Universal Messaging Installation Guide

Version 10.1

October 2017

This document applies to Universal Messaging Version 10.1 and to all subsequent releases.

Specifications contained herein are subject to change and these changes will be reported in subsequent release notes or new editions.

Document ID: NUM-IG-101-20190220

Table of Contents

1 Installation Overview

This guide describes how to install and configure the Universal Messaging product.

The guide contains the following information:

- How to perform the installation procedure.
- How to perform various configuration and administration tasks after you have installed the product.
- How to use the Instance Manager to create instances of realm servers, the Enterprise Manager and template applications.

2 Installation using the Software AG Installer

The Software AG Installer is a generic tool for installing Software AG products.

Universal Messaging can be licensed as part of a product bundle, so when you run the Software AG Installer, a dialog allows you to select the appropriate bundle. A subsequent dialog shows the individual products that you can install from the bundle, including Universal Messaging.

Overview of the Documentation for using the Software AG Installer

For the installation using the Software AG Installer, refer to the following documents:

- **Using the Software AG Installer**

  This document describes how to use the Software AG Installer tool. The usage of the Software AG Installer is the same for all products, so the documentation of the Software AG Installer does not refer explicitly to Universal Messaging.

  To access the document *Using the Software AG Installer*, do the following:

  1. Log in to the Software AG documentation web site at "http://documentation.softwareag.com/", using the Empower login ID and password that you have received by email when you licensed the product.
  2. Select the link for the Software AG Installer.
  3. The selected page lists several versions of the installer documentation, each shown with a release date. Select the version of the installer documentation that corresponds to the release date of the Software AG Installer you are using. The release date of the Software AG Installer is generally included in the file name of the downloaded executable file. You can also find the release date of the Software AG Installer by clicking the "About" link when you run the Software AG Installer.

- **Installing Software AG Products**

  This document contains specific installation information about many Software AG products, including Universal Messaging.

  The most recent version of the document *Installing Software AG Products* is available in the documentation web site using the following URL:

  "http://documentation.softwareag.com/webmethods/Installing_Software_AG_Products.htm"

- **Upgrading Software AG Products**

  If you are upgrading from a previous product version, refer also to this document. This document contains information about how to upgrade an existing Software AG product version to a new version.

The most recent version of the document *Upgrading Software AG Products* is available in the documentation web site using the following URL:

"http://documentation.softwareag.com/webmethods/Upgrading_Software_AG_Products.htm"

If you need to access versions of the documents *Installing Software AG Products* and *Upgrading Software AG Products* for previous product releases, proceed as follows:

1. From the starting page at "http://documentation.softwareag.com/", follow the "webMethods" link.
2. Navigate to the webMethods Product Suite, then select the Suite version number that matches the Universal Messaging version number.
3. Select the link that deals with installation topics.

3 Post-Installation Procedures

The information in the following sections describes procedures that you can use after the installation has completed.

Starting the Realm Server

The method you use to start the Universal Messaging realm server depends on the installation operating system.

For Windows operating systems, you can use the Start menu shortcut called *Start Universal Messaging Realm Server*.

For Linux/Solaris/Generic UNIX operating systems, starting the realm server can be done using the softlink inside the Server directory:

```
$ cd <Software_AG_directory>/UniversalMessaging/links/Server/<InstanceName>/
$ nohup ./Start\ Universal\ Messaging\ Realm\ Server  &
```

where <Software_AG_directory> is the product installation directory, and <InstanceName> is the name of the realm server instance. Note the use of the `nohup` command: if you issue these commands from a command window of a logged-on user, the realm server will continue to run even if the user logs off or the command window is terminated. Without the `nohup` command, the realm server will terminate when the command window is closed.

Alternatively, for all operating systems, you can start the realm server in a console window as follows:

1. Open a console window
2. Go to the `server/<InstanceName>/bin` directory
3. Run the command `nserver`

You can run the realm server as a Windows service or UNIX daemon. The product installation procedure allows you to install the default realm server as a service/daemon. Additionally, you can use the `nserverdaemon` script that is located in the `server/<InstanceName>/bin` directory. This script offers various options, as follows:

`nserverdaemon --install`	Install the realm server as a service/daemon. The service/daemon will be started automatically at every subsequent reboot of your machine.
`nserverdaemon --start`	Start the realm server as a service/daemon if it was not already running.

Note: On Windows, the options for `nserverdaemon` are preceded by a double hyphen "--". On UNIX systems, the double hyphens should be omitted. Also on Windows, there is a script `registerService.bat` that has the same effect as `nserverdaemon --install`.

The `nserverdaemon` script offers several other options relating to the realm server that are not detailed here. You can see the list by using the command `nserverdaemon --help`.

For an alternative method of registering the realm server to run as a UNIX daemon, refer to Appendix A of the generic Software AG installation guide "Installing Software AG Products".

For information about managing your realm server via Software AG's generic administration tool "Command Central", refer to the Command Central documentation.

The startup sequence

At this point the server should have started and is now ready for operation. To confirm this it is easy to check the realm server log file and ensure there are no errors being reported. Check the log file for a completion message such as the following:

```
[Mon Apr 07 10:59:22 BST 2014],Startup: Realm Server Startup sequence completed
```

As the realm server starts up it reports the current log level settings, currently set to 1000000 characters and a log level of 4. The log level was set during the installation, a level of 0 reports every action the server does and can roll log files every minute on a busy realm.

The realm server then reports the JRE environment and the currently installed security providers. These are important if running SSL and cause the majority of configuration problems. After this the server then reloads all the configuration parameters, channels and topics and then is ready to accept connections.

If there are any problems binding to a port or creating an SSL instance these exceptions are reported into the realm server log file with as much detail as the server can produce.

To test that the realm server is up and ready to accept connections, a simple test is to request information about the realm itself. You can do this as follows:

1. Open a client command prompt.

   To open a client command prompt on a Windows installation, click on the Command Prompt shortcut in your Start Menu.

   To open a client command prompt on a Linux/Solaris/Generic UNIX installation, open a console and source the Command Prompt script as follows:

   ```
   $ cd <Software_AG_directory>/UniversalMessaging/links/Client/<InstanceName>/
   $ ./Java\ Examples\ Command\ Prompt
   ```

   If the script does not take you automatically to the `java/<InstanceName>/bin` directory of your installation, change your working directory to `java/<InstanceName>/bin`.

2. Issue the command `ngetrealm`. This command returns a confirmation message if the server `<InstanceName>` is up and running.

Stopping the Realm Server

The method you use to stop the Universal Messaging realm server depends on the installation operating system.

For Windows operating systems, you can use the Start menu shortcut called *Stop Universal Messaging Realm Server*.

For Linux/Solaris/Generic UNIX operating systems, stopping the realm server can be done using the softlink inside the Server directory:

```
$ cd <Software_AG_directory>/UniversalMessaging/links/Server/<InstanceName>/
$ ./Stop\ Universal\ Messaging\ Realm\ Server
```

where <Software_AG_directory> is the product installation directory, and <InstanceName> is the name of the realm server instance.

Alternatively, for all operating systems, if you started the realm server by using the command `nserver`, use the following procedure in another console window to stop the realm server:

1. Open a console window
2. Go to the `server/<InstanceName>/bin` directory
3. Run the command `nstopserver`

If the realm server is running as a Windows service or UNIX daemon, you can stop it by using the `nserverdaemon` script that is located in the `server/<InstanceName>/bin` directory. This script offers various options, as follows:

`nserverdaemon --stop`	Stop the realm server if it is running as a service/ daemon.
`nserverdaemon --remove`	Uninstall the realm server as a service/daemon.

Note: On Windows, the options for `nserverdaemon` are preceded by a double hyphen "--". On UNIX systems, the double hyphens should be omitted. Also on Windows, there is a script `unregisterService.bat` that has the same effect as `nserverdaemon --remove`.

There as several ways of starting and stopping the realm server, and each way uses its own internal mechanisms. It's therefore important to always use the matching start/stop combination, such as:

- nserver / nstopserver
- nserverdaemon --start / nserverdaemon --stop
- StartUniversalMessagingRealmServer / StopUniversalMessagingRealmServer

- Command Central start / stop

The shutdown sequence

At this point the server will try and close down all client links and resources and P2P services. This may take some time to perform a graceful shutdown due to the Operating System resources that the realm server uses. The realm server will automatically perform a complete shutdown within 10 seconds if the graceful shutdown has not yet completed at which the realm server will perform an immediate shutdown. The realm server will generate a thread dump which will be written to the log file prior to shutdown.

The realm server will then log the shutdown within the server log and can be confirmed as complete by the following entries:

```
[Tue Apr 08 09:26:58 BST 2014],Shutdown: Realm Server shutdown sequence started
[Tue Apr 08 09:26:58 BST 2014],Shutdown: Removed all logger listeners
[Tue Apr 08 09:26:58 BST 2014],Shutdown: Disabling client requests
[Tue Apr 08 09:26:58 BST 2014],Shutdown: Stopping Realm status updates
[Tue Apr 08 09:26:58 BST 2014],Shutdown: Stopping Interface Manager
[Tue Apr 08 09:26:58 BST 2014],Shutdown: Closing all accept handlers
[Tue Apr 08 09:26:58 BST 2014],Shutdown: Closing all sessions with connected clients
[Tue Apr 08 09:26:58 BST 2014],Shutdown: Stopping Cluster management
[Tue Apr 08 09:26:58 BST 2014],Cluster> Cluster Agent restoring client request table
[Tue Apr 08 09:26:58 BST 2014],Cluster> Cluster Agent finished restoring client request table
[Tue Apr 08 09:26:58 BST 2014],--------- Log File Closed ---------
```

Server Memory Modes

Server Memory Modes

The performance and behaviour of the Realm Server is inseparably linked to the amount of maximum heap memory allocated to the Java VM hosting it. The Realm Server is capable of scaling depending on the hardware platform it is hosted on, and that is determined by the memory available to the Java VM. The Realm detects this and switches its mode of operation to *Small Memory Mode, Medium Memory Mode* or *Large Memory Mode*.

Small Memory Mode

Allocating 16MB or less of heap memory to the Java VM hosting the Universal Messaging Realm will make it operate in small memory mode. This is confirmed at start-up by a log entry like the following:

```
Audit,Setting Server mode to Small Memory Mode
```

The Universal Messaging Realm small memory mode should be used when running a Realm on mobile or embedded devices, or other machines with very limited memory resources available. Apart from limited memory available to store events in reliable channels, all thread pooled sub systems are changed to have only one thread. It is therefore recommended that persistent channels should always be used on such Realms. The performance will also be reduced by the fact that all caching is disabled in this mode.

All of the functionality provided by the innovative Universal Messaging server side Realms are available in the small memory mode and hence on handheld devices etc.

Medium Memory Mode

Allocating 16MB or more of heap memory to the Java VM hosting the Universal Messaging Realm will make it operate in medium memory mode. This is confirmed at start-up by a log entry like the following:

```
Audit,Setting Server mode to Medium Memory Mode
```

The Universal Messaging Realm medium memory mode should be used when running Realms on development or where memory is at a premium. All thread pooled sub systems will start up with our recommended default values for this mode. Tuning the Realm to higher values for those sub systems will increase the Realm's memory requirements and increase caching age values.

Large Memory Mode

Allocating 70MB or more of heap memory to the Java VM hosting the Universal Messaging Realm will make it operate in large memory mode. This is confirmed at start-up by a log entry like the following:

```
Audit,Setting Server mode to Large Memory Mode
```

The Universal Messaging Realm large memory mode should be used when running Realms on development, staging or production environments, or when using reliable channels. All thread pooled sub systems will start up with our recommended default values for this mode.. Tuning the Realm to higher values for those sub systems will increase the Realm's memory requirements and increase caching age values.

Client Deployment

Client Deployment

Depending on the functionality used by your Universal Messaging application, different jar files are required. This following table illustrates the deployment dependencies between the jar libraries installed by the Universal Messaging installer.

JAR File	Description	Dependency
nClient.jar	Provides Universal Messaging Client functionality (Pub/Sub and Queues)	None
nJMS.jar	Provides Universal Messaging Provider for JMS functionality	nClient.jar

JAR File	Description	Dependency
nAdminAPI.jar	Provides Universal Messaging Administration & Monitoring functionality	nClient.jar
nAdminXMLAPI.jar	Provides Universal Messaging Configuration XML Import / Export functionality	nClient.jar, nAdminAPI.jar
nEnterpriseManager.jar	Contains the Enterprise Manager tool	nClient.jar, nAdminAPI.jar, nAdminXMLAPI.jar (Optional)
nServer.jar	Contains the Universal Messaging Realm Server	None
Universal Messaging.tlb	Provides Universal Messaging Client Functionality for ActiveX applications	nClient.jar & Universal Messaging ActiveX Bridge

JMS Configuration

The Universal Messaging Realm server is designed to automatically support applications that use the provided Universal Messaging Provider for JMS. Such applications however need a valid JNDI context configuration in order to remain vendor neutral.

Universal Messaging features a JNDI service provider that can operate using any of the Universal Messaging transport protocols (nsp, nhp, nsps and nhps) as well as tools that allow configuration of the latter or any JNDI context is required from LDAP to NIS. The Universal Messaging JNDI provider uses a channel called / naming/defaultContext to store JMS references and the implementation class is com.pcbsys.nirvana.nSpace.NirvanaContextFactory.

JNDI configuration can be performed in 2 ways. The first is by using a command line application (with full source code provided) called JMSAdmin. For more information about how to use this application please check the appropriate developer guide section.

The second is by using the realm JNDI configuration panel in the Universal Messaging Enterprise Manager.

On the client side, you would need to set the standard java.naming.factory.initial key to the aforementioned com.pcbsys.nirvana.nSpace.NirvanaContextFactory and pass the JNDI provider URL of the realm under the nirvana.provider.url key or the standard java.naming.provider.url key. Note that the JNDI API suggests two ways of defining these properties - either by setting them as system properties, or by passing them in a JDNI environment Hashtable argument. When the NirvanaContextFactory is searching for the provider URL, it will by default first check the system properties, and if not found, it will consult the environment Hashtable argument. To reverse the order of lookup, you can set a system property key nirvana.provider.urlpref.sysprops to the value "N".

Command Prompts

Command Prompts

In order to make it easy for new users to test the rich functionality provided by Universal Messaging, the Universal Messaging installation includes compiled versions of all the sample applications as well as native launchers that wrap them and use environment variables to minimize the input required by the user. A command prompt is therefore a console/shell with some environment variables already set.

You require a *Client Command Prompt* to run sample client applications and sample administration (AdminAPI) applications with the minimum possible input.

You require a *Server Command Prompt* to run server side administration commands such as start a realm server, stop a realm server etc.

Client Command Prompt

The client command prompt is a console/shell with environment variables set by a client environment script. Examples of such environment variables include RNAME, PATH, CLASSPATH, Certificate stores etc.

On Windows platforms a client command prompt is opened using the shortcut in the start menu.

On Unix platforms it is opened by executing the appropriate softlink in the following folder:

```
$ cd <Software_AG_directory>/UniversalMessaging/links/Client/<InstanceName>/
```

where <Software_AG_directory> is the product installation directory, and <InstanceName> is the name of the realm server instance.

To open a command prompt to run the Java sample applications, use the following command:

```
$ ./Java\ Examples\ Command\ Prompt
```

Alternatively, to open a command prompt to run the C++ sample applications, use the following command:

```
$ ./C++\ Examples\ Command\ Prompt
```

Server Command Prompt

The server command prompt is a console/shell with environment variables set by a server environment script. Examples of such environment variables include PATH, CLASSPATH, server certificate stores etc.

On Windows platforms a server command prompt is opened using the shortcut in the start menu.

On Unix platforms it is opened by executing the appropriate softlink in the links folder under the install directory:

```
$ cd <Software_AG_directory>/UniversalMessaging/links/Server/<InstanceName>/
$ ./Command_Prompt
```

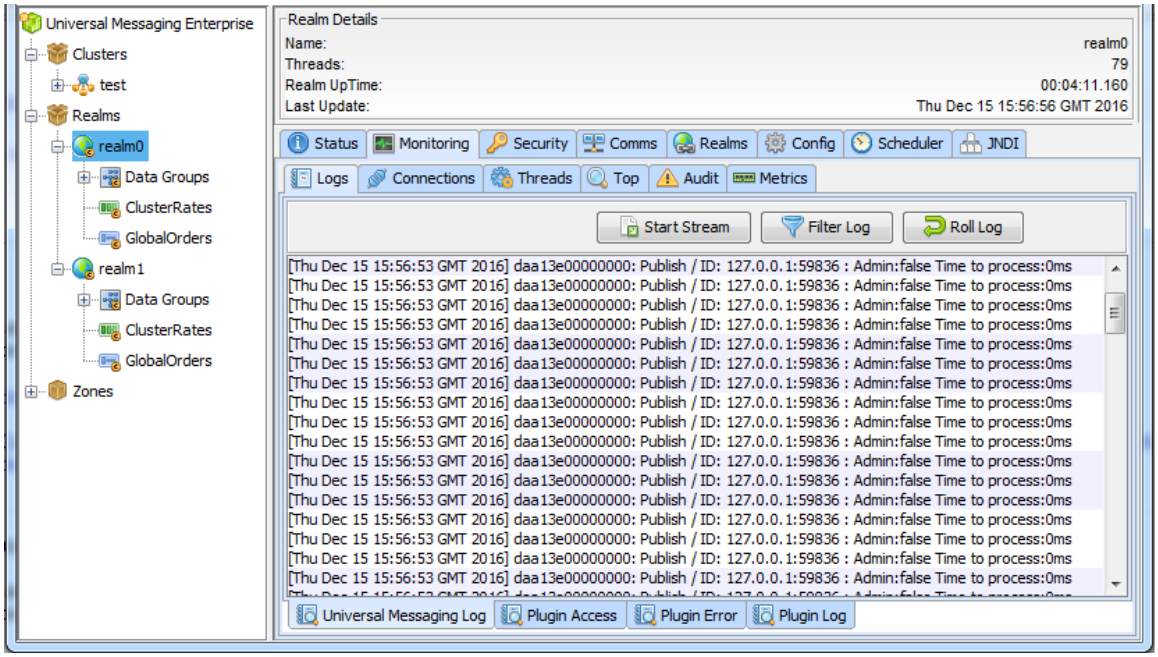
How to access the Universal Messaging log file

The Universal Messaging log file can be accessed using the Universal Messaging Enterprise Manager GUI by selecting the **Monitoring** tab and then the **Logs** tab. This provides access to the Universal Messaging log file itself as well as the log files associated with the Universal Messaging Realm Plugins. To switch between the available log files select the appropriate tab from the bottom of the logs panel.

See the description of the **Logs** panel in the documentation of the Enterprise Manager for more details about the log file.

Universal Messaging log files can also be accessed programmatically using the Universal Messaging Admin API.

If multiple realms have been added to the namespace then the log file for each can be accessed by clicking on that realm in the namespace and then selecting the **Monitoring** tab which will by default display the log panel.



How to Administer a Remote Universal Messaging Realm

A typical development setup involves installing a Universal Messaging Realm in a remote development server as well as the developer's workstation. This guide will help you connect to the remote development Universal Messaging realm for administration purposes.

A Universal Messaging realm by default enforces a security model that allows only the username running the server process to connect to it with full privileges, and that can only be done when connecting from localhost (127.0.0.1). Therefore, in order to be able to connect remotely you need to add an ACL entry for the user and the IP address you will connecting from. To do this you need to use the naddrealmacl command line tool on the development server as follows:

Windows OS Server Steps

1. Open a client command prompt from the Start Menu shortcut. (see "Command Prompts" on page 16)
2. Type "naddrealmacl <user> <ip> full", where <user> is the OS username that the development workstation will use to connect and <ip> is the ip address of the development workstation. In this instance we specify that full access should be given to this user.

Linux / Solaris Server Steps

1. Open a console window (shell)

2. Type "cd <install dir>/client/<realm_name>/bin", where <install_dir> is your installation path and <realm_name> is the name you assigned to the realm during installation.
3. Type "export RNAME=nsp://localhost:9000", this sets an environment variable called `RNAME` that all samples and command line tools use in order to know how to connect to the server. In this instance we are using the Universal Messaging Socket Protocol on localhost and port 9000. If you have chosen a different port please adjust accordingly.
4. Type " ./naddrealmacl <user> <ip> full", where <user> is the OS username that the development workstation will use to connect and <ip> is the ip address of the workstation. In this instance we specify that full access should be given to this user.

Development Workstation Steps

1. Run your enterprise manager on the development workstation and click on the Connections menu, selecting Connect To Realm.
2. A dialog will pop up asking you to specify the RNAME to use. Similarly to what we did for the command line tool, we specify "nsp://<server ip>:9000", where <server ip> is the IP address where the server is running and 9000 is the port the server is listening on. Again if you have chosen a different port please adjust accordingly
3. Click ok and you should see your realm appear in the tree under the Realms folder.

Upgrading from a Trial to a Production License

Note: Universal Messaging ships with a trial license, which allows the server to run for a maximum of 90 days from first run.

Trial Users

To purchase a production license, please contact us.

Production Users

If you already have a production license, and download a new build of the version for which you are licensed, then you should overwrite the shipped trial license with your production license to avoid being restricted to only 90 days' usage.

To do this, simply copy your production `licence.xml` over the trial `licence.xml` and restart your server.

The location of the license file is by default as follows:

<*Software_AG_directory*> /UniversalMessaging/server/<*realm_server_name*>

where <*Software_AG_directory*> is the default location for the installation of webMethods products, and <*realm_server_name*> is the name of the realm server to which the license applies.

If you encounter any problem with this process, please contact us for further support.

4 Universal Messaging Instance Manager

During the installation of Universal Messaging, you have the option of creating a default instance (called "umserver" by default) for all the components installed. If you need to create additional instances, this can be done using the ninstancemanager command line tool, which can be found under <*Software_AG_directory*> /UniversalMessaging/tools/InstanceManager/.

Components

The ninstancemanager tool can create instances of realm servers (RS), Enterprise Manager (EM) and template applications (TA). In order to create an instance of a component, this needs to have been installed first.

Usage Message

Executing the ninstancemanager tool without any arguments provides a usage message as follows:

```
ninstancemanager <Action> <InstanceName> <Component> <Host> <Port> [DataDirectory]
```

- <Action> can be either create, delete, query, deleteAll (followed directly by a component), or configure.
- <InstanceName> can be any instance name.
- <Component> is the component the action applies on, namely RS (for Realm Server), EM (for Enterprise Manager), TA (for Template Application) or ALL (for everything installed).
- <Host> is the hostname or IP that the template apps & Enterprise manager will point to, and the adapter the realm will bind to.

  You can use the hostname instead of the IP when you wish to provide an environment that is not specific to the underlying IP address of the server. This will allow the UM server to be accessed only by its hostname, so if the IP address changes, the server will still be accessible.
- <Port> is the TCP port that the template apps & Enterprise Manager will point to, and the adapter the realm will bind to.
- <DataDirectory> is the realm server working directory. This parameter is optional, and the default value is "UniversalMessaging/server/<*InstanceName*>".

Example 1: To create a new instance called umserver2 and listening to all IPs on port 9001, you would run:

```
ninstancemanager create umserver2 all 0.0.0.0 9001
```

Example 2: To create a new EM instance called umserver2 and pointing to a realm on 192.168.1.100 port 9001 you would run:

```
ninstancemanager create umserver2 all 192.168.1.100 9001
```

Example 3: To delete all instances called umserver2 you would run:

```
ninstancemanager delete umserver2 all
```

Example 4: To delete an EM instance called umserver2 you would run:

```
ninstancemanager delete umserver2 em
```

Example 5: To query installed instances you would run:

```
ninstancemanager query
```

Example 6: To delete all realm server instances you would run:

```
ninstancemanager deleteAll rs
```

Querying Installed Instances

Running the ninstancemanager tool with the query action displays a list of currently installed instances. For example:

```
ninstancemanager query
```

will display an output similar to the following in a default installation (taking release 9.8.0 as an example):

```
Universal Messaging installation query
--------------------------------------
Realm Server Installed Version:
e2ei/11/NUM_9.8.0.0.13321/UniversalMessaging/NUMRealmServer
Instances: umserver
Enterprise Manager Installed Version:
e2ei/11/NUM_9.8.0.0.13321/UniversalMessaging/NUMEnterpriseManager
Instances: umserver
Template Applications Installed Version:
e2ei/11/NUM_9.8.0.0.13321/UniversalMessaging/NUMTemplateApplications
Instances: umserver
```

Creating Instances

Running the ninstancemanager tool with the create action allows you to create instances of all the installed components or a subset. In order to create an instance, you need to run the ninstancemanager as follows:

```
ninstancemanager create <InstanceName> <Component> <Host> <Port>
```

Where:

- <InstanceName> is a logical name for the instance which needs to be unique for each installation.
- <Component> is the component you wish to create an instance of. The possible values are ALL (for all components installed), RS (for a realm server instance), TA (for template applications instance) or EM (for enterprise manager instance).

Example: If we wanted to create an instance of all components installed called testinstance, bound to all IPs of the machine and listening on port 9002 you would enter:

```
ninstancemanager create testinstance all 0.0.0.0 9002
```

Output:

```
Created RS instance testinstance
```

```
Created TA instance testinstance
Created EM instance testinstance
```

You can then verify the instance's presence by issuing a query action:

```
ninstancemanager query
```

Output:

```
Universal Messaging installation query
--------------------------------------
Realm Server Installed Version:
e2ei/11/NUM_9.8.0.0.13321/UniversalMessaging/NUMRealmServer
Instances: testinstance , umserver
Enterprise Manager Installed Version:
e2ei/11/NUM_9.8.0.0.13321/UniversalMessaging/NUMEnterpriseManager
Instances: testinstance , umserver
Template Applications Installed Version:
e2ei/11/NUM_9.8.0.0.13321/UniversalMessaging/NUMTemplateApplications
Instances: testinstance , umserver
```

Deleting Instances

The ninstancemanager tool can be used to delete any instances created, including the default instance created using the installer. The components specified allow you to remove an instance for one component while keeping it for the others.

In order to delete an instance, you need to run the ninstancemanager as follows:

```
ninstancemanager delete <InstanceName> <Component>
```

Where:

- `<InstanceName>` is a logical name for the instance which needs to be unique for each installation.
- `<Component>` is the component you wish to create an instance of. The possible values are ALL (for all components installed), RS (for a realm server instance), TA (for template applications instance) or EM (for enterprise manager instance).

Example: If we wanted to delete a previously created instance of all components called testinstance, you would enter:

```
ninstancemanager delete testinstance all
```

Output:

```
RS instance testinstance has been deleted
TA instance testinstance has been deleted
EM instance testinstance has been deleted
```

You can then verify the instance's presence by issuing a query action:

```
ninstancemanager query
```

Output:

```
Universal Messaging installation query
--------------------------------------
Realm Server Installed Version:
e2ei/11/NUM_9.8.0.0.13321/UniversalMessaging/NUMRealmServer
Instances: umserver
Enterprise Manager Installed Version:
e2ei/11/NUM_9.8.0.0.13321/UniversalMessaging/NUMEnterpriseManager
```

```
Instances: umserver
Template Applications Installed Version:
e2ei/11/NUM_9.8.0.0.13321/UniversalMessaging/NUMTemplateApplications
Instances: umserver
```

Deleting All Instances of a Component

You can delete all instances of a component (RS, EM, TA or ALL) by using the "deleteAll" action and passing the component:

Example: If we wanted to delete all previously created instances of the component type "template application" (TA), you would enter:

```
ninstancemanager deleteAll ta
```

Output:

```
TA instance umserver has been deleted
```

You can then verify that the instance(s) have been deleted by issuing a query action:

```
ninstancemanager query
```

Output:

```
Universal Messaging installation query
--------------------------------------
Realm Server Installed Version:
e2ei/11/NUM_9.8.0.0.13321/UniversalMessaging/NUMRealmServer
Instances: umserver
Enterprise Manager Installed Version:
e2ei/11/NUM_9.8.0.0.13321/UniversalMessaging/NUMEnterpriseManager
Instances: umserver
Template Applications Installed Version:
e2ei/11/NUM_9.8.0.0.13321/UniversalMessaging/NUMTemplateApplications
Instances:
```

Configuring an existing Server Instance

The ninstancemanager command line tool can be used to configure existing server instances.

It provides two ways to import a configuration for an instance:

- Import a configuration from a predefined profile. These predefined profiles are shipped with the installation. The configure command allows you to view the available predefined profiles.
- Import a custom configuration XML file that you have previously exported. This option can be used to apply a specific configuration, unique for a separate customer use case.

Note:

- In both cases, before the import is performed, a backup of the existing configuration will be made. Backups will be created under <Software_AG_directory>/UniversalMessaging/server/<InstanceName>/data/configBackup, where <InstanceName> is the name of your server instance.

- The server instance needs to be stopped when importing a configuration, otherwise an error message will be printed and the configuration will not be imported.

Displaying the command's help text

You can use the following command to display the command usage help text:

```
ninstancemanager configure
```

Listing available predefined profiles

To list information about currently available profiles, use the following command:

```
ninstancemanager  configure  --listProfiles
```

The command lists information about the available predefined profiles. A short description about each of the profiles will be also printed.

Currently two predefined profiles are available: wM (for webMethods) and TC (for Terracotta). Each profile contains a set of Universal Messaging server configuration values that fits best to the profile's use case.

Profile name	Description
wM	webMethods suite use cases (large messages, transactions, persistence)
TC	Standalone use cases (small messages, non-transactional, transient data

Importing a predefined profile

You can import a predefined configuration profile. Use the --listProfiles command as shown above to list the available predefined profiles.

The command to import a predefined profile is as follows:

```
ninstancemanager configure umserver --importProfile=<ProfileName>
    --dataDir=<Software_AG_directory>/UniversalMessaging/server/<InstanceName>
```

Here, <ProfileName> is the name of the profile you wish to import.

The parameter <InstanceName> is the name of the server instance.

The parameter `--dataDir` is optional. If you do not supply this parameter, the default base data folder is assumed, which is <Software_AG_directory>/UniversalMessaging/server/umserver.

Importing a custom configuration file

You can import a custom configuration file using a command of the following form (note the use of the parameter --import rather than --importProfile):

```
ninstancemanager configure umserver --import=<customConfigFile>
```

```
    --dataDir=<Software_AG_directory>/UniversalMessaging/server/<InstanceName>
```

For example:

```
ninstancemanager configure umserver --import=c:\myConfig\myCustomConfig.xml
    --dataDir=<Software_AG_directory>/UniversalMessaging/server/umserver
```

Exporting configuration to a file

You can export the current configuration into an external file, using a command of the form:

```
ninstancemanager  configure <InstanceName>  --export=<path>
```

For example:

```
ninstancemanager  configure umserver  --export=c:\myBackup\myBackup.xml
```

The command will export the current configuration in a file, which afterwards can be imported with the `--import` command.

Note: The export command can be run even if the server instance is running.