The functions of a COM object are accessible by means of the interface of the COM object. The COM object itself is regarded as a black box that implements one or more functions accessible through its interfaces.

Applications that support COM can create COM objects. By accessing the interface functions of the COM object, the functions of the COM object are executed by the calling application. The syntax to create COM objects and access the interface functions of a COM object is extremely general: this is the main reason why COM objects can be used by so many different applications: The same COM object can be created and used in ASP-pages, Word documents and Visual Basic scripts.

All applications that support COM use some kind of programming or script language to implement COM. The procedure used is always the same:

The COM object is created;

The interface functions of the COM object are accessed;

Returned variables can be processed in the application.

An application can use multiple COM objects and COM objects can use other COM objects.

COM registration

In order for an application to use a COM object and the interface functions of the object, the COM object must be registered. Once a COM object is registered, all applications that support COM can use the COM object. In most cases, COM object are registered automatically.

Type library

In most cases, the COM object code is contained in a file with the .DLL extension. Such a file contains all code needed to use the COM objects it implements. Besides the COM objects, the file can also contain a so called type library that describes the COM objects and the interfaces that are used to access the COM objects.

UMRA COM - Main functions

User Management Resource Administrator contains several COM objects as part of the UMRA Automation module. The main functions of UMRA COM are:

Connect to an UMRA Service and execute a UMRA project on the UMRA Service;

Read and process the information returned by the project variables.

UMRACOM.DLL

All UMRA COM objects are contained in a single file: UMRACOM.DLL. The COM objects are automatically registered on the computer when the UMRA Console or UMRA Service application is installed.

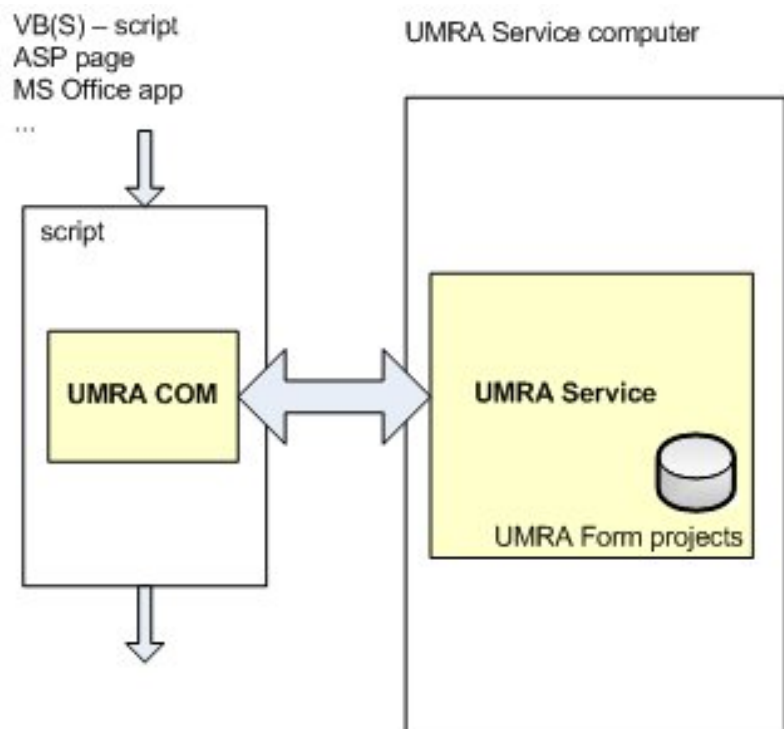


Figure 1: UMRA COM objects used in a script accessing the UMRA Service.

Any (form) project that is maintained by the UMRA Service can be accessed using UMRA COM.

ASP and IIS

One of the many applications which can utilize UMRA COM can be used is Microsoft's Internet Information Services (IIS). In web applications running on IIS, Active Server Pages (ASP) may contain UMRA COM objects to interface with the UMRA Service. A substantial part of this document focuses on UMRA COM objects used in ASP pages.

There is a lot of documentation available on both COM and ASP. This document only briefly describes the backgrounds of COM and ASP. A number of references to these subjects are listed in at the end of this document.

UMRA COM is part of the User Management Resource Administrator Automation module (UMRA Automation). To use UMRA COM, a license of UMRA Automation is required.

CHAPTER 2

UMRA COM objects and software

UMRA COM objects

UMRA COM supports four types of COM objects. The main functions of the UMRA COM objects are to:

- access the UMRA Service,
- *execute UMRA projects on the UMRA Service*
- manage the data retrieved from the UMRA Service.

UMRA COM object	Description
<i>Umra</i>	The main UMRA COM object. Used to connect to the UMRA Service, execute an UMRA Form project on the UMRA Service and manage variables, values and form project tables.
<i>UmraFormProject</i>	Intermediate COM object used to access the data of a form table.
<i>UmraFormTable</i>	COM object used to access the data of a form project table.
<i>UmraDataTable</i>	COM object used to access the data of table variable.

Table 1: The UMRA COM objects.

UMRA COM software

The COM objects are contained in a single DLL: **UMRACOM.DLL**. This DLL is part of the **UMRA Automation** module and installed with the **UMRA Console** application. The COM objects are registered automatically. Part of the UMRA COM software is the type library that describes the COM objects and interfaces of UMRA COM.

In This Chapter

Scenarios for using UMRA COM.....	8
UMRA COM object reference	10

Scenarios for using UMRA COM

The UMRA COM objects can be used in many environments. The most important functions implemented with UMRA COM objects are the following three scenarios:

1. *Executing an UMRA project on the UMRA Service* and processing the results.
2. *Accessing the data of a form table* contained in the form of an UMRA Form project.
3. *Accessing the data of a table variable* generated by a UMRA project script.

Executing an UMRA project on the UMRA Service

The procedure to execute a UMRA project on the UMRA Service is always the same. The project must be setup using the **UMRA Console** application. The project type must be a form project. The project only needs to contain a script, not a form. A form project with a form can also be used. The security settings of the project must be set so that the project can be accessed from the security context of the UMRA COM objects accessing the UMRA Service. In general, the UMRA COM objects are executed with the security context of the user account that is logged on to the computer that runs the application that uses the UMRA COM objects.

Executing a project on the UMRA Service

Procedure to execute the script of the UMRA project on the UMRA Service using UMRA COM:

1. An instance of the **Umra COM** object is created.
2. The **Umra.Connect** method is used to connect to the UMRA Service.
3. The list with variables maintained by the **Umra COM** object is created. All variables that are required to execute the script of the UMRA project must be initialized. For each variable, the name of the variable and the value must be specified. Normally, the values are taken from the application that uses the UMRA COM objects. Several methods of the **Umra COM** object are available to setup the list with variables.
4. The method **Umra.ExecuteProjectScript** is called to request the UMRA Service to execute the script of the specified project. On return, the variables updated by the script are stored in the list with variables.
5. The list with variables can be accessed to process the returned information.

Accessing the data of a form table

UMRA Form projects can contain a form. The form can contain a table presented with UMRA Forms. The table data can be access using UMRA COM using the following procedure:

1. An instance of the following UMRA COM objects must be created: **Umra**, **UmraFormProject**, **UmraFormTable**.
2. The **Umra.Connect** method is used to connect to the UMRA Service.
3. The method **Umra.LoadFormProject** is used to load a form project in the UMRA COM object. As one of the arguments, the **UmraFormProject** object is passed. On success, the passed COM object **UmraFormProject** contains the form of the project, including the form tables that are part of the form.
4. The method **UmraFormProject.GetFormTable** is called to load the **UmraFormTable** object. The object **UmraFormTable** is one of the arguments of the method.
5. On success, the COM object **UmraFormTable** contains the table data. The method **UmraFormTable.GetCellText** can be used to access the table data.

Accessing the data of a table variable

UMRA supports many script actions that generate table data. Such a table is normally stored in a single variable and the action is part of a UMRA project script. When the project is executed using UMRA COM, the variable is returned. The following procedure describes how to access the data of the table variable:

1. Instances of the UMRA COM objects **Umra** and **UmraDataTable** are created.
2. The UMRA COM object **Umra** is connected to the UMRA Service and a project is executed. See **Executing an UMRA project on the UMRA Service** for more information. The list with variables maintained by the **Umra** COM object now contains a variable with table data.
3. The member **Umra.GetVariableDataTable** is called to load the table data. The method requires two arguments: The name of the variable and the **UmraDataTable** *object*. On success, the **UmraDataTable** contains the table data of the variable.
4. The method **UmraDataTable.GetCellText** is used to access the data of the individual table cells.

UMRA COM object reference

UMRA COM object: Umra

This is the main UMRA COM object. The object is used to connect to an UMRA Service, execute a project on the UMRA Service and retrieve data from the UMRA Service.

The Umra COM object maintains the following information:

Umra.ClearVariables

void ClearVariables()

Argument	Type	Description
no arguments..		

The interface method deletes all variable-value pairs from the list with variables maintained by the UMRA COM object. When completed, the list is empty.

Umra.Connect

long Connect([in] BSTR Host, [in] long PortNumber)

Argument	Type	Description
Host	in	The name of the host that runs the UMRA Service to which the COM object must connect. The name can be specified as DNS or NETBIOS name.
PortNumber	in	The port of host running the UMRA Service. The default port is 56814.

The interface method connects to an UMRA Service. The name of the computer running the UMRA Service and the port number must be specified. Normally, this is the first method called when the UMRA COM object is created.

When the connection is established, the connection parameters are stored within the object. The COM object can be connected to only one UMRA Service at one time. The return value should be zero on success. If the return value is not zero, the COM object is not connected to the UMRA Service.

Umra.ExecuteProjectScript

long ExecuteProjectScript([in] BSTR ProjectName)

Argument	Type	Description
ProjectName	In	The name of the project of which the script must be executed by the UMRA Service.

The interface method requests the UMRA Service to execute the script of the specified project immediately. The COM object must be connected to the UMRA Service (see `Umra.Connect`) and the project must be maintained by the UMRA Service. The current list with variable-value pairs is used to execute the script of the project.

When the project is executed successfully, the return value is zero. In this case, the list with variables is updated as defined in the UMRA project script. The returned variables can be used for further processing. The log information in the COM object is updated. The method *Umra.GetLogMsg* can be used to retrieve the log information.

Umra.GetHostName

long GetHostName([out] VARIANT * HostName)

Argument	Type	Description
HostName	Out	The name of the host to which the UMRA Service is connected.

The interface method retrieves the name of the host to which the UMRA COM object is connected. The name corresponds with specified host of method `Connect`.

The return value should be zero on success. If the return value is not zero, the COM object is not connected to the UMRA Service and no name is returned.

Umra.GetLogMsg

long GetLogMsg([out] VARIANT * Msg)

Argument	Type	Description
Msg	Out	The log information stored by the COM object.

The interface method retrieves the log information stored by the interface. The log information is refreshed when the interface communicates with the UMRA Service. The log information describes the success or failure of the request that was last executed. When a new method is called that involves communication with the UMRA Service, the contents is always reset first.

Umra.GetPortNumber

long GetHostPortNumber([out] long * PortNumber)

Argument	Type	Description
PortNumber	Out	The number of the port to which the COM object is connected.

The interface method retrieves the port number of the host to which the UMRA COM object is connected. The number corresponds with specified port number of method **Connect**.

The return value should is zero on success. If the return value is not zero, the COM object is not connected to the UMRA Service and no port number is returned.

Umra.GetScriptExecutionInfo

long GetScriptExecutionInfo(

[out] long * ScriptErrorCount,
[out] VARIANT * ScriptMessage)

Argument	Type	Description
ScriptErrorCount	Out	The number of errors occurred executing the script of the project on the UMRA Service..
ScriptMessage	Out	The log information generated by the UMRA Service when the last project was executed.

The interface method retrieves the logging and error count information from the last project executed through the COM object on the UMRA Service. The number of errors that occurred when the script was executed is returned and the log information generated by the UMRA Service.

On success, the method returns zero. If the information cannot be obtained from the COM object, a nonzero value is returned.

Umra.GetVariableDataTable

long GetVariableDataTable([in] BSTR * VariableName, [in] IUnknown * pDataTable)

Argument	Type	Description
VariableName	In	The name of the variable that holds the table data to be retrieved.
pDataTable	In/out	The retrieved table data. The argument must be specified as one of the other UMRA COM object types: UmraDataTable. This COM object can then be used to access the data of the table.

The interface method retrieves the data of a table variable. When an UMRA Form project generates a variable with table data, the variable and table data is returned in the list with variables maintained by the UMRA COM object. To access the table data, a special UMRA COM object is used: **UmraDataTable**. An instance of such a COM object must be passed as one of the arguments of this method. On success, the table data is copied from the variable into the passed COM object. The interface methods of the **UmraDataTable** object can then be used to access the table data.

On success, the method returns zero. If the variable is not found in the list with variables, or when the variable does not contains table data, a non zero value is returned.

Introduction to using UMRA COM in Visual Basic scripts

A very common usage of UMRA COM is in Visual Basic scripts. Visual Basic is a very general development environment to create Windows applications and scripts. Managing Active Directory directly from Visual Basic programs and scripts is possible but rather complex and not secure.

The integration of Visual Basic and User Management Resource Administrators using UMRA COM offers a very flexible and powerful solution to manage the complex Active Directory with simple and robust program. The complex Active Directory tasks are implemented with UMRA projects and executed by the UMRA Service. The projects are initiated using UMRA COM from within Visual Basic scripts/programs.

Since the UMRA projects are executed by the UMRA Service in such an environment, the Visual Basic scripts and programs can be delegated to less experienced users. The UMRA Service will check the access rights of the connecting user account before the UMRA project script is executed.

The technique to use UMRA COM objects in Visual Basic scripts is best described with an example.

Example project - Goal

Goal of the example application is to create a number of user accounts in Active Directory for which the first and last name are available from a table in an MS-Access database.

In this example, a simple UMRA project is configured on the UMRA Service. The project creates a user account in Active Directory. The input of the project is the first and last name of the new user account. The input values are passed using the variables %FirstName% and %LastName%.

The MS-Access database contains a table with first and last names. Goal is to create the user accounts for all table entries. A Visual Basic script with UMRA COM objects is used to read the data from the MS-Access database and create a user account for each row.

Umra.GetConnectionInfo

long GetConnectionInfo(

[out] VARIANT * ServerName,
[out] unsigned long * RpcPortNumber,
[out] unsigned long * RpcHandle)

Argument	Type	Description
ServerName	out	The name of computer (host) to which the COM object is connected.
RpcPortNumber	out	The port number used on the connected host.
RpcHandle	out	An internal identifier used by the COM object to communicate with the UMRA Service.

The interface method retrieves connection information from the UMRA COM object. The name of the host, port number and an internal identifier is retrieved. On success, the return value is zero. If the COM object is not connected to an UMRA Service, the method returns a non zero value.

Umra.GetVariableText

long GetVariableText([in] BSTR VariableName, [out] VARIANT* ValueText)

Argument	Type	Description
VariableName	in	The name of the variable of which the textual value must be retrieved.
ValueText	out	The text value of the specified variable.

The interface method retrieves the text value of a specified variable. The variable must be contained by the list of variables maintained by the COM object. The method is normally used when a project is executed on the UMRA Service and variable values are generated by the UMRA project script.

On success, the return value is zero. If the variable is not found in the list with variables, or if the variable contains no text value, a non zero value is returned.

Umra.GetVersion

long GetVersion(

[out] long * VersionMajor,
[out] long * VersionMinor,
[out] long * BuildNumber)

Argument	Type	Description
VersionMajor	Out	The major version number of the UMRA COM object
VersionMinor	Out	The minor version number of the UMRA COM object.
BuildNumber	Out	The build number of the UMRA COM object.

The interface method retrieves version information of the UMRA COM object. The information must correspond with an eventually connected UMRA Service. Otherwise, the COM object cannot connect to the UMRA Service.

Umra.LoadFormProject

long LoadFormProject([in] BSTR FormProjectName, [in] IUnknown * pFormProject)

Argument	Type	Description
FormProjectName	In	The name of the project for which the form must be obtained.
pFormProject	in/out	The returned form project. The argument must be specified as another UMRA COM object: UmraFormProject. This COM object can then be used to obtain form table objects.

The interface method retrieves the form of an UMRA Form project. The method is used to obtain table information from a table of an UMRA Form project.

On success, the return value is zero. If the specified form does not exist, or if the user has no access rights for the form, a non zero value is returned.

Umra.SetVariableBool

void SetVariableBool([in] BSTR VariableName, [in] VARIANT_BOOL ValueBool)

Argument	Type	Description
VariableName	in	The name of the variable set by this method. Example: %SomeFlag%.
ValueBool	in	The boolean value of the variable to set. Example: 0. A value of 0 corresponds with false. All other values correspond with true.

The interface method adds a boolean variable-value pair to the list of variables maintained by the COM object. The list is used to execute UMRA projects on the UMRA Service. To reset the list with variables, call method **ClearVariables**. The value of the variable must be either true or false. Other methods are available for different variable value data types.

When a variable with the same name already exists in the list with variables, it is overwritten by this method.

Umra.SetVariableLong

void SetVariableLong([in] BSTR VariableName, [in] long ValueLong)

Argument	Type	Description
VariableName	in	The name of the variable set by this method. Example: %SomeValue%.
ValueLong	in	The numeric value of the variable to set. Example: 5.

The interface method adds a numeric variable-value pair to the list of variables maintained by the COM object. The list is used to execute UMRA projects on the UMRA Service. To reset the list with variables, call method **ClearVariables**. The value of the variable must be numeric. Other methods are available for different variable value data types.

When a variable with the same name already exists in the list with variables, it is overwritten by this method.

Umra.SetVariableText

```
void SetVariableText([in] BSTR VariableName, [in] BSTR ValueText)
```

Argument	Type	Description
VariableName	In	The name of the variable set by this method. Example: %FirstName%.
ValueText	In	The textual value of the variable to set. Example: John.

The interface method adds a textual variable-value pair to the list of variables maintained by the COM object. The list is used to execute UMRA projects on the UMRA Service. To reset the list with variables, call method **ClearVariables**. The value of the variable must be text. Other methods are available for different variable value data types.

When a variable with the same name already exists in the list with variables, it is overwritten by this method.

UMRA COM object: UmraFormProject

This UMRA COM object is used to access a table that is part of a form of an UMRA Form project. The COM object should not be used to access the data of a variable that holds table data. The UMRA COM object **UmraDataTable** must be used for table data variables instead.

UmraFormProject.GetFormTable

```
long GetFormTable([in] BSTR * FormTableName, [in] IUnknown * pFormTable)
```

Argument	Type	Description
FormTableName	In	The name of the form table as defined in the UMRA form.

pFormTable	In/out	The retrieved form table. The argument must be specified as one of the other UMRA COM object types: UmraFormTable . This COM object can then be used to access the data of the table.
------------	--------	--

The interface method retrieves the table of a form. When the form of an UMRA Form project contains a table, this method is used to access the table of the form. To access the form table, a special UMRA COM object is used: **UmraFormTable**. An instance of such a COM object must be passed as one of the arguments of this method. Before this method is called, the method **Umra.LoadFormProject** must be executed first. On success, the form table data is copied into the passed COM object. The interface methods of the **UmraFormTable** object can then be used to access the table data.

On success, the method returns zero. If the form table is not found in the UMRA project maintained by the UMRA COM object, a non-zero value is returned.

UMRA COM object: **UmraFormTable**

This UMRA COM object is used to access the data of a form project table. See also **UmraFormProject.GetFormTable**.

The COM object should not be used to access the data of a variable holding table data. Instead, the UMRA COM object **UmraDataTable** should be used for table data variables.

UmraFormTable.GetCellText

long GetCellText(

[in] long RowIndex,
[in] long ColumnIndex,
[out] VARIANT* CellText)

Argument	Type	Description
RowIndex	In	The index of the row (0,1,2,...,N-1) that contains the requested cell.
ColumnIndex	In	The index of the column (0,1,2,...,M-1) that contains the requested cell.
CellText	Out	The result text value of the requested cell.

The interface method retrieves the text value from one specific cell of the form table. The method returns zero on success. If the cell does not exist or when the cell value cannot be converted into text, a non zero value is returned.

UMRA COM object: UmraDataTable

This UMRA COM object is used to access the table data of a variable. If the script of an UMRA Form is executed by UMRA COM (`Umra.ExecuteProjectScript`), a variable with table data can be generated. This variable and table data value is returned to the UMRA COM project. To access the variable table data value, first the method `Umra.GetVariableDataTable` must be called. With this step, an instance of COM object `UmraDataTable` is initialized. Next, the single interface function of this COM object is used to access the table data.

UmraDataTable.GetCellText

long GetCellText(

[in] long RowIndex,
 [in] long ColumnIndex,
 [out] VARIANT* CellText)

Argument	Type	Description
RowIndex	In	The index of the row (0,1,2,...,N-1) that contains the requested cell.
ColumnIndex	In	The index of the column (0,1,2,...,M-1) that contains the requested cell.
CellText	Out	The result text value of the requested cell.

The interface method retrieves the text value from one specific cell of the table data. The method returns zero on success. If the cell does not exist or when the cell value cannot be converted into text, a non zero value is returned.

CHAPTER 3

UMRA COM in VB scripts

In This Chapter

Configuring the UMRA project	21
MS Access database.....	23
Visual Basic script.....	26

Configuring the UMRA project

In the example, an UMRA project is used to create a user account. The project takes two input variables: %FirstName% and %LastName%. From these names, the project generates a unique user account name (%UserName%) and creates the user account. Also, a random password (%Password%) is generated.

The UMRA project is a form project that contains only a script, not a form. The UMRA form project is contained in file

.\Example Projects\Automation\VBScript\MsAccess\ CreateAccount.ufp

relative to the UMRA Console program directory.

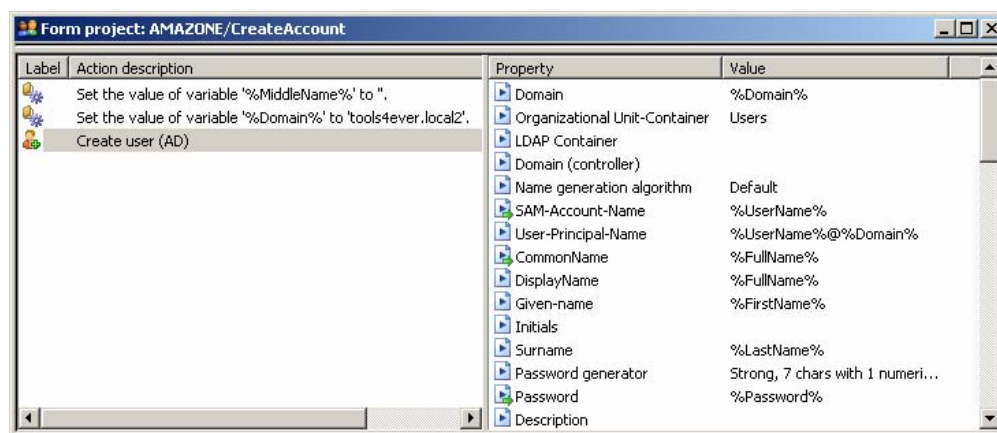


Figure 2: The script of the UMRA Form project used to create the account.

The script contains 3 lines only. In the first 2 lines the following variables are set: %MiddleName% and %Domain%. The %MiddleName% variable is set to an empty string. The %MiddleName% is used in the name generation algorithm and always specified as an empty string. The %Domain% is set equal to the DNS name of the domain in which the user accounts are created: **tools4ever.local2**. You need to adjust this value to make the project work in your own environment.

Create user action

The **Create user (AD)** script action is the main action of the form project. The action creates the user account in the specified domain. The inputs of the script (action) are the values of the variables %FirstName% and %LastName%. These variables are used by the name generation algorithm. The **Default** name generation algorithm is specified for this property.

The algorithm uses 3 input variables: %FirstName%, %MiddleName% and %LastName%. The %MiddleName% is always empty, the %FirstName% and %LastName% variables are specified in the Visual Basic Script. The outputs of the name generation algorithm are the values of the variables %UserName% and %FullName%. These values are used to create the user account. The %UserName% variable is exported to the list of variables. The password of the new user account is generated as part of the action and exported to variable %Password%.

In UMRA, variables play an important role. Some variables are used as input and others as output variables. Other variables are generated by the script and only used in the script actions. The interface functions of the UMRA COM objects support input and output variables.

Input and output variables

The following table lists the input and output variables that are used by this example project and communicated with UMRA COM.

Variable	Type	Description
%FirstName%	Input	First name of the user account that must be created. Specified in the database read by the Visual Basic script.
%LastName%	Input	Last name of the user account that must be created. Specified in the database read by the Visual Basic script.
%UserName%	Output	The resulting name (SAM account name) of the created user account. Presented to the end-user in the browser.

%Password%	Output	The password generated and set for the created user account. Presented to the end-user in the browser.
------------	--------	--

Table 2: In- and output variables of the example project.

Note that for the form project, security access rights must be configured: The end-users that are allowed to run the script of this project must be configured. The end-users are the user accounts logged on to the computer that run the Internet Explorer and access the UMRA application.

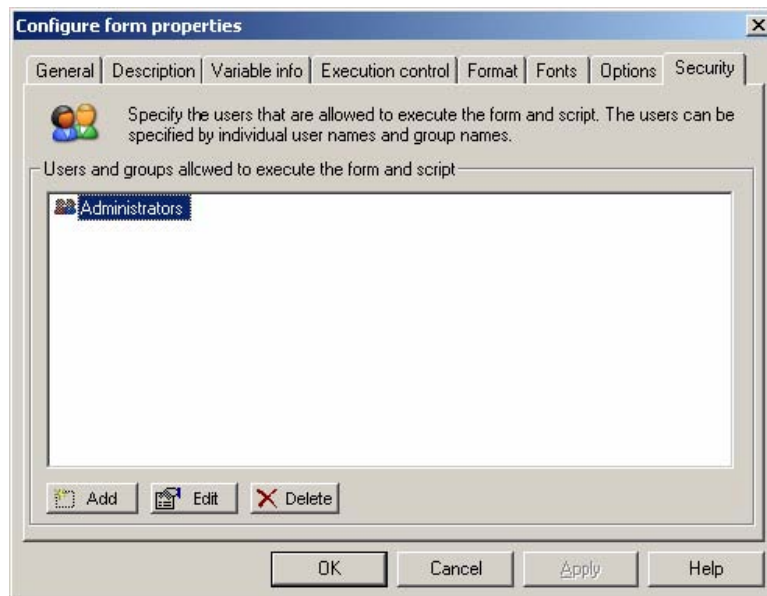


Figure 3: The security access rights specified for the UMRA project that creates the account.

In this example project, the **Administrators** are configured as the users that are allowed to run this project.

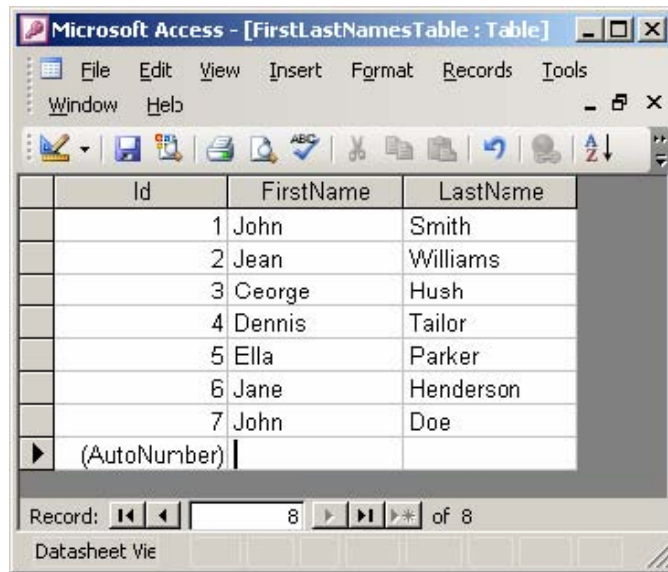
MS Access database

Jet engine

The database is a simple MS-Access (Jet-engine) database with a single table only. The database can be found in the following location:

.\Example Projects\Automation\VBScript\MsAccess\FirstLastNames.mdb

relative to the UMRA Console program directory. The database holds table `FirstLastNamesTable` with the columns `Id`, `FirstName`, and `LastName`.



Id	FirstName	LastName
1	John	Smith
2	Jean	Williams
3	George	Hush
4	Dennis	Taylor
5	Ella	Parker
6	Jane	Henderson
7	John	Doe
(AutoNumber)		

Microsoft Office Access

Note that in order to use the database with UMRA in Visual Basic Script, you do not need to have Microsoft Office Access installed. The Jet-engine to access the database is installed by default.

IIS configuration Windows 2003

This topic describes how to setup IIS on a computer running Window Server 2003, Standard Edition. The procedure is similar for computer running other versions of Windows, including all versions of Windows 2000.

When IIS is not already installed, log on as an administrator. Select **Start, Control Panel, Add or Remove Programs**. Click button **Add/Remove Windows Components**. In the **Windows Components Wizard** window, enable and select **Application Server** from the list.

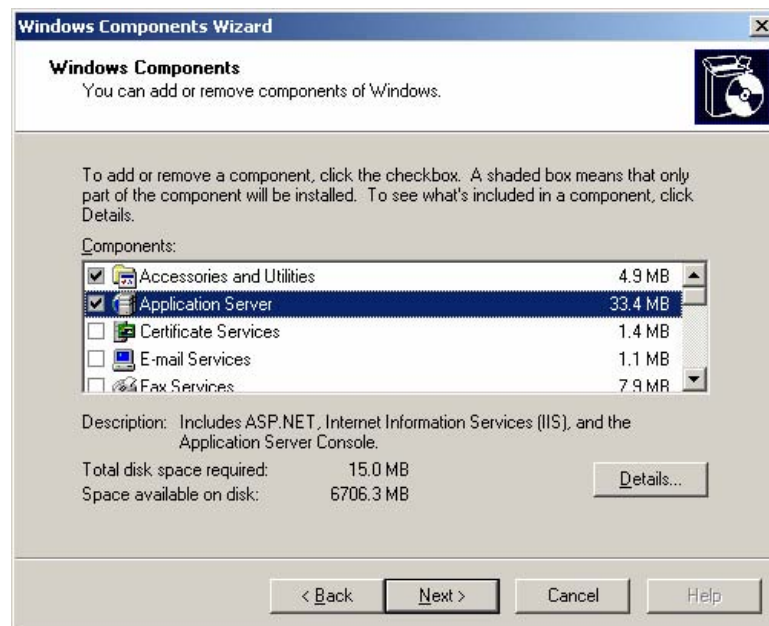


Figure 6: Setting up IIS by enabling Application Server.

Click **Next** and **Finish** to install the selected options. To complete the setup, you need the Windows installation CD. When done, the list with services running on the computer will list the IIS service: **World Wide Web Publishing Service**.

When ready, the IIS Service needs to be configured. Select **Start, Administrative Tools, Internet Information Services (IIS) Manager**. Browse to the current computer and select **Web Service Extensions**.

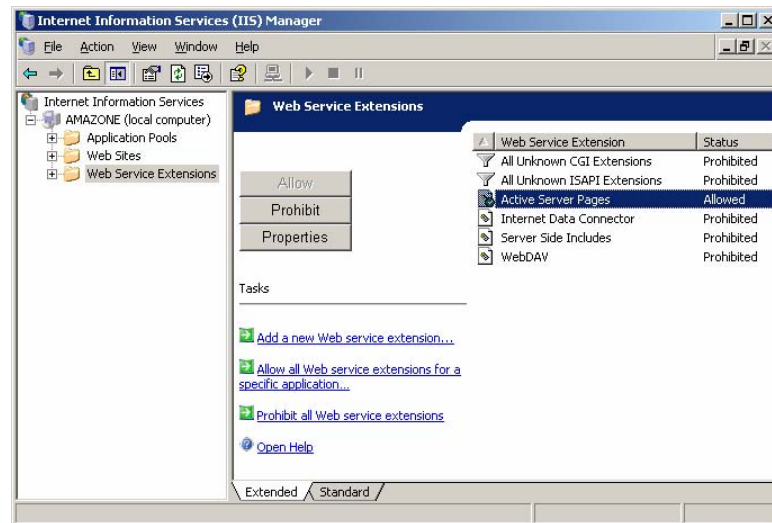


Figure 7 Allow Active Server Pages to run on IIS.

From the list with **Web Service Extensions**, select **Active Server Pages** and click the **Allow** button. This will allow ASP pages to run as part of a web site maintained by the IIS server.

Visual Basic script

The Visual Basic script is a standard script that uses a database connection to read the MS-Access database. UMRA COM objects are used to execute the project that creates a user account on the UMRA Service.

Visual Basic Script

The Visual Basic script can be found at the location

```
.\Example Projects\Automation\VBScript\MsAccess\
CreateAccountAccess.vbs
```

relative to the UMRA Console program directory.

The following listing shows the complete script. Note that the script is kept as simple as possible to make it easier to understand. More error handling should be added to the script in operational environments.

```
Dim adoConn, adoRS

Set adoConn = CreateObject("ADODB.Connection")

Set adoRS = CreateObject("ADODB.Recordset")

adoConn.Provider = "Microsoft.Jet.OLEDB.4.0"

adoConn.Open "FirstLastNames.mdb"

Set adoRS.ActiveConnection = adoConn

adoRS.Open "SELECT * FROM FirstLastNamesTable"

Dim Umra, Username, UserPassword

Set Umra = CreateObject("UmraCom.Umra")

RetVal = Umra.Connect("AMAZONE", 56814)

While adoRS.EOF = False

    Umra.SetVariableText "%Firstname%",
adoRS("FirstName")

    Umra.SetVariableText "%LastName%",
adoRS("LastName")

    RetVal=Umra.ExecuteProjectScript("CreateAccount")

    RetVal=Umra.GetVariableText("%Username%",
Username)

    RetVal=Umra.GetVariableText("%Password%",
UserPassword)

    wscript.echo "User created: " & Username &
" - " & UserPassword

    adoRS.MoveNext

Wend
```

```
IfRetVal = 0 Then
    WScript.Echo "Project executed
successfully, code: " &RetVal
Else
    WScript.Echo "Project execution failed,
code: " &RetVal
End If
```

In the next section, parts of the scripts are shown with some comments for each section.

Script section: Setting up the database connection

ADO database connection - OLE DB provider

The first section of the script is used to setup a connection with the database. In this script, the Microsoft ADO (ActiveX Data Objects) standard is used to access the database. This is a very common technique used in VBScript to access a database. With ADO, the data is accessed using the OLE DB provider. Almost all database types can be accessed using OLE DB (including ODBC connections) so this method can be used for almost all databases.

```
Dim adoConn, adoRS

Set adoConn = CreateObject("ADODB.Connection")

Set adoRS = CreateObject("ADODB.Recordset")

adoConn.Provider = "Microsoft.Jet.OLEDB.4.0"

adoConn.Open "FirstLastNames.mdb"

Set adoRS.ActiveConnection = adoConn

adoRS.Open "SELECT * FROM FirstLastNamesTable"
```

Two script variables are initialized: the connection (`adoConn`) and the record set (`adoRS`). With the `CreateObject` function, the `ADODB.Connection` and `ADODB.RecordSet` COM objects are created. The `CreateObject` function creates an instance of the specified COM object. So this is also COM, not UMRA COM but ADO COM.

Next, the `Provider` of the connection object is set to `Microsoft.Jet.OLEDB.4.0`. The provider member specifies the type of the database connection. For a different database, this provider specification differs. The database is opened with the `Open` method of the `ADODB.Connection` object. The name of the database file, `FirstLastNames.mdb` is specified as the argument of the call.

With the statement `Set adoRS.ActiveConnection = adoConn` the recordset object is initialized. The `Open` method of the `ADODB.Recordset` is then used to perform a query in the database: `SELECT * FROM FirstLastNamesTable`: All records from the table are returned and can be accessed through the recordset object.

Script section: Connecting to the UMRA Service

The database connection is not initialized. For each record returned from the database, a user account must be created. Before the loop to create the accounts is implemented, a connection must be setup with the UMRA Service. This is done by creating the **Umra** COM object of UMRA COM and calling the **Connect** method of the object.

```
Dim Umra, Username, UserPassword

Set Umra = CreateObject("UmraCom.Umra")

RetVal = Umra.Connect("AMAZONE", 56814)
```

First, the variables are declared. The **Umra** variable will hold the COM object. The variables **Username** and **UserPassword** will hold the text values of the user name and password for the created user account. The values will be shown to the end user

The UMRA COM object **Umra** is then created with the **CreateObject** call. The argument **UmraCom.Umra** specifies the UMRA COM library and the type of object (**Umra**) of which an instance must be created. If the UMRA COM library is not installed and/or registered, the **CreateObject** call will fail.

The UMRA COM object now connects to the UMRA Service with the statement **RetVal=Umra.Connect("AMAZONE",56814)**. The **Connect** method takes two arguments: the name of the computer that runs the UMRA Service and the port number used by the UMRA Service. 56814 is the default UMRA Service port number.

On success, the return value of the **Connect** method is zero. The UMRA COM object is now connected to the UMRA Service.

Script section: Executing projects on UMRA Service

The database connection is initialized and the UMRA COM object is connected to the UMRA Service. Now the loop is implemented. For each returned database record, the variables list of the UMRA COM object is initialized and the UMRA Service is requested to execute the UMRA project that creates the user account.

```
While adoRS.EOF = False

    Umra.SetVariableText "%Firstname%",
adoRS("FirstName")

    Umra.SetVariableText "%LastName%",
adoRS("LastName")

   RetVal=Umra.ExecuteProjectScript("CreateAccount")

   RetVal=Umra.GetVariableText("%Username%",
Username)

   RetVal=Umra.GetVariableText("%Password%",
UserPassword)

    wscript.echo "User created: " & Username &
" - " & UserPassword

    adoRS.MoveNext

Wend
```

The **While ... Wend** construction is used for the loop. The loop is terminated when no more records are found: when `adoRS.EOF=False` no longer holds.

For each cycle of the loop, first the list with variables maintained by the UMRA COM object is initialized. The method `SetVariableText` is used. The method takes two arguments: the name of the variable (`%FirstName%`) and the value. The value is copied from the record set: `adoRS("FirstName")`. Here, `FirstName` is the name of the column of the database table from which the value must be obtained. Using this method, the `%FirstName%` and `%LastName%` UMRA variables are initialized.

With method `Umra.ExecuteProjectScript` the UMRA Service is requested to execute the passed project. With some other information, the list with variables and values is sent to the UMRA Service. The UMRA Service will check the access rights of the requesting user account, load the project and execute the script of the project. The variables updated or generated by the script of the project are returned to the UMRA COM object. On success, the return value is zero.

The `Umra.GetVariableText` method of the UMRA COM object is now used to retrieve the values of the variables `%UserName%` and `%Password%`. The values are stored in the script variables `Username` and `UserPassword`. With the standard `wscript.echo` method, the user name and password of the created user account is presented to the end user.



For each user account created, the message above is shown. In a more practical script, this is probably not convenient and should be changed.

Finally, the record set moves to the next record: `adoRS.MoveNext`.

Testing and executing the script

To execute the script, log on to a computer of the domain with an administrative Active Directory account. Update or enter the Visual Basic script with your favorite editor. Make sure the UMRA Service maintains the project referenced in the script and that the UMRA COM object connects to the appropriate UMRA Service. To execute the script, open a command prompt and enter the name of the script:
CreateAccountAccess.vbs.

This will automatically initialize **Windows Scripting Host** to execute the script. For each account created, the UMRA Service is requested to execute the UMRA project and create the account. A message box is shown for every account that is created.

The UMRA Service log file will contain the log information generated by the service when the script is executed. A listing for a single user account is shown below.

```
14:57:14 01/05/2006 Variable 1: %Firstname%=John
```

```
14:57:14 01/05/2006 Variable 2: %LastName%=Smith
```

```
14:57:14 01/05/2006 Variable 3:  
%UmraFormSubmitAccount%=T4ELOC2\Administrator
```

```
14:57:14 01/05/2006 Creating AD account in  
specified domain: 'tools4ever.local2'.
```

```
14:57:15 01/05/2006 Creating AD account in  
container 'Users'.
```

```
14:57:15 01/05/2006 Creating AD account in  
Organizational Unit-Container:  
'LDAP://CN=Users,DC=tools4ever,DC=local2'.
```

```
14:57:15 01/05/2006 Creating AD user account in  
container/OU  
'LDAP://CN=Users,DC=tools4ever,DC=local2': Using  
name generation algorithm 'Default', 100  
iterations maximum for duplicate names.
```

```
14:57:15 01/05/2006 Creating AD user account in  
container/OU  
'LDAP://CN=Users,DC=tools4ever,DC=local2'. Common  
name of user set to 'John Smith'.
```

```
14:57:15 01/05/2006 Creating AD user account in  
container/OU  
'LDAP://CN=Users,DC=tools4ever,DC=local2'. SAM  
account name (username) of user set to 'smithj'.
```

14:57:15 01/05/2006 Creating AD user account in container/OU
'LDAP://CN=Users,DC=tools4ever,DC=local2'. LDAP attribute 'userPrincipalName' of object 'John Smith' set to 'smithj@tools4ever.local2'.

14:57:15 01/05/2006 Creating AD user account in container/OU
'LDAP://CN=Users,DC=tools4ever,DC=local2'. LDAP attribute 'displayName' of object 'John Smith' set to 'John Smith'.

14:57:15 01/05/2006 Creating AD user account in container/OU
'LDAP://CN=Users,DC=tools4ever,DC=local2'. LDAP attribute 'givenName' of object 'John Smith' set to 'John'.

14:57:15 01/05/2006 Creating AD user account in container/OU
'LDAP://CN=Users,DC=tools4ever,DC=local2'. LDAP attribute 'sn' of object 'John Smith' set to 'Smith'.

14:57:15 01/05/2006 Creating AD user account in container/OU
'LDAP://CN=Users,DC=tools4ever,DC=local2'. Password generated ('Strong, 7 chars with 1 numeric, 1 special (variable: %Password%)') and set in variable '%Password%'.

14:57:15 01/05/2006 Creating AD user account in container/OU
'LDAP://CN=Users,DC=tools4ever,DC=local2'. Boolean parameter 'Account disabled'=FALSE (101034). Result not changed.

14:57:15 01/05/2006 Creating AD user account in container/OU
'LDAP://CN=Users,DC=tools4ever,DC=local2'. Boolean parameter 'Password never expires'=FALSE (101032). Result not changed.

14:57:15 01/05/2006 Creating AD user account in container/OU
'LDAP://CN=Users,DC=tools4ever,DC=local2'. Boolean parameter 'Store password using reversible encryption'=FALSE (101033). Result not changed.

```
14:57:15 01/05/2006 Creating AD user account in
container/OU
'LDAP://CN=Users,DC=tools4ever,DC=local2'.
Boolean parameter 'Smart card is required for
interactive logon'=FALSE (101035). Result not
changed.

14:57:15 01/05/2006 Creating AD user account in
container/OU
'LDAP://CN=Users,DC=tools4ever,DC=local2'.
Boolean parameter 'Account is trusted for
delegation'=FALSE (101036). Result not changed.

14:57:15 01/05/2006 Creating AD user account in
container/OU
'LDAP://CN=Users,DC=tools4ever,DC=local2'.
Boolean parameter 'Account is sensitive and
cannot be delegated'=FALSE (101037). Result not
changed.

14:57:15 01/05/2006 Creating AD user account in
container/OU
'LDAP://CN=Users,DC=tools4ever,DC=local2'.
Boolean parameter 'Use DES encryption types for
this account'=FALSE (101038). Result not changed.

14:57:15 01/05/2006 Creating AD user account in
container/OU
'LDAP://CN=Users,DC=tools4ever,DC=local2'.
Boolean parameter 'Don't require Kerberos
preauthentication'=FALSE (101039). Result not
changed.

14:57:15 01/05/2006 Creating AD user account in
container/OU
'LDAP://CN=Users,DC=tools4ever,DC=local2'.
Account expiration date not specified for new
user 'John Smith' object.

14:57:15 01/05/2006 Creating AD user account in
container/OU
'LDAP://CN=Users,DC=tools4ever,DC=local2'. User
'John Smith' successfully created.

14:57:15 01/05/2006 Creating AD user account in
container/OU
'LDAP://CN=Users,DC=tools4ever,DC=local2'.
Password set for new user object 'John Smith'.

14:57:15 01/05/2006 Creating AD user account in
container/OU
'LDAP://CN=Users,DC=tools4ever,DC=local2'. Flag
'User Cannot Change Password' not changed from
default value (FALSE).
```

```
14:57:15 01/05/2006 Form message:  
'01/05/2006,14:57:14,T4ELOC2\Administrator,"Autom  
ation run script",OK,CreateAccount'
```

CHAPTER 4**UMRA COM in IIS****In This Chapter**

Introduction to using UMRA COM in IIS	37
Security and authentication	39
Creating the web site	40
Configuring the UMRA project	44
Setting up the IIS web site	45

Introduction to using UMRA COM in IIS

The functions of User Management Resource Administrator can be accessed through an internet browser by using UMRA COM object in IIS ASP pages.

In such an environment: 3 components are involved:

1. The internet browser that shows the web-pages and is used to enter input fields;
2. The IIS web-server to which the browser connects;
3. The UMRA Service, contacted by the UMRA COM objects contained on the ASP pages running on the IIS web-server.

As an example, consider a computer running Internet Information Services (IIS). One of the web sites maintained by IIS is used to create a user account with UMRA. To implement such an application, the web site contains ASP pages with UMRA COM objects to access the UMRA Service.



Figure 4: Example project to create a user account with UMRA using a browser.

When the end user enters the first, middle and last name and clicks the Create user button, the browser information is sent to Internet Information Services. The ASP page that processes the input from the browser creates an UMRA COM object. Through the interface functions of the COM object, the UMRA Service is contacted. The variables and values are initialized and the project is executed by the UMRA Service.

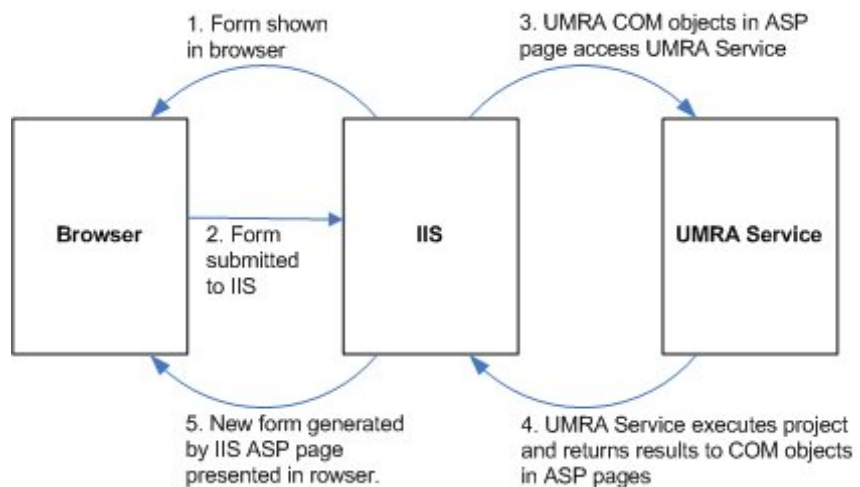


Figure 5: Sequence of steps of an UMRA application with UMRA COM on IIS. The ASP pages of the IIS web site contain UMRA COM objects that communicate with the UMRA Service.

In this chapter, the example shown is described in great detail for Windows 2003/2000. All files of the example project can be found at the following location:

.\Example Projects\Automation\ASP\CreateAccount

relative to the UMRA Console program directory.

Security and authentication

When running web sites with ASP pages containing UMRA COM objects, the preferred authentication method is **integrated Windows authentication**. With such configuration, the most simple and secure configuration is established.

The internet browser used is the Microsoft Windows Internet Explorer. On the computer that runs the Internet Explorer, a user is logged on to Active Directory. When the browser connects to IIS, the scripts contained by the ASP pages are executed on behalf of this user account. So when the UMRA COM object is used to access the UMRA Service, the UMRA Service will check the access rights for this target user account.

Creating the web site

Create a folder where the UMRA web site files will be stored. In this example this folder is: E:\UMRA\web site.

Start the **Internet Information Service Manager**, browse to the computer and right click option **web sites**. Select menu option **New, web site....** Follow the wizard instructions to setup a basic web site.

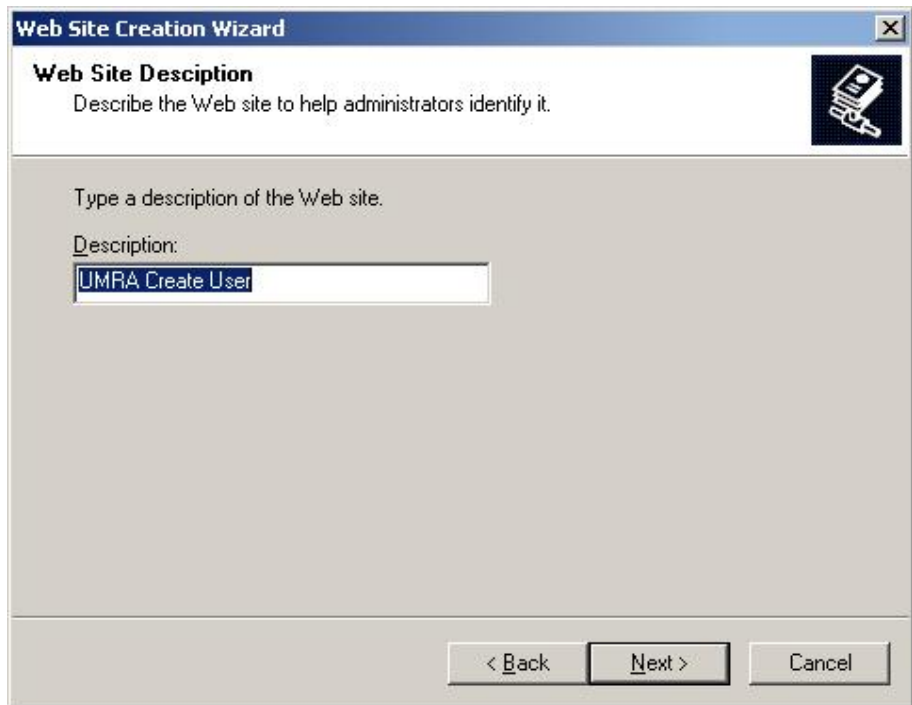
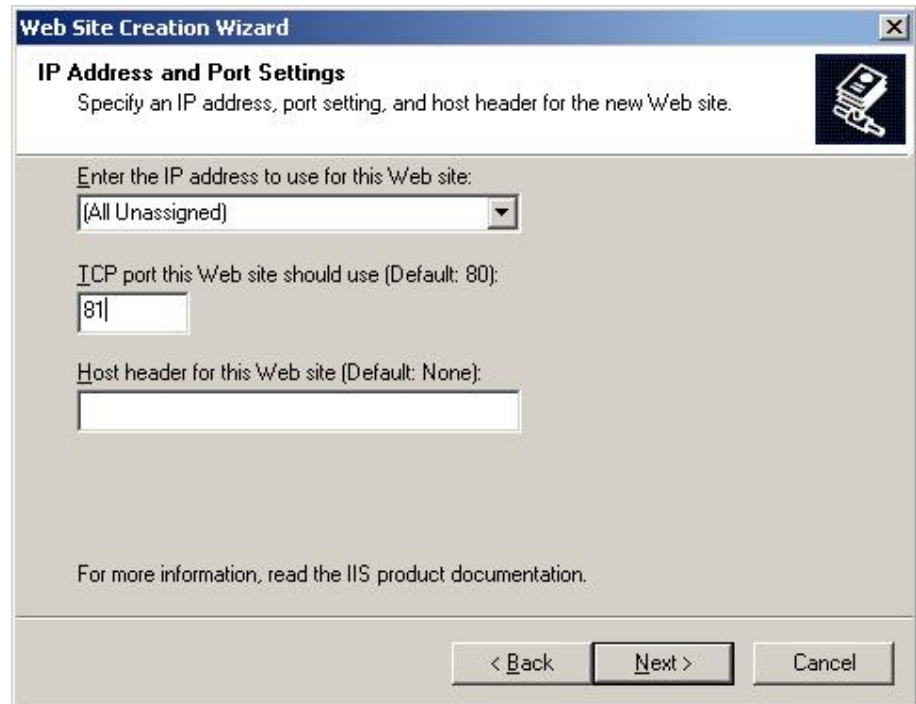


Figure 8: Enter any description of the web site.

The description of the web site is not that important. Just enter some text and click **Next** to continue.



The screenshot shows a Windows-style dialog box titled "Web Site Creation Wizard". The main heading is "IP Address and Port Settings" with a sub-instruction: "Specify an IP address, port setting, and host header for the new Web site." There are three input fields: a dropdown menu for IP address set to "[All Unassigned]", a text box for TCP port containing "81", and an empty text box for host header. At the bottom, there are three buttons: "< Back", "Next >", and "Cancel".

Web Site Creation Wizard

IP Address and Port Settings
Specify an IP address, port setting, and host header for the new Web site.

Enter the IP address to use for this Web site:
[All Unassigned]

TCP port this Web site should use (Default: 80):
81

Host header for this Web site (Default: None):
[Empty text box]

For more information, read the IIS product documentation.

< Back Next > Cancel

Figure 9: Specify the TCP port to be used by the web site.

In the wizard window to enter the **IP Address and Port Settings** you do not need to change the default options, except for the TCP port the web site should use. Each web site running on the computer must have a unique port. The default web site that is automatically installed uses default port 80. In order to use default port 80, you must first stop the default web site to prevent port conflicts.

In the example shown, port 81 is entered. Click **Next** to continue with the web site home directory specification.

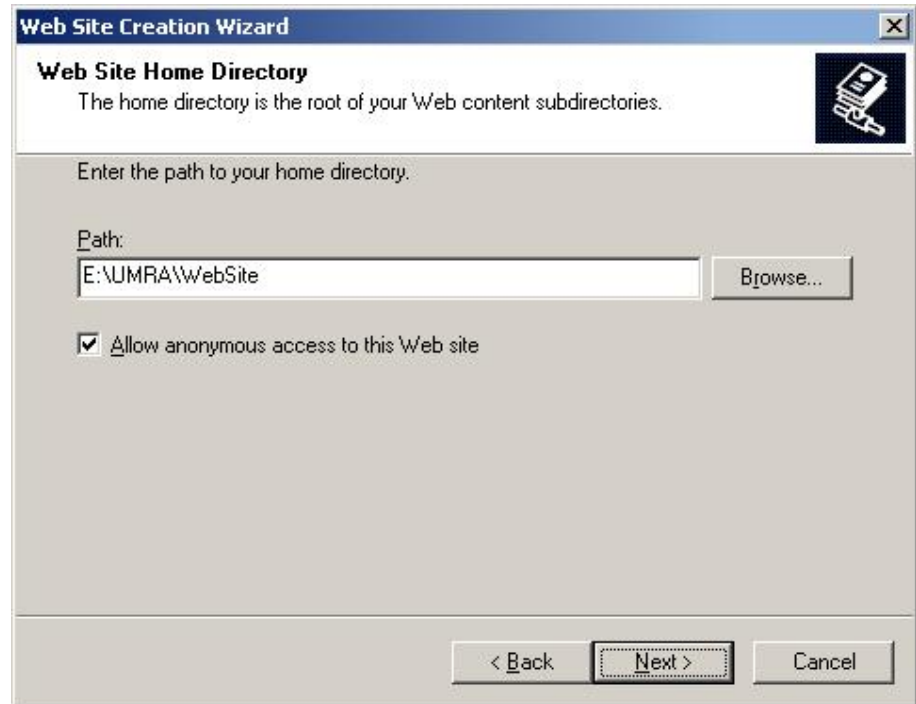


Figure 10: Enter the home directory of the web site. The directory must exist but does not need to contain any files yet.

Enter the name of the folder that is going to contain the web site ASP and HTML files. In the example shown, this folder is E:\UMRA\web site. The folder needs to exist but does not need to contain any files yet. Click **Next** to continue with the configuration of the web site Access Permissions.

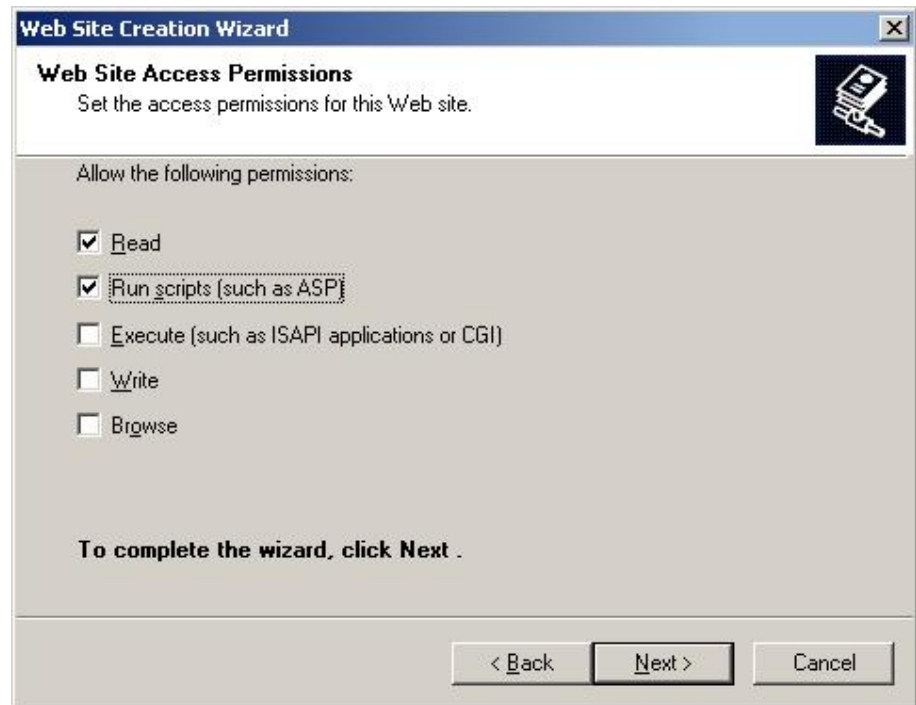


Figure 11: Specify the permissions of the new web site.

You need to specify the **Read** and **Run scripts** permissions. The **Run scripts** permissions are required to allow ASP pages to run. You can specify additional permissions but this is not required. Click **Next** to complete the web site Creation Wizard. The web site is now created as shown in the Internet Information Services (IIS) Manager.



Figure 12: The new web site is created and shown in the Internet Information Services Manager.

Finally, you need to configure the authentication method for the web site. In order to be able to check the access rights, the ASP pages must be executed in the security context of the end-user of the browser. Right click the web site and select **Properties**. Select **Directory security** and click the **Edit...** button in section **Authentication and access control**.



Figure 13: Specify the Authentication method: Disable anonymous access and enable integrated Windows authentication.

Disable the option **Enable anonymous access** and enable option **Integrated Windows authentication**. Click **OK** twice to exit the configuration.

Later, you can always change one of the configured options of the web site: Right click the web site and select **Properties**.

Configuring the UMRA project

The UMRA project is a form project that contains only a script, not a form. The UMRA form project is contained in file

.\Example Projects\Automation\ASP\CreateAccount\CreateAccount.ufp

relative to the UMRA Console program directory. The UMRA project is the same as the one used for the example project used for the Visual Basic script example. See section *Configuring the UMRA project* (on page 21) on page **Error! Bookmark not defined.** for more information.

Setting up the IIS web site

The UMRA project is now ready and the (empty) web site exists. You can now set up the web site contents to complete the configuration.

For demonstration purposes, the web site is kept very simple and contains only 2 pages:

CreateAccount.asp: The page that shows the input fields for first and last name and a button to submit the form. This is the first page the end-user connects to when accessing the web site.

ShowResults.asp: The page that actually creates the account using UMRA COM and shows the results (%UserName% and %Password%) to the end-user in the browser. The page is generated as a response when the end-user clicks the submit button of the previous page.

Since the web site pages contain ASP code, the pages have the .ASP extension. You can create the pages with your favorite editor or copy and edit the pages from the location

.\Example Projects\Automation\ASP\CreateAccount

relative to the UMRA Console program directory. The resulting ASP pages must be stored in the specified home directory of the new web site.

Note that an ASP page generates HTML code that is presented in a browser. Instead of the literal HTML text, the ASP page can contain script sections that produce HTML. The UMRA COM objects are used in the ASP script sections.

Web site page: ShowResults.asp

This page actually contains the UMRA COM objects and creates the user account. The contents of the page are listed below:

```
<HTML>
```

```
<HEAD>
```

```
<Title>User Account Created</TITLE>
```

```
</HEAD>
```

The user account created:

```
<%
```

```
    Set Umra =  
    Server.CreateObject ("UMRAcom.Umra")
```

```
    RetVal=Umra.Connect ("AMAZONE", 56814)
```

```
    Umra.SetVariableText  
    "%FirstName%", Request.Form ("FirstName")
```

```
    Umra.SetVariableText  
    "%LastName%", Request.Form ("LastName")
```

```
RetVal=Umra.ExecuteProjectScript ("CreateAccount")
```

```
RetVal=Umra.GetVariableText ("%UserName%", UserName  
)
```

```
    Response.Write "<BR>"
```

```
    Response.Write "User name: "
```

```
    Response.Write UserName
```

```
RetVal=Umra.GetVariableText ("%Password%", Password  
)
```

```

Response.Write "<BR>"

Response.Write "Password: "

Response.Write Password

%>

<FORM NAME=NextUser ACTION="CreateAccount.asp"
METHOD="POST">

<INPUT TYPE="SUBMIT" VALUE="Next user">

</FORM>

</HTML>

```

The page contains 3 sections:

1. the header section;
2. the ASP section
3. a form section.

Explanation of the ASP code

The header section is straightforward and contains the title of the page only. The ASP section contains the interesting part of the page and creates the user account and shows the results. The ASP section is described in detail below. The form section navigates the browser to the first page of the web site, CreateAccount.asp when the user clicks the Next user submit button.

Web site page line	Description
The user account created:	This is not part yet of the ASP section that starts with the <% characters. The text shown is simply copied to the HTML output and presented in the browser.
<%	Start of the ASP section. The lines that follow are ASP specified and terminated with the characters sequence %>.

<pre>Set Umra = Server.CreateObject("UMRAcom.Umra")</pre>	<p>The UMRA COM object is now created. The CreateObject method is a member of the ASP built-in Server object. The Server object is available in all ASP pages and offers a number of useful methods (function call). The Server.CreateObject method is the standard method in ASP to initiate an instance of a registered COM object. The argument of the method is the name of the library that holds the COM object (UMRAcom) and the name of the object (Umra). The Set Umra = ... construction creates an ASP script variable with name Umra and sets the value of the variable equal to the result of the Server.CreateObject method: the COM object. In the sequel of the ASP script, the COM object can now be used and must be referenced as the variable name: Umra.</p>
<pre>RetVal=Umra.Connect("AMAZ ONE",56814)</pre>	<p>The Connect method is part of the interface of the Umra COM object. It is used to setup a connection between the COM object itself and an UMRA Server. The method takes 2 arguments: the name of the computer and the port number used by the UMRA Service. The return value RetVal should be 0 on success and can be used for error handling purposes.</p>
<pre>Umra.SetVariableText "%FirstName%",Request.For m("FirstName")</pre>	<p>The method SetVariableText of the Umra COM object is used to initialize a variable-value pair. The COM object can hold a (single) list with multiple variable-value pairs. This list is sent to the UMRA Service when a project is executed. The SetVariableText method takes two arguments: the name of the variable (%FirstName%) and the value of the argument. In this case, the value is copied from the specified input field using the ASP Request object. The Form("FirstName") phrase refers to the input field with name FirstName of the original form of page CreateAccount.asp: <INPUT TYPE="TEXT" NAME="FirstName">. When this line of the script is executed, the list with variables stored in the Umra COM object holds the new variable-value pair.</p>
<pre>Umra.SetVariableText "%LastName%",Request.For m("LastName")</pre>	<p>Another variable-value pair is added to the list maintained by the UMRA COM object: The last name of the user account specified by the end-user in the form of web site page CreateAccount.asp is stored as variable %LastName% and copied from the input text field with name LastName.</p>

<code>RetVal=Umra.ExecuteProjectScript("CreateAccount")</code>	The interface member <code>ExecuteProjectScript</code> of the COM object is now used to execute the project script on the UMRA Service. The only argument of the member function is the name of the project. The current variable list stored in the COM object is used as the input of the form project. Note that the project script is not executed by the UMRA COM object. Instead, the UMRA COM object instructs the connected UMRA Service to execute the project. The <code>RetVal</code> numerical variable returns as zero on success. The value can be used for error handling. When the project script is executed successfully, the variable list of the COM object is updated with the values that are generated by the UMRA project script.
<code>RetVal=Umra.GetVariableText("%UserName%",UserName)</code>	The <code>GetVariableText</code> interface member function is used to obtain the text value of the specified variable. The variable should be part of the variable list of the UMRA COM object. On success, the <code>RetVal</code> value should be zero and the ASP script variable <code>UserName</code> is filled with the actual value of the variable.
<code>Response.Write "
"</code>	A break is written to the output HTML sequence that is generated by this ASP page.
<code>Response.Write "User name: "</code>	The text <code>User name: </code> is written to the HTML output.
<code>Response.Write UserName</code>	The value of the ASP variable <code>UserName</code> , as collected from UMRA variable <code>%UserName%</code> , written to the HTML output.
<code>RetVal=Umra.GetVariableText("%Password%",Password)</code>	The value of UMRA variable <code>%Password%</code> is searched for in the list maintained by the UMRA COM object. When found, the value is stored in ASP variable <code>Password</code> .
<code>Response.Write "
"</code>	A break is written to the HTML output.
<code>Response.Write "Password: "</code>	The text <code>Password: </code> is written to the HTML output.
<code>Response.Write Password</code>	The value of the password is written to the HTML output.
<code>%></code>	Termination of the ASP section. Normal HTML code follows after this character sequence or a new ASP section can start. In this example project, a small form is shown to navigate to the first page of the web site.

Table 3: Detailed description of the ASP page section using UMRA COM objects.

Testing the web site

The UMRA application is now ready for testing. Start Windows Internet Explorer and connect to the IIS web site by entering the URL address of the first web page:

http://amazone:81/CreateAccount.asp
(<http://amazone:81/createaccount.asp>)

Here, **amazone** is the name of the computer that runs IIS. The port for the web site is configured as 81. When automatic logon is enabled, the user (you) is authenticated by IIS and the HTML code of the ASP page is shown. If automatic logon is not enabled, you need to log on first.

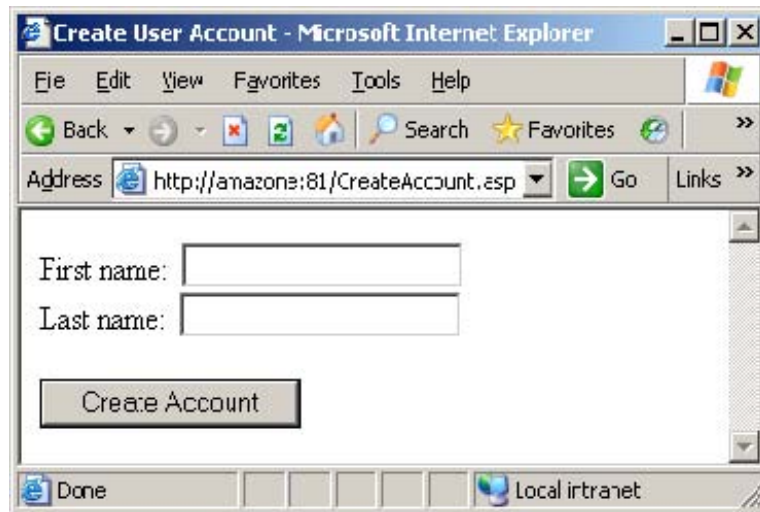


Figure 14: First page of the UMRA web site on IIS.

Enter the first and last name of a user account and click button **Create Account**. The form is submitted to IIS and the ASP page **ShowResults.asp** is executed.

As part of the ASP page, the UMRA Service is contacted and requested to execute the project script that creates the account. Results are returned by the service and shown in the web page.

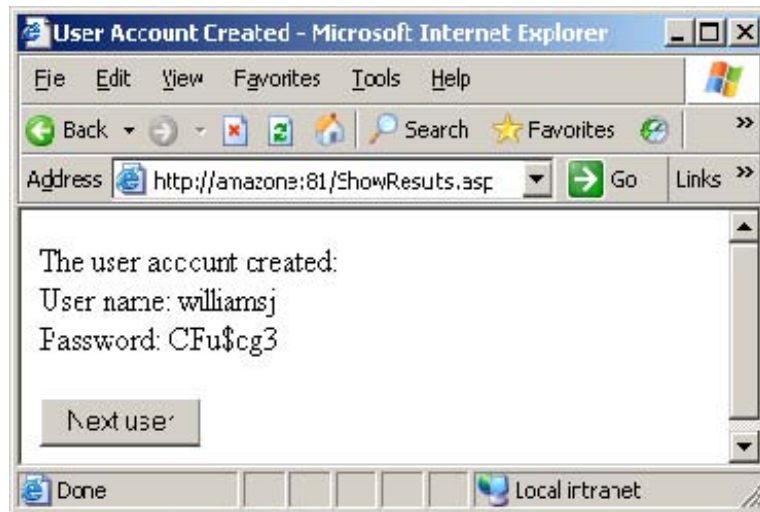


Figure 15: Result page of the UMRA web site.

You can check the UMRA Service log file for progress and debugging information. The log file **UmraSvcLogX.txt** can be found in the Log directory of the **UmraService** program directory.

UMRA Service log file

The COM object connects to the UMRA Service and the variables are listed in the log file. The %UmraFormSubmitAccount% variable shows the end-user of the browser.

The log file contains the following section upon completion of a successful session:

11:16:53 01/03/2006 Variable 1: %FirstName%=John

11:16:53 01/03/2006 Variable 2: %LastName%=Williams

11:16:53 01/03/2006 Variable 3:
%UmraFormSubmitAccount%=T4ELOC2\Administrator

11:16:53 01/03/2006 Creating AD account in specified domain:
'tools4ever.local2'.

11:16:53 01/03/2006 Creating AD account in container 'Users'.

11:16:53 01/03/2006 Creating AD account in Organizational Unit-Container:
'LDAP://CN=Users,DC=tools4ever,DC=local2'.

11:16:53 01/03/2006 Creating AD user account in container/OU
'LDAP://CN=Users,DC=tools4ever,DC=local2': Using name generation
algorithm 'Default', 100 iterations maximum for duplicate names.

11:16:53 01/03/2006 Creating AD user account in container/OU
'LDAP://CN=Users,DC=tools4ever,DC=local2'. Common name of user set to
'John Williams'.

11:16:53 01/03/2006 Creating AD user account in container/OU
'LDAP://CN=Users,DC=tools4ever,DC=local2'. SAM account name
(username) of user set to 'williamsj'.

11:16:53 01/03/2006 Creating AD user account in container/OU
'LDAP://CN=Users,DC=tools4ever,DC=local2'. LDAP attribute
'userPrincipalName' of object 'John Williams' set to
'williamsj@tools4ever.local2'.

11:16:53 01/03/2006 Creating AD user account in container/OU
'LDAP://CN=Users,DC=tools4ever,DC=local2'. LDAP attribute 'displayName'
of object 'John Williams' set to 'John Williams'.

11:16:53 01/03/2006 Creating AD user account in container/OU
'LDAP://CN=Users,DC=tools4ever,DC=local2'. LDAP attribute 'givenName'
of object 'John Williams' set to 'John'.

11:16:53 01/03/2006 Creating AD user account in container/OU
'LDAP://CN=Users,DC=tools4ever,DC=local2'. LDAP attribute 'sn' of object
'John Williams' set to 'Williams'.

11:16:53 01/03/2006 Creating AD user account in container/OU
'LDAP://CN=Users,DC=tools4ever,DC=local2'. Password generated ('Strong, 7 chars with 1 numeric, 1 special (variable: %Password%)') and set in variable '%Password%'.

11:16:53 01/03/2006 Creating AD user account in container/OU
'LDAP://CN=Users,DC=tools4ever,DC=local2'. Boolean parameter 'Account disabled'=FALSE (101034). Result not changed.

11:16:53 01/03/2006 Creating AD user account in container/OU
'LDAP://CN=Users,DC=tools4ever,DC=local2'. Boolean parameter 'Password never expires'=FALSE (101032). Result not changed.

11:16:53 01/03/2006 Creating AD user account in container/OU
'LDAP://CN=Users,DC=tools4ever,DC=local2'. Boolean parameter 'Store password using reversible encryption'=FALSE (101033). Result not changed.

11:16:53 01/03/2006 Creating AD user account in container/OU
'LDAP://CN=Users,DC=tools4ever,DC=local2'. Boolean parameter 'Smart card is required for interactive logon'=FALSE (101035). Result not changed.

11:16:53 01/03/2006 Creating AD user account in container/OU
'LDAP://CN=Users,DC=tools4ever,DC=local2'. Boolean parameter 'Account is trusted for delegation'=FALSE (101036). Result not changed.

11:16:53 01/03/2006 Creating AD user account in container/OU
'LDAP://CN=Users,DC=tools4ever,DC=local2'. Boolean parameter 'Account is sensitive and cannot be delegated'=FALSE (101037). Result not changed.

11:16:53 01/03/2006 Creating AD user account in container/OU
'LDAP://CN=Users,DC=tools4ever,DC=local2'. Boolean parameter 'Use DES encryption types for this account'=FALSE (101038). Result not changed.

11:16:53 01/03/2006 Creating AD user account in container/OU
'LDAP://CN=Users,DC=tools4ever,DC=local2'. Boolean parameter 'Don't require Kerberos preauthentication'=FALSE (101039). Result not changed.

11:16:53 01/03/2006 Creating AD user account in container/OU
'LDAP://CN=Users,DC=tools4ever,DC=local2'. Account expiration date not specified for new user 'John Williams' object.

11:16:53 01/03/2006 Creating AD user account in container/OU
'LDAP://CN=Users,DC=tools4ever,DC=local2'. User 'John Williams' successfully created.

11:16:53 01/03/2006 Creating AD user account in container/OU
'LDAP://CN=Users,DC=tools4ever,DC=local2'. Password set for new user object 'John Williams'.

11:16:53 01/03/2006 Creating AD user account in container/OU
'LDAP://CN=Users,DC=tools4ever,DC=local2'. Flag 'User Cannot Change Password' not changed from default value (FALSE).

11:16:53 01/03/2006 Form message:
'01/03/2006,11:16:53,T4ELOC2\Administrator,"Automation run
script",OK,CreateAccount'

Web site page: CreateAccount.asp

This is the first page of the web site. It does not contain UMRA COM objects but it contains the form and input fields that are needed to initialize the variables.

```
<HTML>

<HEAD>

<Title>Create User Account</TITLE>

</HEAD>

<FORM NAME=CreateAccount ACTION="ShowResults.asp"
METHOD="POST" >

First name: &nbsp;<INPUT TYPE="TEXT"
NAME="FirstName">

<BR>

Last name: &nbsp;<INPUT TYPE="TEXT"
NAME="LastName">

<BR>

<BR>

<INPUT TYPE="SUBMIT" VALUE="Create Account">

</FORM>
```

The page contains completely standard HTML. It contains a short header section (<HEAD>) and a form. The name of the form is CreateAccount and the action executed when the form is submitted is the generation and presentation of page ShowResults.asp.

The form contains two text input fields with names **FirstName** and **LastName**. The button is a submit type button with text **Create Account**.

This web site page does not contain any specified ASP code. It is all straightforward HTML.

The page shows a title, some text, two input fields and a button. When the button is pressed, the content of the input fields is sent to the IIS server and the page ShowResults.asp is generated and presented.

CHAPTER 5

References

Beginning ATL COM Programming, by Richard Grimes, Alex Stockton, George Reilly and Julian Templeman. Wrox Press Ltd, 1998

DCOM – Microsoft Distributed Component Object Model, by Frank E. Redmond III. IDG Books, 1997